Universiteti i Europës Juglindore
Fakulteti i Shkencave dhe i Teknologjive të Komunikimit

Agni Dika

Bazat e Programimit në C++

2005

U lejua për botim nga Komisioni për Botime pranë Universitetit të Europës Juglindore në Tetovë.

Parathënie

Libri të cilën e keni në dorë, së pari u dedikohet studentëve të vitit të parë të cilët studjojnë në Fakultetin e Shkencave dhe të Teknologjive të Komunikimit në Universitetin e Europës Juglindore në Tetovë. Por, sigurisht, ai mund të shfrytëzohet edhe nga studentë të universiteteve të tjera, ose edhe nga të gjithë ata që dëshirojnë ta përvetësojnë teknikën e programimit me kompjuter.

Materiali i përfshirë në libër është shkruar ashtu që lexuesi t'i bëjë hapat e parë në mësimin e gjuhës programuese `C++`, e cila gjendet në përbërje të pakos me programe `Microsoft Visual Studio .NET`. Gjatë kësaj, një dije paraprake lidhur me algoritmet që shfrytëzohen gjatë zgjidhjes së problemeve të ndryshme me kompjuter, sigurisht se mund ta lehtësoi kuptimin e programeve të shumtë të cilët jepen si shembuj.

Lexuesi mund komunikojë me autorin përmes adresave elektronike **`agnidika@yahoo.com`** dhe **`a.dika@see-university.edu.mk`**, ose edhe përmes lidhjes që ofrohet në faqen e internetit **`www.agnidika.net`**.

Autori

Përmbajtja

5. Unazat

6. Funksionet

1

Hapat e parë në C++

Për shkruarje të programeve të ndryshme në kompjuter, aktualisht shfrytëzohen disa gjuhë programuese. Por, njëra ndër gjuhët programuese më të përhapura është gjuha programuese C++. Këtu, me qëllim të përvetësimit të teknikës së programimit në këtë gjuhë, do të shfrytëzohet versioni 7.0 i saj, që gjendet në përbërje të pakos me programe `Visual Studio .NET`, i kompanisë softverike *Microsoft*. Në kuadër të pakos në fjalë gjendet edhe e ashtuquajtura *Rrethinë Zhvilluese e Microsoftit* (ang. Microsoft Development Environment, ose shkurt `MDE`), përmes së cilës programet në këtë gjuhë shkruhen, editohen, kompajlohen, ekzekutohen, ruhen në disk, thirren nga disku etj.

Programi i shkruar në gjuhën programuese `C++` paraqet *program burimor* dhe si i tillë ruhet në një *fajll burimor*. Me procesin e kompajlimit, kontrollohet saktësia e komandave të shkruara dhe nëse nuk ka gabime, programi burimor përkthehet në *program objektiv*, gjë që shihet edhe në paraqitjen skematike të dhënë në *Fig.1.1*.

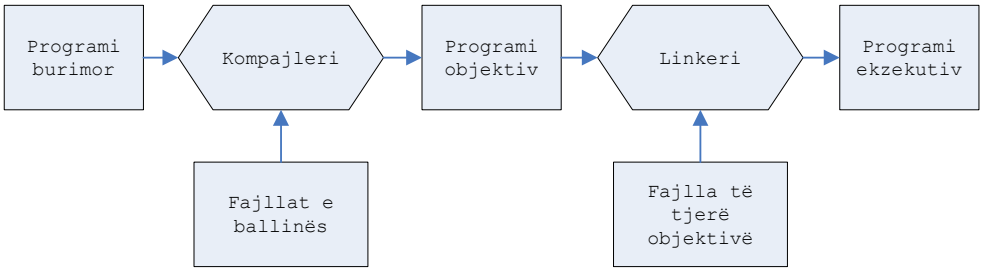


Fig.1.1 Procesi i krijimit të programit ekzekutiv nga programi burimor

Njëkohësisht, gjatë procesit të kompajlimit, në program insertohen edhe fajllat e ballinës, të cilët shënohen në fillim të programit burimor. Në fund, për ta fituar *programin ekzekutiv*, përmes *linkerit* fajllit objektiv i shtohen edhe fajlla të tjerë objektivë, të marrë nga biblioteka standarde ose edhe nga biblioteka të tjera me fajlla. Programi ekzekutiv ruhet në një *fajll ekzekutiv*.

Në gjuhën programuese C++, fajllat burimorë zakonisht e kanë prapashtesën .cpp, fajllat e ballinës - prapashtesën .h, ose janë pa prapashtesë, fajllat objektiv - .obj, kurse fajllat ekzekutivë dallohen me prapashtesën .exe.

Aktivizimi i Visual Studios

Për aktivizimin e *Visual Studios*, në desktopin e kompjuterit duhet të klikohet ikona përkatëse. Nëse kjo ikonë nuk është e vendosur në desktop, për aktivizimin e *Visual Studios*, si edhe gjatë aktivizimit të programeve të tjera, mund të shfrytëzohet pulla Start e *Windowsit* dhe pastaj në menynë rritëse përkatëse – opcioni Programs si dhe nënopcioni Microsoft Visual Studio .NET. Pavarësisht se cila nga dy mundësitë e përmendura shfrytëzohet për aktivizimin e Visual Studios, si rezultat do të hapet ekrani kryesor i tij, i cili fillimisht mund ta ketë pamjen e dhënë në *Fig.1.2*.

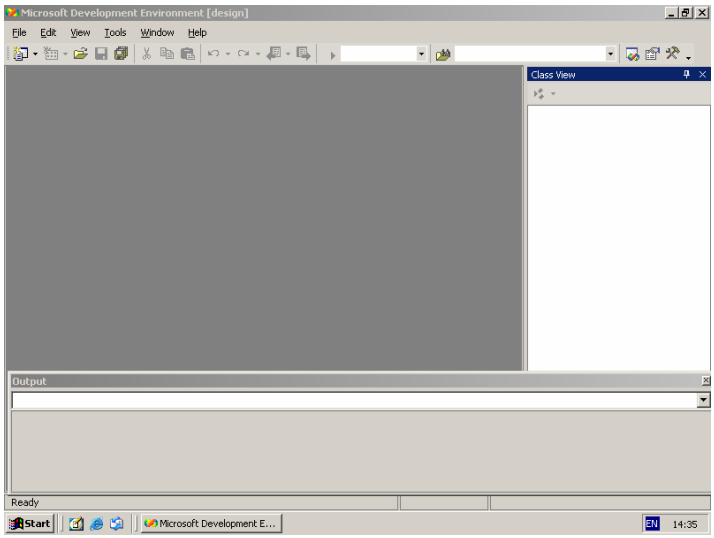


Fig.1.2
Pamja fillestare e ekranit kryesor

Në rreshtin e parë të ekranit kryesor gjendet *menya kryesore*, e cila në versionin fillestar të saj duket si në *Fig.1.3*.

Fig.1.3 Menya kryesore

Si zakonisht, opcionet e veçanta të menysë kryesore mund të zgjedhen duke i klikuar ato me mi. Por, ato mund të zgjedhen edhe pasi paraprakisht të pozicionohemi në këtë meny, duke e shfrytëzuar në tastierë tastin Alt. Pastaj,

përmes tasteve me shigjeta për lëvizje majtas e djathtas, pozicionohemi në opcionin e dëshiruar, kurse për aktivizimin e tij duhet të shtypet tasti `Enter`. Opcioni i dëshiruar zgjedhet edhe pasi në tastierë të mbahet i shtypur tasti `Alt` dhe njëkohësisht shtypet tasti i shkronjës së nënvizuar tek ai opcion. Kështu, p.sh., për zgjedhje të opcionit `Tools`, pasi në tastierë të mbahet i shtypur tasti `Alt`, duhet të shtypet tasti me shkronjën `T`, gjë që shkurt shkruhet edhe si `Alt+T`.

Hapja e një projekti të ri

Programimi në gjuhën `C++` fillon me *hapjen e një projekti të ri*. Për këtë qëllim, në menynë kryesore zgjedhet opcioni `File`, për t'i fituar nënopcionet e menysë rënëse përkatëse ashtu siç shihet në *Fig.1.4*.

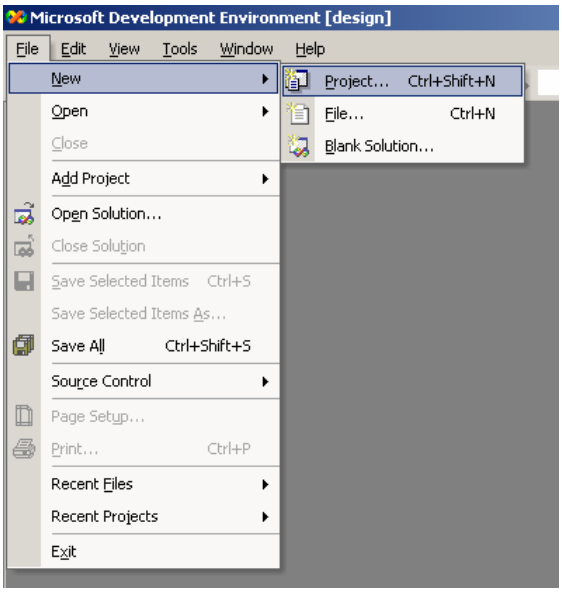


Fig.1.4
Nënopcioni New i menysë rënëse të opcionit File

Pastaj, në menynë rënëse të opcionit `New`, zgjedhet nënopcioni `Project...` (zgjedhja mund të bëhet edhe përmes kombinimit të tasteve `Ctrl+Shift+N` në tastierë), pas së cilës si rezultat hapet dritarja *New Project*, e cila shihet në *Fig.1.5*.

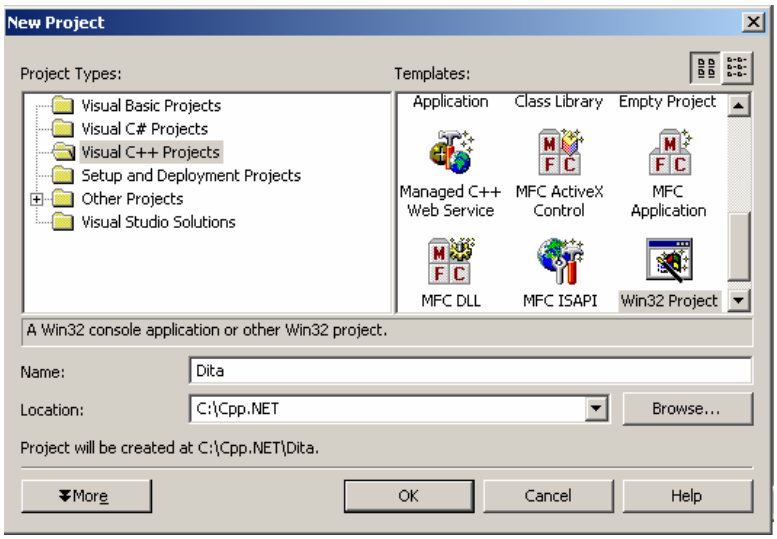


Fig.1.5 Dritarja New Project

Në pjesën e majtë të dritares duhet të zgjedhet gjuha në të cilën do të realizohet projekti i ri, ku, mes të tjerash, si opcione ofrohen gjuhët `Visual Basic`, `Visual C#` dhe `Visual C++`. Kurse, në pjesën e djathtë të dritares, meqë duam të shkruajmë programe të zakonshme në gjuhën `C++`, e zgjedhim opcionin `Win32Project`.

Në fund të dritares duhet të përcaktohet emri i projektit, duke e shkruar atë në kornizën para së cilës është shënuar teksti `Name:`. Njëkohësisht, në kornizën `Location:` zgjedhet edhe folderi i diskut në të cilin do të ruhet projekti. Këtu, si shembull, projekti është quajtur me emrin `Dita` dhe vendoset në folderin `C:\Cpp.NET`. Zgjedhja e folderit të dëshiruar mund të bëhet duke e klikuar pullën `Browse...`, pas së cilës në ekran paraqitet pema e folderëve në diskun aktual, ku mund të zgjedhet njëri prej tyre.

Pasi të zgjedhet emri i projektit dhe folderi ku ai do të ruhet, procedura e hapjes së një projekti të ri mund të vazhdojë, nëse klikohet pulla `OK`. Si rezultat, aktivizohet magjistari për gjenerim të projektit të zgjedhur, me dritaren e cila shihet në *Fig.1.6*.

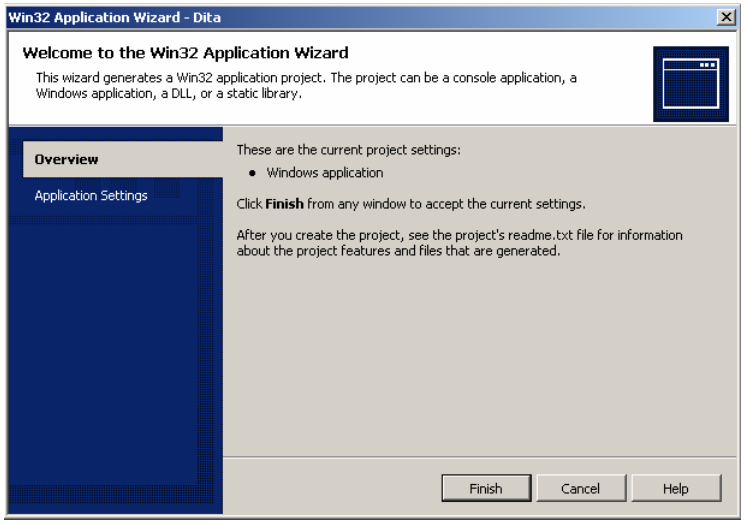


Fig.1.6
Dritarja e magjistarit për projektin e ri

Për zgjedhje të tipit të projektit, në dritare duhet të klikohet pulla `Application Settings`, pas së cilës fitohen opcionet që shihen në *Fig.1.7*.

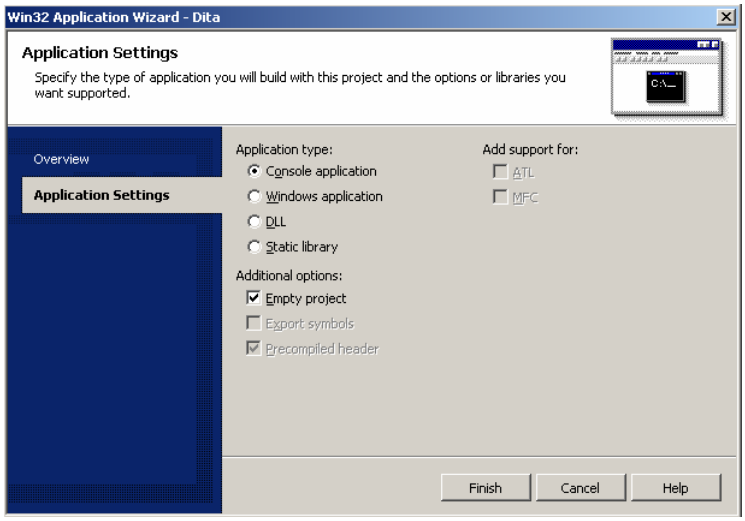


Fig.1.7
Zgjedhja e tipit të projektit

Kompjuteri, për tipin e projektit (shih pjesën mbi të cilën shkruan *Application type:*) na e propozon opcionin `Windows application`. Por, meqë duam t'i bëjmë hapat e parë në shkruarje të programeve në gjuhën `C++`, duhet ta zgjedhim opcionin `Console application`. Njëkohësisht, në pjesën mbi të cilën shkruan *Additional options:* duhet ta zgjedhim opcionin `Empty project`.

Në fund, pasi të klikohet pulla `Finish`, aktivizimi i një projekti të ri përfundon dhe si rezultat hapet *ekrani kryesor* i gjuhës `C++`, ashtu siç shihet në *Fig.1.8*.

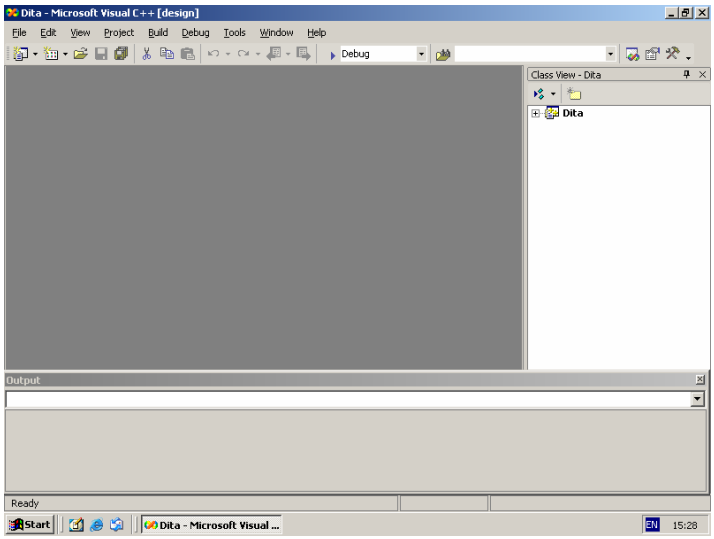


Fig.1.8
Ekrani kryesor i një projekti të ri në gjuhën C++

Këtu, në fakt, dritarja *Class View*, e cila gjendet në pjesën e djathtë të ekranit bëhet dritare aktive. Por, sipas nevojës, kjo dritare dhe dritarja *Output* në fund të ekranit, mund të eliminohen nga ekrani, ose atyre mund t'u ndryshohet pozita në ekran si dhe dimensionet, duke i shfrytëzuar procedurat që shfrytëzohen gjatë punës nën Windows. Për këto arsye, pamja e dritareve në ekranin kryesor të gjuhës C++, përkatësisht pamja e ekranit kryesor, mund të jetë edhe ndryshe nga ajo që është dhënë në *Fig.1.8*. Zakonisht, ekrani kryesor mund ta ketë pamjen edhe pa dy dritaret që u përmendën më sipër.

Pas hapjes së projektit, menya kryesore më nuk është ajo që u dha në *Fig.1.3*, por tani duket si në *Fig.1.9*.

Fig.1.9 Menya kryesore pas hapjes së projektit

Për hapje të një projekti të ri, siç u tha edhe më sipër, mund të shfrytëzohet kombinimi i tasteve `Ctrl+Shift+N`, ose edhe pulla `New Project`, e cila gjendet në fillim të shiritit me vegla dhe duket ashtu siç është treguar majtas.

Hapësira punuese e programit

Shkruarja e një programi në kuadër të projektit fillon me hapjen e *hapësirës punuese* përkatëse. Për këtë qëllim, në menynë rënëse të opcionit `File` zgjedhet nënopcioni `Add New Item...`, ashtu siç shihet në *Fig.1.10*.

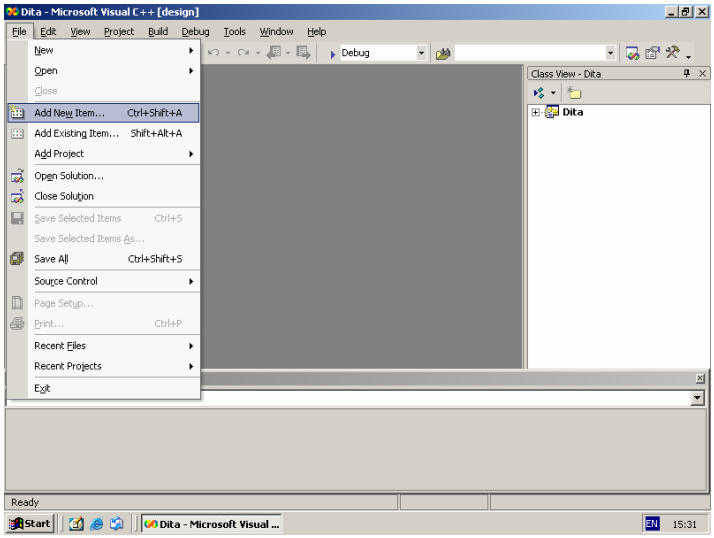


Fig.1.10
Hapja e hapësirës punuese për një program të ri

Si rezultat i opcionit të zgjedhur, në ekran do të paraqitet dritarja *Add New Item*, e dhënë në *Fig.1.11*.

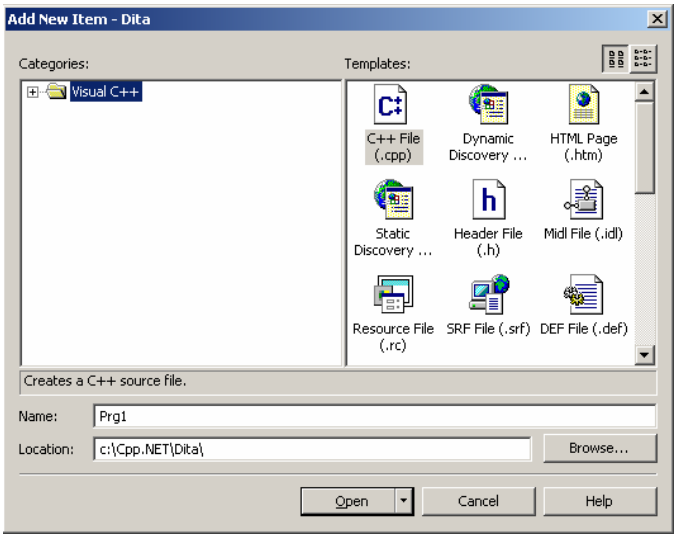


Fig.1.11
Dritarja Add New Item

Pas kësaj, në pjesën e majtë të dritares duhet të zgjedhet opcioni `Visual C++`, kurse në pjesën e djathtë të saj – opcionin `C++ File (.cpp)`. Njëkohësisht, në kornizën para së cilës gjendet teksti `Name:`, duhet shënuar emrin e programit të ri, p.sh., `Prg1`, i cili do të vendoset në përbërje të projektit `Dita`. Në fund, pasi të klikohet pulla `Open`, hapet *hapësira punuese e programit* mbi të cilën shkruan `Prg1.cpp` (shih *Fig.1.12*), ku mund të fillojë shkruarja e programit.

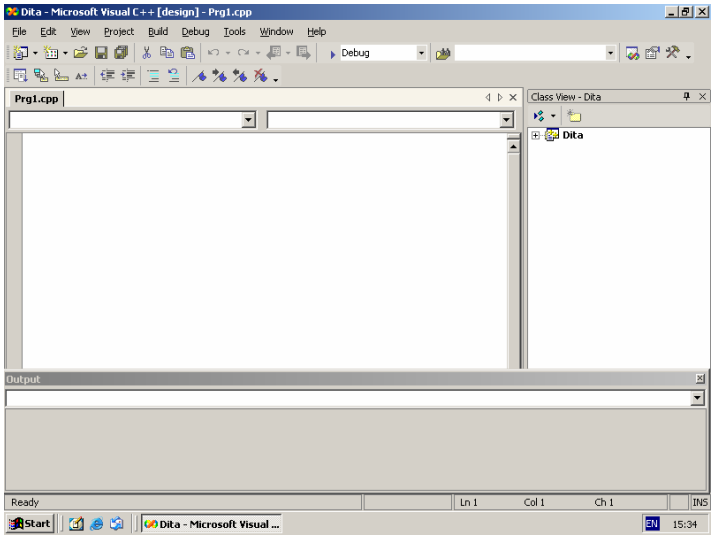


Fig.1.12
Pamja e ekranit me hapësirën punuese të programit Prg1.cpp

Hapësira punuese e një programi të ri mund të hapet edhe duke e zgjedhur nënopcionin `Add New Item...`, në menynë rënëse të opcionit `Project`, gjë që shihet në Fig.1.13.

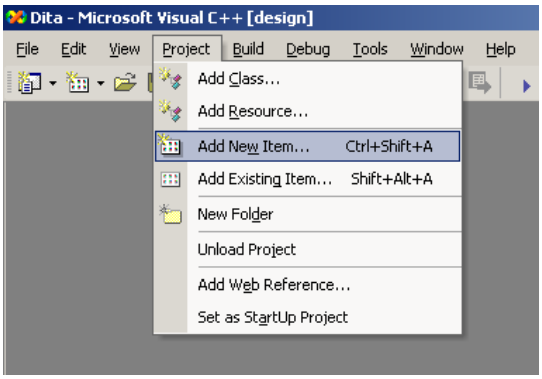


Fig.1.13
Opcioni për hapje të hapësirës punuese për shkruarje të programit

Por, procedura e hapjes së hapsirës punuese të programit që shkruhet në kuadër të projektit mund të startohet edhe përmes kombinimit të tasteve `Ctrl+Shift+A`, ose duke e klikuar në shiritin me vegla pullën `Add New Item`, e cila shihet majtas.

Shkruarja e programit

Çdo program në gjuhën C++ patjetër duhet ta përmbajë *funksionin* `main()`, të cilit në fund të programit i shoqërohet edhe komanda `return 0`. Programi më i thjeshtë në gjuhën C++ mund të shkruhet vetëm me këto dy komanda dhe do të duket:

```
int main()
{
      return 0;
}
```

Komanda `return 0` e paraqet komandën e fundit të funksionit `main()`, kurse vlera `0` e shënuar në vazhdim të kësaj komande shfrytëzohet për të treguar se programi ka përfunduar pa gabime.

Nëse ekzekutohet programi i dhënë, si rezultat nuk do të fitohet asgjë, sepse në pjesën ku shkruhen komandat, përkatësisht në bllokun brenda kllapave të mëdha, ose siç thuhet - brenda *trupit të programit*, nuk ka asnjë komandë që jep ndonjë rezultat.

Programi në fjalë fizikisht mund të shkruhet më shkurt:

```
int main()
{return 0;}
```

ose edhe vetëm në një rresht:

```
int main() {return 0;}
```

Për shkak të dukshmërisë më të mirë, në gjuhën C++ praktikohet që programet të shkruhen si në formën e parë, ku çdo komandë shkruhet në një rresht.

Shembull Programi përmes së cilit në ekran shtypet teksti `Programi i parë në C++`.

```
// Programi Prg1
#include <iostream>
using namespace std;
int main()
{
      cout << "Programi i parë në C++"
           << endl;
      return 0;
}
```

Këtu, brenda trupit të programit (nën kllapat e mëdha) gjendet komanda `cout`, përmes së cilës në ekran shtypet fjalia:

```
Programi i parë në C++
```

që është shënuar nën thonjëza.

I shkruar në hapësirën punuese përkatëse, programi `Prg1` do të duket si në *Fig.1.14*.

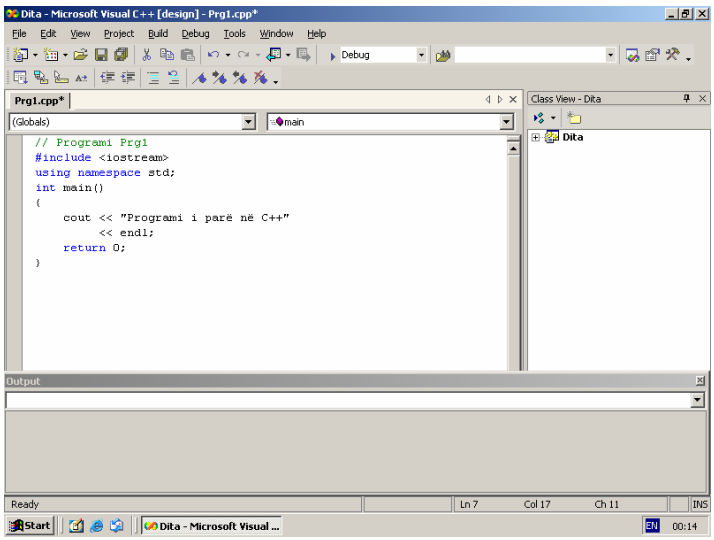


Fig.1.14
Programi Prg1 i shkruar në hapësirën punuese të ekranit

Në ekran, komandat e shkruara saktë kompjuteri i ngjyros me ngjyrë të kaltër. Brenda trupit të programit, në fund të çdo komande shënohet pikëpresja (`;`) për të treguar fundin e saj, përkatësisht ajo shërben si *terminator i komandës*.

Me dy vijat e pjerrëta `//` në rreshtin e parë të programit, kompjuteri njoftohet se nuk kemi të bëjmë me ndonjë komandë, por me koment të cilin e shfrytëzon vetë përpiluesi i programit.

Komanda në rreshtin e dytë të programit fillon me simbolin # (shenja për numër) dhe paraqet një *direktivë paraprocesorike* (ang. preprocessor directive). Kjo komandë i jepet pjesës së *kompajlerit* (ang. compiler), e cila quhet *paraprocesor* (ang. preprocessor), për ta insertuar (ang. include) në fillim të programit fajllin `iostream`, para se të fillojë kompajlimi i tij. Ky fajll bën pjesë në fajllat e ballinës (të cilët përpunohen nga procesori para se të kompajlohet programi) dhe lidhet me komandat për lexim dhe shtypje (nga Input-Output-Stream), p.sh., siç është në program komanda për shtypje `cout`.

Direktiva `using`, e cila gjendet në rreshtin e tretë të programit, e njofton kompjuterin se komandat që shfrytëzohen në program gjenden në *hapësirën me emra* `std`. Nëse nuk shfrytëzohet direktiva `using`, para çdo komande duhet të shënohet `std::`, p.sh., kështu:

```
std::cout << "Programi i parë në C++"
          << endl;
```

gjë që ndikon në optimizimin e hapësirës memoruese në të cilën ruhet programi.

Me pjesën `endl` (nga end of line) të komandës `cout`, e cila gjendet në rreshtin e dytë të kësaj komande, kompjuterit i urdhërohet që pas shtypjes në ekran të fjalisë së dhënë më sipër të kalohet në rreshtin vijues.

Kompajlimi i programit

Për ta kompajluar programin e shkruar, në menynë kryesore duhet të zgjedhet opcioni `Build` dhe pastaj nënopcioni `Compile`, ashtu siç shihet në *Fig.1.15*.

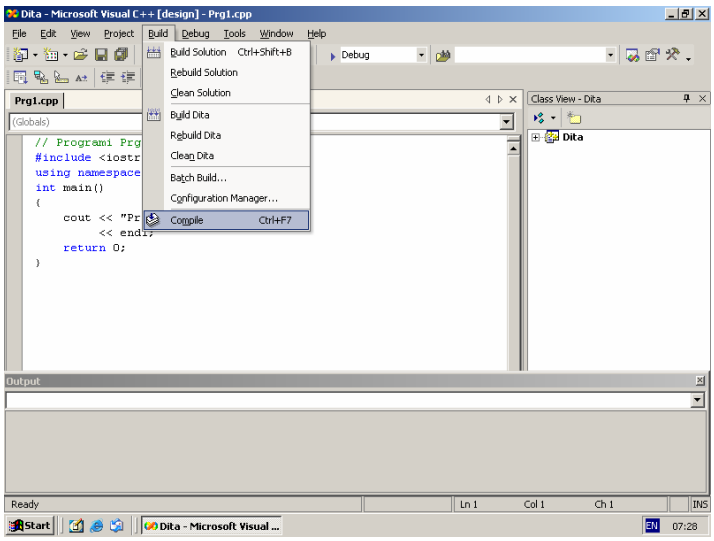


Fig.1.15 Zgjedhja e opcionit Compile për kompajlim të programit

Pas dhënies së komandës për

kompajlim, kompjuteri e përpunon së pari direktivën paraprocesorike, e cila në kodin burimor fillon me simbolin #, si dhe komandat që kanë të bëjnë me këtë direktivë. Pas kësaj, vazhdon kompajlimi i programit, duke i shfrytëzuar edhe rezultatet që fitohen nga paraprocesimi.

Rezultati i procesit të kompajlimit shihet në dritaren *Output*, e cila gjendet në fund të ekranit. Nëse kompajlimi është pa asnjë gabim, në këtë dritare do të paraqitet mesazhi:

```
Build: 1 succeeded, 0 failed, 0 skipped
```

ashtu siç shihet në *Fig.1.16*.

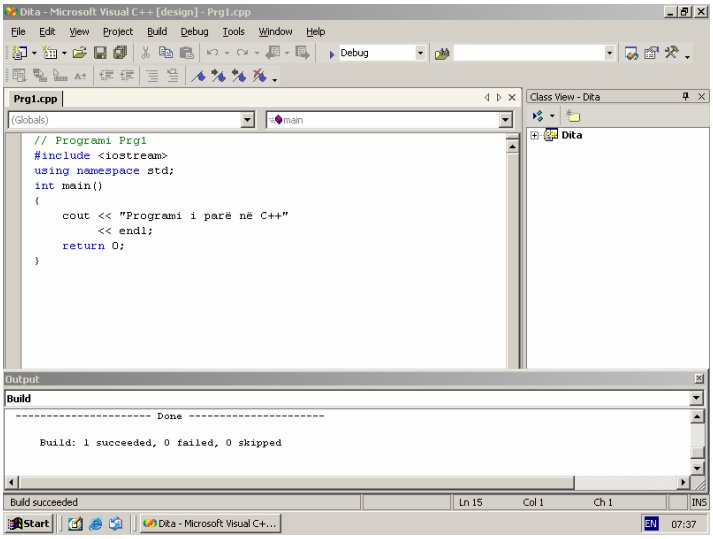


Fig.1.16
Rezultati i kompajlimit

Procesi i kompajlimit të një programi mund të aktivizohet edhe përmes tastierës, duke e shtypur kombinimin e tasteve `Ctrl+F7`.

Krijimi i programit ekzekutiv

Nëse në menynë rënëse të opcionit `Build` (shih *Fig.1.15*) zgjedhet njëri nga nënopcionet `Build` ose `Rebuild`, nga programi burimor do të fitohet programi ekzekutiv. Gjatë kësaj, kompjuteri fillimisht e kompajlon programin, plotësisht njëlloj si edhe me zgjedhjen e opcionit `Compile` dhe pastaj, nëse nuk ka gabime, vazhdon me procesin e linkimit të tij. Nga kjo shihet se zgjedhja e opcionit `Compile` nuk është e domosdoshme.

Meqë këtu kemi të bëjmë me projektin i cili përmban vetëm një fajll (një program), nuk ka rëndësi se cili nga 4 nënopcionet e menysë rënëse të opcionit `Build` zgjedhet (qofshin opcionet `Build` ose ato `Rebuild`).

Në pjesën vijuese të këtij teksti, për krijimin e fajllit ekzekutiv, në menynë rënëse të opcionit `Build` do të shfrytëzohet nënopcioni `Build Solution`, direkt, ose përmes tastierës, me kombinimin e tasteve `Ctrl+Shift+B`.

Ekzekutimi i programit

Për ta fituar rezultatin e një programi, fajlli ekezutiv i tij duhet të ekzekutohet duke e zgjedhur nënopcionin `Start`, ose `Start Without Debugging`, ashtu siç shihet në *Fig.1.17*.

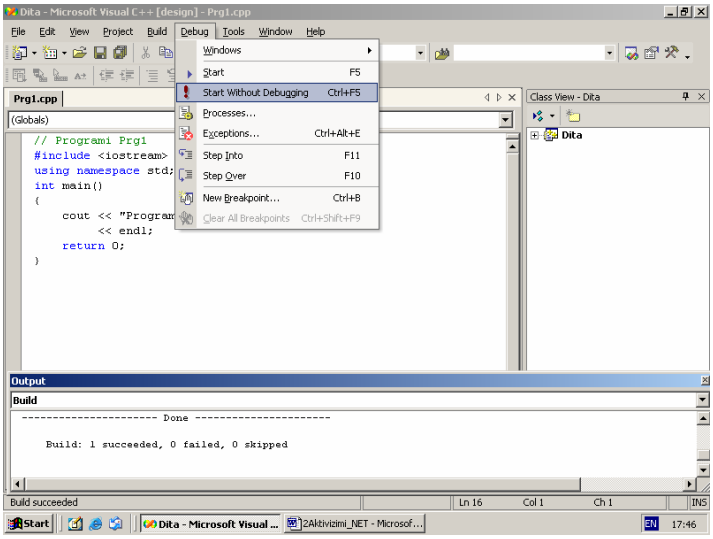


Fig.1.17
Nënopcionet për ekzekutim të programit

Komandat për ekzekutim të fajllit ekzekutiv të programit, siç është ai i cili ruhet në fajllin e krijuar `Prg1.exe`, mund të jepen edhe përmes tastierës, duke e shtypur tastin `F5`, ose kombinimin e tasteve `Ctrl+F5`.

Pavarësisht se cili nga opcionet e përmendur më sipër zgjedhet për ekzekutim të programit, në ekran do të paraqitet *dritarja e rezultateve*, ashtu siç shihet në *Fig.1.18*.

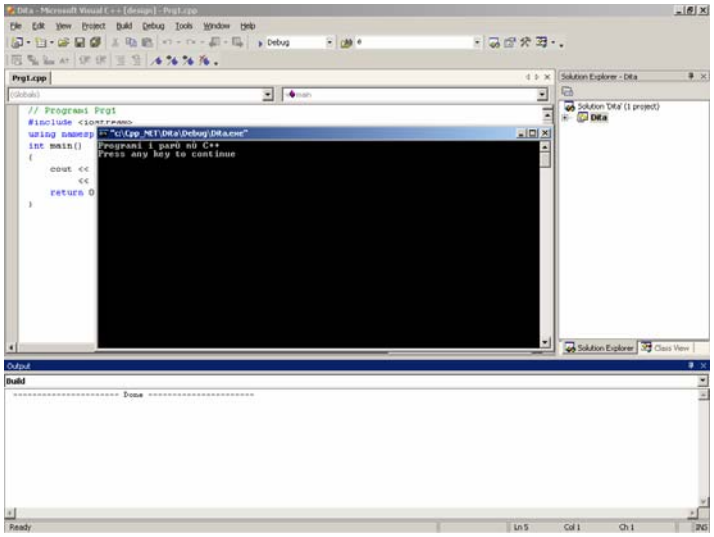


Fig.1.18
Dritarja e rezultateve

Në *dritaren e rezultateve* të dhënë në *Fig.1.18*, si rezultat është shkruar fjalia:

```
Programi i parù nù C++
```

që shënohet nën thonjëza në vazhdim të komandës për shtypje `cout`. Siç shihet edhe në figurën e dhënë, në vend të shkronjës shqipe ë është gjeneruar një simbol tjetër. Për ta eliminuar këtë problem, në program gjatë gjenerimit të shkronjës ë, duhet të shfrytëzohet shkurtesa `Alt+0137` në vend të asaj `Alt+137`. Si rezultat, në komandën për shtypje të tekstit në fjalë, në vend të shkronjës ë do të paraqitet simboli %, përkatësisht komanda do të duket kështu:

```
cout << "Programi i par% n% C++"
     << endl;
```

Nëse pas kësaj ekzekutohet programi në fjalë, në ekran do të shkruhet fjalia:

```
Programi i parë në C++
```

Njëkohësisht, menjëherë nën këtë fjali, paraqitet edhe mesazhi:

```
Press any key to continue
```

me ç'rast kompjuteri njofton se për ta eliminuar *ekranin e rezultateve*, përkatësisht për të kaluar në hapësirën punuese të programit, në tastierë duhet të shtypet një tast.

Gabimet në kompajlim

Nëse gjatë shkruarjes së programit nuk u jemi përmbajtur rregullave për shkruarje të komandave të veçanta, gjatë kompajlimit kompjuteri do t'i lajmërojë gabimet përmes mesazheve në dritaren në fund të ekranit. Kështu, p.sh., nëse në programin `Prg1`, komanda për shtypje në ekran cout shkruhet pa pikëpresje në fund të rreshtit të dytë të saj, në këtë mënyrë:

```
cout << "Programi i parë në C++"
     << endl
```

gjatë kompajlimit të programit, në dritaren *Output*, e cila gjendet në fund të ekranit, kompjuteri do ta shtypë mesazhin:

```
Build: 0 succeeded, 1 failed, 0 skipped
```

për të na njoftuar se e ka detektuar `1` gabim. Njëkohësisht, në ekran paraqitet edhe dritarja në të cilën tregohet vendi dhe lloji i gabimit, ashtu siç është treguar në *Fig.1.19*.

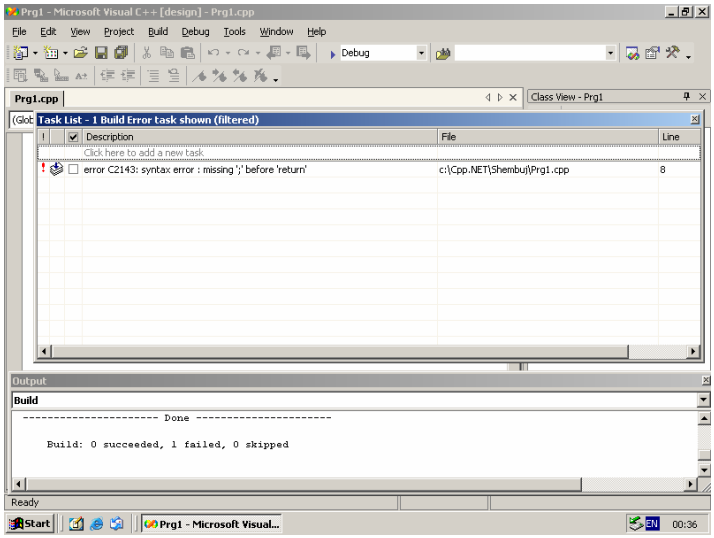


Fig.1.19
Dritarja me informata për vendin dhe llojin e gabimit

Për ta gjetur rreshtin me gabim, duhet të shfrytëzohet numri i rreshtit, të cilin kompjuteri e jep në kolonën e fundit të mesazhit për gabim (kolona mbi të cilin është shënuar titulli `Line`). Kështu, në shembullin e gabimit të mësipërm kompjuteri tregon se gabimi është në rreshtin e `8`, bën pjesë në grupin e gabimeve sintaktike dhe në bazë të pjesës së mesazhit:

```
missing ';' before 'return'
```

detektohet plotësisht ky gabim.

Kur në program ka më shumë gabime, shpesh ngjet që gabime të caktuara të jenë si rezultat i gabimeve të tjera, ashtu që, pas eliminimit të një pjese të tyre, gabimet që ndërlidhen me to eliminohen automatikisht.

Shtimi i numrave rendorë para rreshtave

Me qëllim të lehtësimit të gjetjes së rreshtave në të cilët kompjuteri gjatë kompajlimit ka zbuluar gabime, rreshtave mund t'u shtohen automatikisht edhe numrat rendorë. Për këtë qëllim, te menya rënëse e opcionit `Tools` në menynë kryesore duhet të zgjedhet nënopcioni `Options`.... Si rezultat në ekran hapet dritarja *Options*, e cila shihet në *Fig.1.20*.

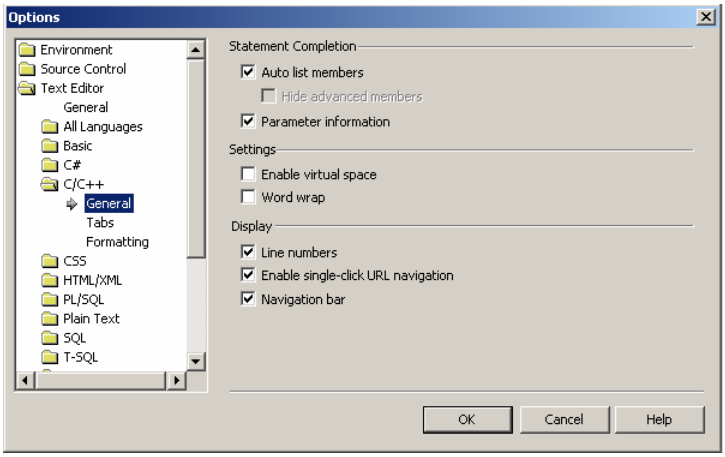


Fig.1.20 Dritarja Options me opcionin Line numbers të zgjedhur

Në pjesën e majtë të dritares në fjalë duhet të zgjedhet fillimisht opcioni `Text Editor` dhe pastaj te nënopconi `C/C++` i saj, edhe opcionin `General`. Në fund, te pjesa e djathtë e dritares (shih *Fig.1.20*), te katrori para opcionit `Line numbers`, me klikim duhet të vendoset simboli √ për selektim. Pas kësaj, me klikimin e pullës `OK`, në ekran do ta kemi pamjen e dhënë në *Fig.1.21*, ku para çdo rreshti të programit shihet numri rendor përkatës.

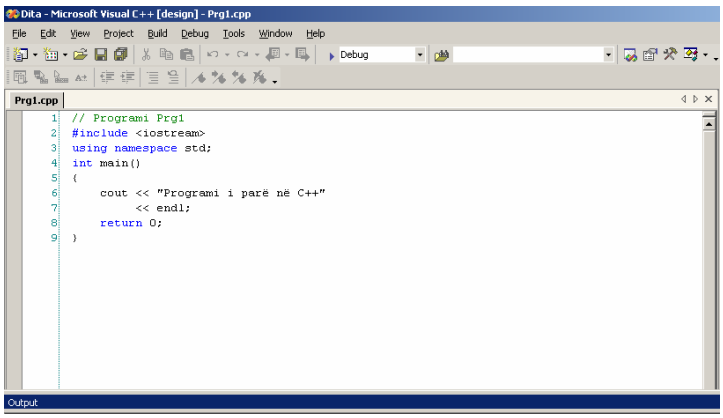


Fig.1.21.
Pamja e programit pas shtimit të numrave rendorë të rreshtave

Ruajtja e programit në disk

Siç pamë edhe më parë, gjatë hapjes së hapësirës punuese për shkruarje të programit, e përcaktojmë edhe emrin e programit. Kështu, për shembullin e shkruar më sipër e zgjodhëm emrin `Prg1.cpp`, ku, nëse nuk e shënojmë prapashtesën, kompjuteri fajllit ia shton prapashtesën `cpp` (meqë kemi të bëjmë me program të shkruar në `C++`). Njëkohësisht, programi ruhet në disk, në folderin aktual ose në folderin që e kemi zgjedhur për vendosje të projektit (këtu u shfrytëzua folderi `C:\Cpp.NET`).

Pas shkruarjes së një pjese të programit, ose edhe të komplet programit, ai mund të ruhet në disk duke shkuar në opcionin `File` të menysë kryesore dhe pastaj duke zgjedhur në menynë rënëse përkatëse nënopcionin `Save Prg1.cpp`, ashtu siç shihet në *Fig.1.22.*

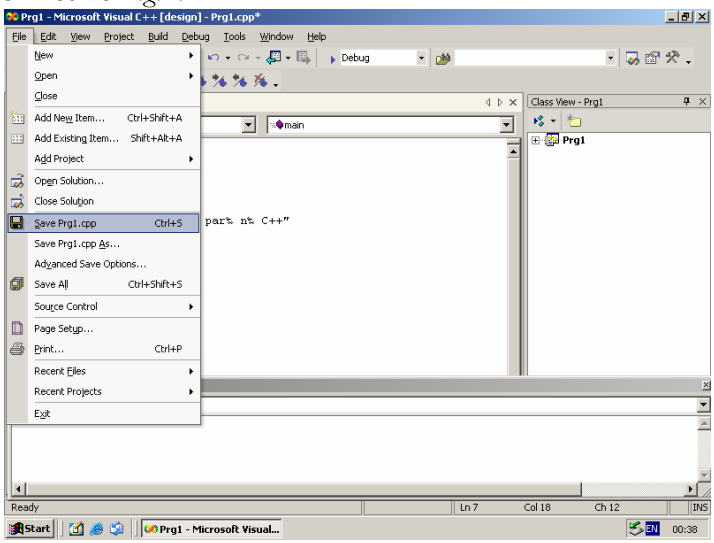


Fig.1.22
Ruajtja e programit në disk

Për zgjedhje të opcionit `Save` mund të shfrytëzohet edhe kombinimi i tasteve në tastierë `Ctrl+S`, ose pulla që shihet majtas, e cila është vendosur në shiritin me vegla që gjendet nën menynë kryesore.

Nëse programin e shkruar duam ta ruajmë me ndonjë emër tjetër, në menynë rënëse të opcionit `File` të menysë kryesore e zgjedhim opcionin `Save Prg1.cpp As...`.

Por, kur njëkohësisht kemi punuar në shkruarjen ose ndryshimin e disa programeve, për ruajtjen e tyre në disk, në menynë rënëse të opcionit `File` të menysë kryesore mund të shfrytëzohet nënopcioni `Save All`. Efekt të njëjtë në ruajtjen e më shumë programeve në disk ka edhe shfrytëzimi i pullës, që shihet majtas, e cila gjendet në menynë kryesore, ose edhe i shkurtesës `Ctrl+Shift+S`.

Rihapja e projektit

Nëse e kemi ndërprerë punën në një projekt, përkatësisht në programet e përfshirë në projekt, p.sh., gjatë ndërprerjes së punës së kompjuterit, rihapja e programeve fillon me rihapjen e projektit. Për këtë qëllim, pasi të jetë aktivizuar gjuha programuese C++, te menya rënëse e opcionit `File` të menysë kryesore, zgjedhen nënopcionet `Open` dhe `Project...`, ashtu siç shihet në *Fig.1.23*.

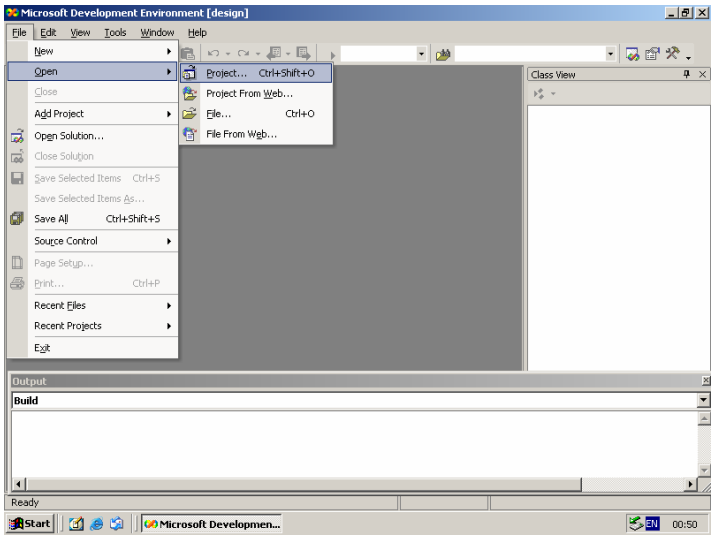


Fig.1.23
Zgjedhja e nënopcioneve për hapje të një projekti ekzistues

Pas kësaj, duhet ta zgjedhim folderin në të cilin është ruajtur projekti që duam ta hapim. Nëse, p.sh., është zgjedhur folderi `Cpp.NET`, në ekran do të shihet përmbajtja e tij, ashtu siç është dhënë në *Fig.1.24*.

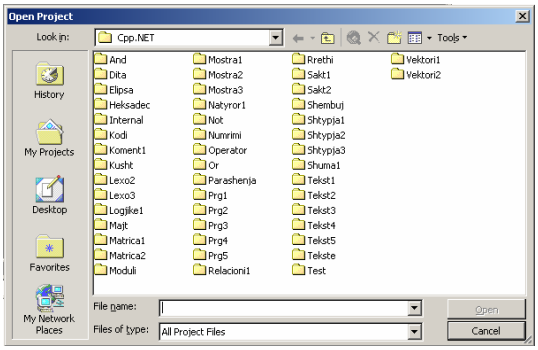


Fig.1.24
Përmbajtja e folderit Cpp.NET

Për ta hapur projektin që duam, folderi përkatës zgjedhet me mi dhe pastaj klikohet pulla `Open`, ose në tastierë shtypet tasti `Enter`, për ta fituar në fund programin përkatës në ekran, p.sh., ashtu siç shihet në *Fig.1.14* programi `Prg1`.

Projekti në të cilin jemi duke punuar mund të hapet edhe direkt, duke e zgjedhur nënopcionin `Recent Projects` në menynë rënëse të opcionit `File`. Pas kësaj, në ekran do të paraqitet lista e projekteve në të cilat është punuar së fundi, p.sh., ashtu siç shihet në *Fig.1.25*.

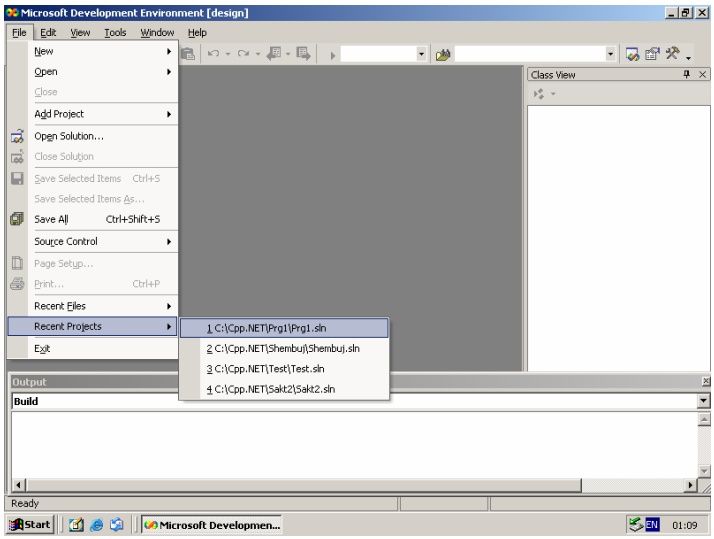


Fig.1.25
Lista e projekteve në të cilët është punuar së fundi

Rihapja e programit

Nëse është i hapur vetëm projekti, për ta hapur fajllin e programit të mbyllur, në menynë rënëse të opcionit `File` të menys kryesore zgjedhet nënopcioni `Open` dhe pastaj edhe nënopcioni `File...`, ashtu siç shihet në *Fig.1.26*.

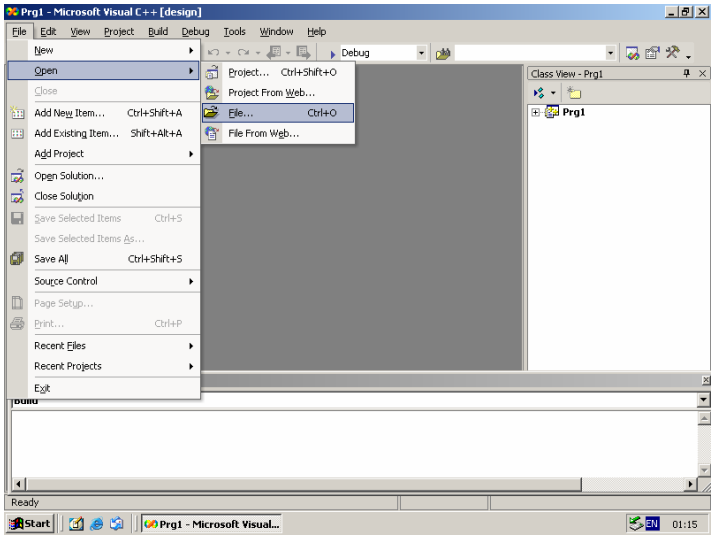


Fig.1.26
Hapja e programit të shkruar më parë

Pas kësaj, në ekran hapet dritarja me folderin aktiv dhe fajllat që përfshihen brenda tij. Nëse, p.sh., aktiv është folderi i projektit `Prg1`, përmbajtja e tij do të duket si në *Fig.1.27*.

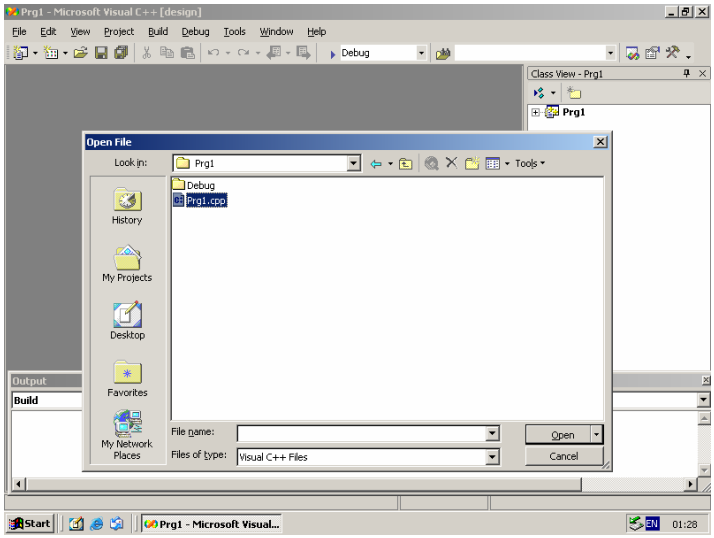


Fig.1.27
Përmbajtja e folderit Prg1

Në fund, duke e klikuar pullën `Open`, ose duke e shtypur në tastierë tastin `Enter`, do të hapet programi `Prg1.cpp`, ashtu siç shihet në *Fig.1.14*.

Edhe gjatë hapjes së fajllave mund të shfrytëzohet nënopcioni `Recent Files` i opcionit `File`, plotësisht njëlloj si edhe gjatë hapjes së projekteve ekzistuese.

Ekzekutimi i programeve përmes një projekti

Hapja e më shumë programeve, të cilat nuk bëjnë pjesë në një projekt të përbashkët, p.sh., të programeve të cilat i shkruajmë me qëllim të mësimit të gjuhës C++, nuk lejohet, sepse, në një projekt, funksioni `main` mund të paraqitet vetëm një herë. Prandaj, për çdo program që shkruhet duhet patjetër të hapet projekti përkatës.

Por, ekziston mundësia që fajllat burimorë të më shumë programeve të ruhen në një folder dhe të ekzekutohen sipas dëshirës, duke e shfrytëzuar vetëm një projekt të hapur, gjë që është shpjeguar në vijim.

Ruajtja e fajllave burimorë në një folder

Nëse duam që të gjithë fajllat e programeve burimorë, të krijuar më parë, t'i ruajmë në folderin përkatës të një projekti të përbashkët, shkojmë dhe i hapim ata një nga një. Pastaj, fajllat e programeve burimore i ruajmë në folderin e projektit të përbashkët, duke e shfrytëzuar opcionin `Save emri.cpp As...`, të menysë rënëse të opcionit `File`, ku me `emri` duhet nënkuptuar emrin e fajllit që ruhet.

Nëse, p.sh., e marrim si të përbashkët projektin `Shembuj`, pasi të hapet si projekt i ri, në disk do të krijohet folderi përkatës, si edhe për çdo projekt tjetër të hapur. Për ta vendosur në këtë folder programin `Prg1.cpp` të shkruar më parë, duhet të hapet projekti përkatës, pasi më parë të jetë mbyllur projekti `Shembuj`. Në menynë rënëse të opcionit `File`, e zgjedhim nënopcionin e përmendur më sipër, ashtu siç shihet në *Fig.1.28*.

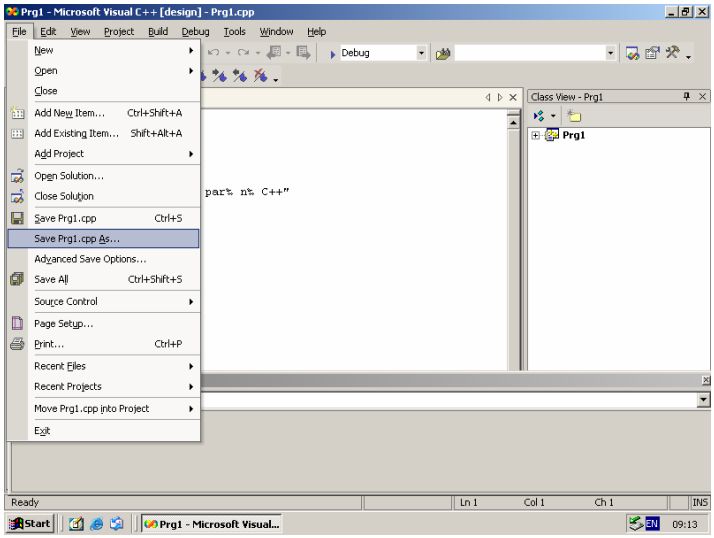


Fig.1.28
Nënopcioni Save As...

Pas kësaj, në vend të folderit `Prg`, i cili na ofrohet nga kompjuteri (shih *Fig.1.29*), duke e klikuar shigjetën në fund të dritares përkatëse, nga pema e folderëve, duhet ta zgjedhim folderin `Shembuj`.

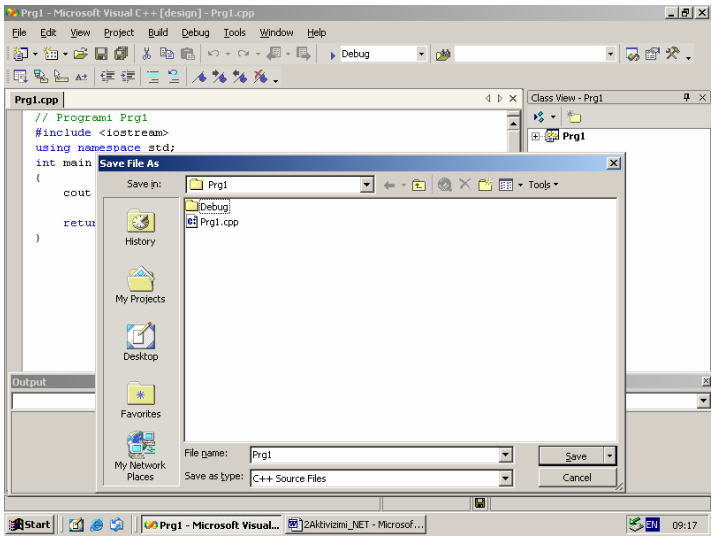


Fig.1.29
Folderi Prg1 i cili ofrohet nga kompjuteri

Në fund, nëse zgjedhet folderi `Shembuj`, që të përfundojë procesi i ruajtjes së programit `Prg1.cpp` në folderin e projektit të përbashkët, duhet të klikohet pulla `Save`, që është vendosur në këndin e djathtë-poshtë të dritares.

Ekzekutimi i programeve nga projekti i përbashkët

Procedura e ekzekutimit të programeve që ruhen në projektin e përbashkët fillon me hapjen e projektit të përbashkët, ashtu siç është shpjeguar më parë për hapje të një projekti. Nëse ky është projekti `Shembuj`, që e përmendëm më sipër, p.sh., si rezultat mund ta kemi pamjen e dhënë në *Fig.1.30*.

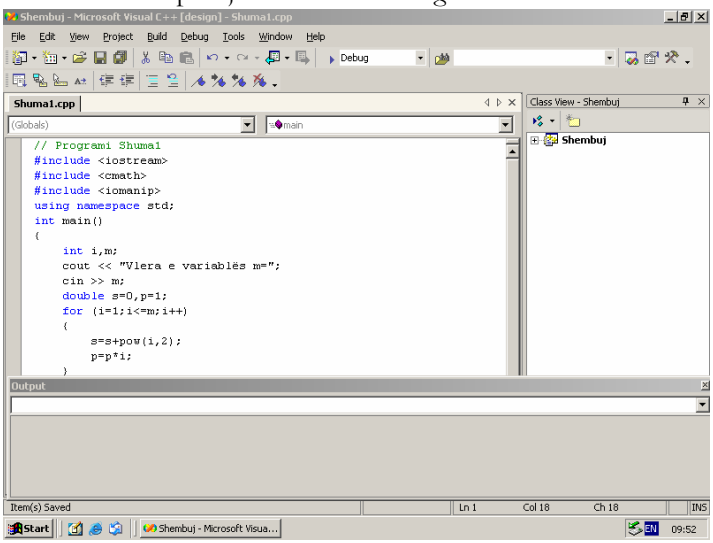


Fig.1.30
Hapja e projektit Shembuj

Programi `Shuma1`, i cili shihet në hapësirën punuese të projektit të përbashkët `Shembuj`, në fakt është programi i cili është shfrytëzuar gjatë hapjes së fundit të tij. Pas kësaj, në menynë rënëse të opcionit `File` e zgjedhim nënopcionin `Add Existing Item...`, ashtu siç shihet në *Fig.1.31*.

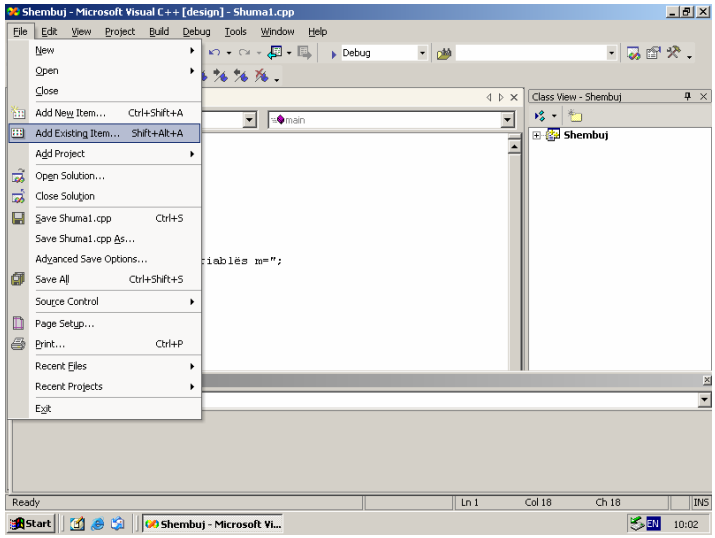


Fig.1.31
Nënopcioni Add Existing Item...

Me zgjedhjen e nënopcionit në fjalë, p.sh., si rezultat do të hapet dritarja me listën e fajllave burimorë të programeve që janë vendosur në folderin `Shembuj`, ashtu siç është treguar në *Fig.1.32*.

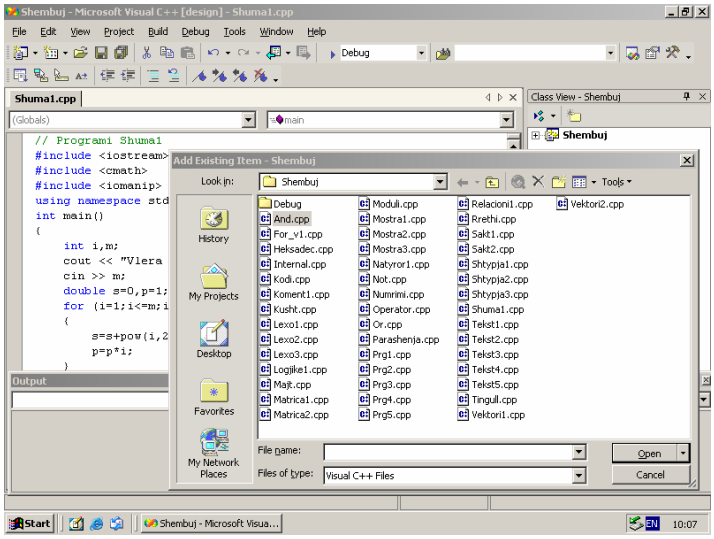


Fig.1.32
Lista e programeve të vendosur në folderin Shembuj

Nëse nga lista e ofruar, p.sh., e zgjedhim programin `Prg1.cpp`, që ta shohim shtimin e tij në përbërje të projektit `Shembuj`, duhet zgjedhur nënopcionin `Solution Explorer` të opcionit `View` në menynë kryesore, ashtu siç shihet në *Fig.1.33*.

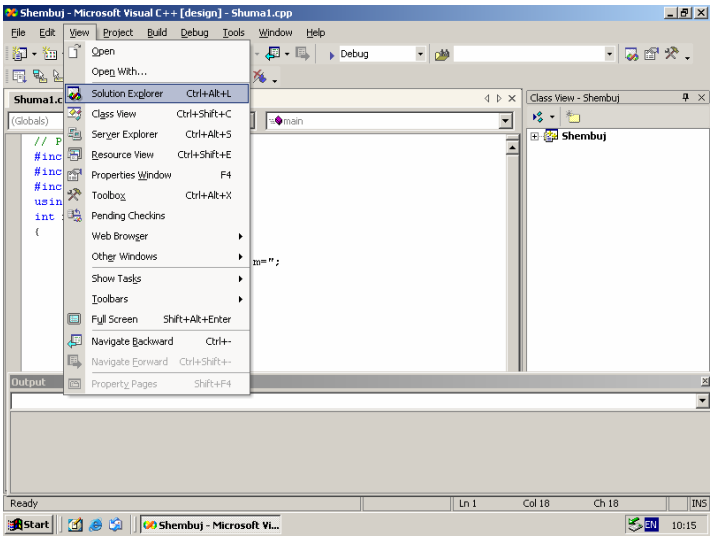


Fig.1.33
Nënopcioni Solution Explorer

Si rezultat, në ekran paraqitet dritarja me listën e programeve burimore që janë përfshirë në kuadër të projektit `Shembuj`, ashtu siç shihet në *Fig.1.34*.

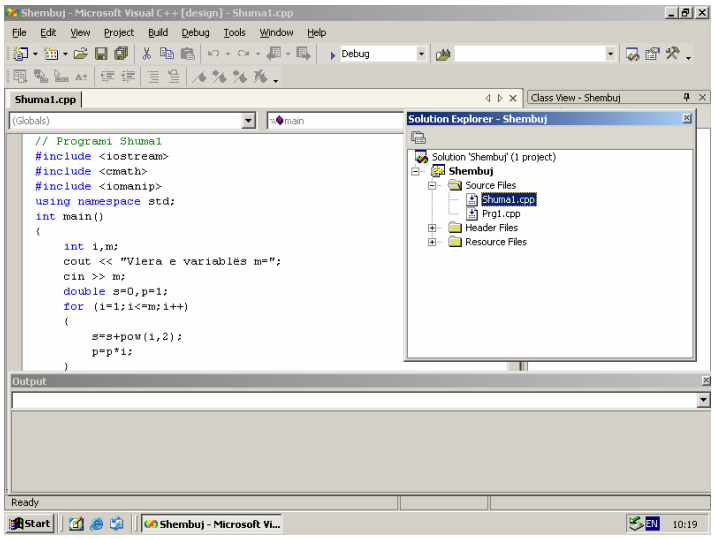


Fig.1.34
Lista e programeve burimorë në kuadër të projektit Shembuj

Këtu, fillimisht, duhet të fshihet fajlli `Shuma1.cpp`. Meqë automatikisht, pas hapjes së dritares në fjalë, ky fajll është i selektuar, për fshirje të tij në tastierë duhet të shtypet tasti `Delete`. Pas kësaj, nëse klikohet dy herë fajlli i mbetur `Prg1.cpp`, ai do të hapet dhe do të vendoset në hapësirën përkatëse në ekran. Në fund, duke e shfrytëzuar procedurën e zakonshme, fajlli i programit `Prg1.cpp` mund të kompajlohet dhe të ekzekutohet.

Përfundimi i punës në C++

Para se të ndërpritet puna në kompjuter, duhet të mbyllen të gjithë programet, përfshirë edhe *Visual Studion*, në përbërje të të cilit gjendet edhe gjuha programuese `C++`. Për këtë qëllim, fillimisht duhet të mbyllet programi (ose programet, nëse janë të hapur më shumë programe) dhe pastaj edhe projekti përkatës.

Për mbyllje të programit shfrytëzohet opcioni `File` i menysë kryesore dhe pastaj në menynë rënëse përkatëse zgjedhet nënopcioni `Close`, pas së cilit mbyllet programi. Pastaj, në të njëjtën meny rënëse zgjedhet nënopcioni `Close Solution`, për ta mbyllur edhe projektin. Por, nëse para mbylljes së programit dhe projektit, në menynë rënëse të opcionit `File` zgjedhet nënopcioni `Close Solution`, do të mbyllet projekti dhe fajlli i hapur i programit që përfshihet brenda tij.

Në fund, pas mbylljes së projektit mund të ndërpritet komplet puna e *Visual Studios*, duke shkuar në nënopcionin `Exit` të menysë rënëse të opcionit `File` në menynë kryesore.

2

Elementet e gjuhës C++

Shkruarja e programeve në njërën nga gjuhët programuese mbështetet në rregulla të caktuara. Para se të jepen njohuri të përgjithshme mbi rregullat e shkruarjes së programeve në gjuhën programuese C++, duhet të përcaktohen elementet e kësaj gjuhe, siç janë: të dhënat, identifikatorët, konstantet, variablat, pointerët, operatorët dhe shprehjet aritmetikore.

Në pjesën vijuese të librit, me qëllim të qartësimit më të mirë të materialit që shpjegohet, jepen shembuj të programeve komplete, pavarsisht se brenda tyre paraqiten komanda të cilat do të mësohen më vonë.

Të dhënat standarde

Informatat të cilat i jepen kompjuterit në një formë të kuptueshme për të, quhen *të dhëna* (ang. data). Varësisht nga natyra dhe kufijt e vlerave të tyre, në gjuhën programuese C++ shfrytëzohen tipe të ndryshme të të dhënave. Në kolonën e parë të tabelës e cila shihet në *Fig.2.1* janë numëruar të dhënat standarde që shfrytëzohen në gjuhën C++.

Tipi	Rangu i vlerave	Bajtë
short short int	-32768..32767	2
int	-2147483648..2147483647	4
long long int	-2147483648..2147483647	4
unsigned short unsignet short int	0..65535	2
unsigned unsigned int	0..4294967295	4
unsigned long unsigned long int	0..4294967295	4
float	1.2e-38..3.4e+38	4
double	1.7e-308..1.7e+308	8
long double	3.4e-4932..1.1e+4932	10
char	-128..127	1
unsigned char	0..255	1
signed char	-128..127	1
wchar_t	0..65535	2
bool	true, false	1

Fig.2.1 Tipe të ndryshme të të dhënave

Për disa nga tipet e të dhënave, gjatë deklarimeve shfrytëzohen edhe sinonime, ashtu siç shihet në kolonën e parë të tabelës në fjalë. Në kolonën e dytë të saj shihen diapazonet e vlerave të mundshme për tipet e veçanta të të dhënave, kurse në kolonën e tretë të tabelës janë dhënë madhësitë e lokacioneve memoruese të shprehura në bajtë, ku vendosen vlerat përkatëse. Kështu, p.sh., për vendosje në memorie të të dhënave të tipit `float`, ku përfshihen vlerat numerike me pikë dhjetore, shfrytëzohen 4 bajtë, kurse vlerat e tyre numerike mund të lëvizin mes vlerës `1.2e-38` dhe `3.4e+38`, përkatësisht mes vlerave 1.2×10^{-38} dhe $3.4 \times 10^{+38}$.

Numrat e plotë

Tipet e të dhënave, te të cilat paraqitet shkurtesa `int`, u përkasin *numrave të plotë*, përkatësisht numrave për të cilët thuhet se janë *numra intexherë* (ang. integer). Të dhënat, tipet e të cilave fillojnë me fjalën *unsigned*, paraqesin numra të plotë *pa parashenjë*. Kështu, p.sh., në grupin e të dhënave `unsigned short` bëjnë pjesë numrat e plotë:

```
35
2458
54397
```

Në këtë grup të dhënash nuk bëjnë pjesë, p.sh., numrat:

```
74895
-27
```

i pari, sepse është më i madh se numri `65535`, kurse i dyti, sepse ka parashenjë.

Si të dhëna të tipit `short`, p.sh., llogariten numrat:

```
3479
-22581
```

por jo edhe numrat:

```
34827
-47915
```

sepse janë jashtë diapazonit përkatës, i cili është dhënë në tabelën e *Fig.2.1*.

Numrat dhjetorë

Të dhënat e tipeve `float` dhe `double` u përkasin numrave dhjetorë (*numrave jo të plotë*), ose, siç thuhet ndryshe, *numrave realë* (ang. real). Këta numra mund të shkruhen si *numra me pikë fikse* dhe si *numra me pikë të lëvizshme*. Numra realë me pikë fikse janë numrat të cilët përmbajnë pikë decimale. Të tillë janë, p.sh., numrat:

```
35.62
-8549.528
```

Kurse, numrat me pikë të lëvizshme janë numrat realë të cilët shkruhen në formë eksponenciale. Kështu, p.sh., numrat të cilët në matematikë shkruhen:

$3.85 \cdot 10^{5}$
$-74.3 \cdot 10^{12}$
$6345.3961 \cdot 10^{-8}$

në gjuhën `C++` shkruhen si numra me pikë të lëvizshme, në këtë mënyrë:

```
3.85e5
-74.3e12
6345.3961e-8
```

Këtu, eksponentëve me vlerë pozitive mund t'u shtohet edhe parashenja, përkatësisht ata mund të shkruhen edhe në këtë formë:

```
3.85e+5
-74.3e+12
```

Të dhënat karakter

Të dhënat tekstuale, të cilat përmbajnë shkronja, shifra numerike ose simbole të tjera, në kompjuter ruhen si *të dhëna të tipit karakter*, duke u deklaruar si të dhëna të tipit `char`, `unsigned char` dhe `signed char`. Për të dhënat e tilla në gjuhën `C++` shfrytëzohet 1 bajt dhe në të ruhet kodi përkatës i karakterit në bazë të kodit ASCII (nga American Standard Code for Information Interchange).

Shembull

Programi përmes së cilit në ekran gjenerohen kodet e karaktereve të cilat shfrytëzohen në kodin `ASCII`.

```
// Programi Kodi
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int i;
     char a;
     for (i=33;i<=255;i++)
     {
          a=i;
          cout << setw(2)
               << a
               << setw(6)
               << i;
     }
     cout << endl;
return 0;
}
```

Rezultati që fitohet në ekran pas ekzekutimit të programit `Kodi` do të duket si në *Fig.2.2.*

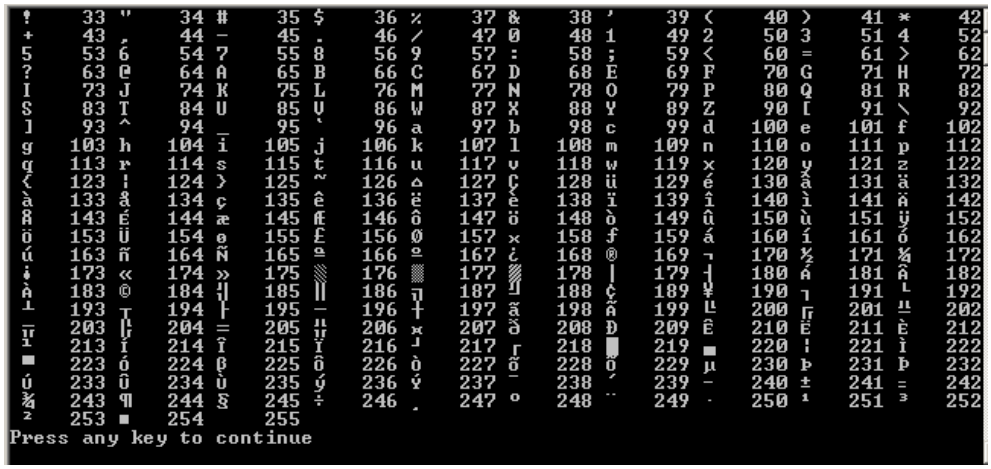


Fig.2.2 Kodi ASCII i cili fitohet pas ekzekutimit të programit Kodi

Nga rezultati i dhënë shihet se, p.sh., shkronjës `D` si kod i është ndarë ekuivalenti i numrit decimal `68`, kurse kodi i shkronjës së gjuhës shqipe `ë` është ekuivalenti i numrit decimal `137`. Me program nuk është përfshirë shtypja e karaktereve me kode mes vlerave decimale `0` dhe `32`, sepse me shtypjen e tyre jepen komanda të ndryshme. Kështu, p.sh., me shtypjen e kombinimit të karaktereve `\n`, të cilit kombinim i përket kodi me vlerë decimale `10`,

kompjuterit i urdhërohet që shtypjen ta vazhdojë në rresht të ri. Duke e shfrytëzuar këtë kombinim karakteresh, komanda cout e shfrytëzuar te programi `Kodi`:

```
cout << endl;
```

mund të shkruhet edhe në këtë mënyrë:

```
cout << "\n";
```

Kombinimet dykarakterëshe, te të cilat si karakter i parë paraqitet vija e pjerrët mbrapsht \, paraqesin *sekuenca dalëse* (ang. escape sequences).

Ngjashëm shfrytëzohen edhe disa sekuenca dalëse të tjera, te të cilat si simbol i parë paraqitet simboli `\` (shih tabelën në fund të librit - **shtesa B**).

Të dhënat e tipit `char`, përkatësisht *karakter*, kompjuterit i jepen duke i shkruar nën thonjëza të njëfishta, p.sh., kështu:

```
'a'
'*'
'5'
```

Edhe zbrazësira llogaritet si e dhënë e tipit karakter dhe ajo njëlloj mund të shënohet nën thonjëza të njëfishta `' '`.

Tipi `wchar_t` (nga wide character type), i dhënë në rreshtin e parafundit të tabelës në *Fig.2.1*, mund të shfrytëzohet për karakteret e kodit 16-bitësh `Unicode`, i cili aktualisht përmban kode për `24` gjuhë me rreth `39000` simbole të ndryshme, prej të cilëve rreth 21000 shfrytëzohen për ideogramet kineze (ky kod gjithsej mund të përmbajë 65536 simbole).

Vlerat kufitare

Variablat e tipeve të ndryshme, të cilat shfrytëzohen në gjuhën programuese C++, i kanë vlerat e tyre kufitare. Këto vlera kufitare janë dhënë në tabelën e cila shihet në Fig.2.1. Por, këto vlera kufitare ruhen edhe si konstante të caktuara, të cilat sipas nevojës mund të shfrytëzohen, p.sh., me qëllim të kontrollimit të mostejkalimit të vlerave përkatëse.

Shembull

Programi përmes së cilit në ekran shtypen vlerat kufitare për variablat që janë deklaruar si variabla të tipit `int`, `char` dhe `unsigned int`.

```
// Programi Prg2
#include <iostream>
#include <climits>
using namespace std;
int main()
{
     cout << "Vlera minimale int: "
          << INT_MIN
          << endl;
     cout << "Vlera maksimale int: "
          << INT_MAX
          << endl;
     cout << "Vlera minimale char: "
          << CHAR_MIN
          << endl;
     cout << "Vlera maksimale char: "
          << CHAR_MAX
          << endl;
     cout << "Vlera maksimale unsigned int: "
          << UINT_MAX
          << endl;
return 0;
}
```

Siç shihet në program, vlerat në fjalë fitohen duke i urdhëruar kompjuterit që t'i shtypë vlerat e konstanteve përkatëse të cilat ruhen në fajllin `climits`, i cili vendoset në ballinën e programit, përmes komandës paraprocesorike `#include <climits>`. Vlerat e konstanteve, të cilat do të shtypen në ekranë, janë vlerat e tipeve përkatëse të dhëna në tabelën e *Fig.2.1*, kurse rezultati në ekran do të duket si në *Fig.2.3*.

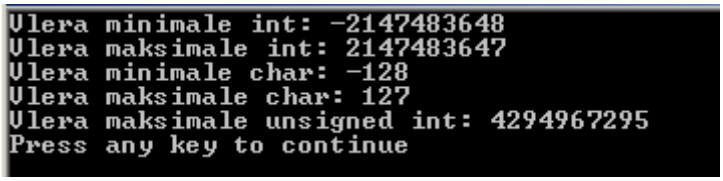


Fig.2.3 Rezultati i programit Prg2

Hapësira memoruese që shfrytëzohet

Për ruajtje të të dhënave të tipit të caktuar, kompjuteri shfrytëzon një hapësirë memoruese të caktuar, ashtu siç shihet në kolonën e fundit të tabelës së dhënë në *Fig.2.1*. Vlerat e këtyre hapësirave memoruese mund të merren edhe me program, duke e shfrytëzuar operatorin `sizeof`.

Shembull Programi përmes së cilit në ekran shtypet madhësia e hapësirës memoruese në bajtë, të cilën kompjuteri e shfrytëzon për vendosjen e të dhënave të tipit `int` dhe `double`.

```
// Programi Prg3
#include <iostream>
using namespace std;
int main()
{
     cout << "\nHapësira për int: "
          << sizeof(int)
          << " bajtë"
          << endl;
cout << "\nHapësira për double: "
          << sizeof(double)
          << " bajtë"
          << "\n\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.2.4*.

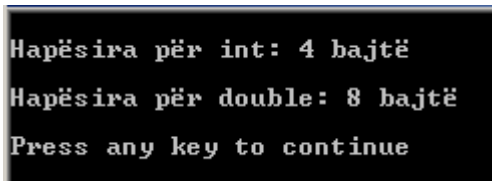


Fig.2.4
Rezultati i programit Prg3

Stringjet

Vargjet e simboleve të ndryshëm (shkronjave, numrave dhe simboleve speciale), përkatësisht tekstet e çfarëdoshme të shkruara brenda thonjëzave, quhen *stringje* dhe llogariten si të dhëna të tipit *string* (ang. string). Kështu, p.sh., të dhëna të tipit string janë:

```
"Gjuha programuese C++"
"Jeta"
"y=3*x+4"
```

Meqë stringjet formohen si vargje të karaktereve, për këto të dhëna thuhet edhe se paraqesin *stringje të karaktereve* (ang. character string), sepse në memorien

e kompjuterit ruhen si vargje karakteresh, të cilave kompjuteri në fund ua shton edhe karakterin zero (ang. null character) `'\0'` për ta përcaktuar fundin e tyre. Kështu, p.sh., stringu `"Jeta"`, në memorien e kompjuterit vendoset si varg karakteresh, në këtë mënyrë:

J	e	t	a	\0

duke shfrytëzuar një bajt për çdo karakter. Në fakt, vlerat e fushave përkatëse në memorie mbushen me vlerat binare të ekuivalentëve decimale të kodeve përkatëse në kodin `ASCII`, ku për karakterin zero në fund të stringut ruhet ekuivalenti binar i numrit `0`.

Shembull Programi përmes së cilit në ekran shtypen kodet e shkronjave të stringut `"Jeta"`, ashtu siç ruhen në memorie në kodin `ASCII`.

```
// Programi Prg4
#include <iostream>
using namespace std;
int main()
{
     int i,k;
     char A[5]="Jeta";
     for (i=0;i<5;i++)
     {
          k=A[i];
          cout << A[i]
               << " ... "
               << k
               << endl;
     }
return 0;
}
```

Rezultati i cili fitohet pas ekzekutimit të programit në kompjuter duket si në *Fig.2.5*.

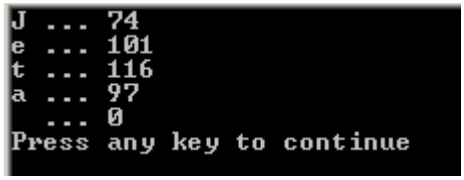


Fig.2.5
Rezultati i programit Prg4

Siç shihet edhe nga *Fig.2.5*, kodi i karakterit zero `\0`, i cili automatikisht vendoset në fund të stringut, është numri decimal `0`.

Duhet pasur kujdes gjatë operimit me stringje, kur si karakter paraqitet shifra decimale `0`, sepse kodi i këtij karakteri në `ASCII` është i barabartë me numrin decimal `48`, kurse njëkohësisht si kod i karakterit zero `\0` shfrytëzohet numri decimal `0`.

Stringu i cili nuk përmban asnjë karakter quhet *string i zbrazët* (ang. empty string) dhe përcaktohet me dy thonjëza të shënuara njëra pas tjetrës, kështu `""`. Në memorien e kompjuterit ky string ruhet në një bajt, sepse edhe stringut të zbrazët kompjuteri ia shton karakterin zero `\0`.

Vlerat logjike

Në gjuhën `C++` përdoren dy vlera logjike, `true` (e vërtetë) dhe `false` (e pavërtetë), të shkruara vetëm me shkronja të vogla. Vlerat logjike kryesisht u ndahen variablave gjatë krahasimeve të ndryshme me qëllim të ruajtjes së informatave për plotësim ose mosplotësim të kushteve. Kompjuteri, në memorien e tij, këto dy vlera i ruan si numra të plotë pozitivë, `0` dhe `1`. Variablat, të cilat shfrytëzohen për ruajtjen e të dhënave logjike, në program deklarohen si variabla të tipit `bool`.

Shembull

Programi përmes së cilit kompjuteri përgjigjet në pyetjen se a është vlera e variablës `x` më e madhe se vlera e variablës `y`. Këto dy vlera kompjuterit i jepen përmes tastierës.

```
// Programi logjike1
#include <iostream>
using namespace std;
int main()
{
     bool z;
     double x,y;
     cout << "Vlera për variablën x: ";
     cin >> x;
     cout << "Vlera për variablën y: ";
     cin >> y;
     cout << "\nA është x>y?: ";
          z=(x>y);
          if (z==true)
             cout << "Po";
          else
             cout << "Jo";
     cout << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe për variablat x dhe y kompjuterit i jepen vlerat hyrëse 3 dhe 7 (përkatësisht), rezultati që fitohet në ekran do të duket si në *Fig.2.6*.

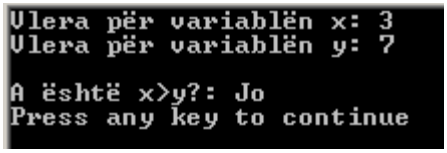


Fig.2.6
Rezultati i programit logjike1

Identifikatorët

Njësitë elementare memoruese në të cilat vendosen të dhënat dhe rezultatet e programeve emërohen duke përdorur *identifikatorë* (ang. identifier). Kështu, identifikatorët përdoren për emrat e konstanteve, variablave, nënprogrameve dhe strukturave të tjera të përfshira në program.

Identifikatorët formohen si kombinim i *shkronjave* (a, b, . . ., z, A, B, . . ., Z), *numrave* (0, 1, 2, . . ., 9) dhe *nënviza* (_). Simboli i parë në identifikatorë mund të jetë shkronjë ose nënvizë. Kështu, p.sh., si identifikatorë mund të merren:

```
dita
Koha5
distanca_mes_rreshtave
TemperaturaDitore
_Fillimi
a234b35
P8
```

Nuk lejohet që simboli i parë në identifikatorë të jetë numër, ose identifikatori të përmbajë simbole speciale (simbole që nuk janë shkronja ose numra), p.sh., siç janë: !, $, %, +, > etj. Kështu, p.sh., gabim do të zgjedhen si identifikatorë kombinimet e simboleve:

`4muaj`	Fillon me numër
`nata+dita`	E përmban simbolin +
`ab jeta`	Përmban zbrazësirë
`US$`	E përmban simbolin $
`Libri#Rend`	E përmban simbolin #
`%fitimi`	E përmban simbolin %

Shkronjat e alfabetit shqip ë, Ë, ç dhe Ç nuk mund të përdoren gjatë formimit të identifikatorëve.

Në gjuhën C++ gjatësia e identifikatorëve nuk është e kufizuar. Por, përdorimi i identifikatorëve të gjatë është jopraktik e ndonjëherë edhe me probleme gjatë kompajlimit.

Fjalët kyçe të cilat përdoren në gjuhën C++ (fjalët e rezervuara nga gjuha C++, shif shtesën A), nuk lejohet të përdoren si identifikatorë. Kështu, p.sh., si identifikatorë nuk mund të përdoren fjalët:

```
else
cout
if
while
```

sepse janë fjalë të rezervuara nga gjuha C++.

Kombinimet që fillojnë me simbolin për nënvizim në gjuhën C++ kanë një përdorim të veçantë, për këtë arsye është e preferueshme që gjatë formimit të identifikatorëve të zakonshëm këto kombinime të mos përdoren. Por, simboli për nënvizim mund të përdoret si lidhës gjatë formimit të identifikatorëve më të gjatë, p.sh., kështu:

```
koha_e_bukur
vlera_fillestare_e_perimetrit
distanca_mes_dy_pikave_te_vecanta
```

Praktikohet edhe kjo formë e formimit të identifikatorëve të dhënë në shembullin e mësipërm:

```
kohaebukur
vlerafillestareeperimetrit
distancamesdypikavetevecanta
```

ose edhe forma:

```
KohaeBukur
VleraFillestareePerimetrit
DistancaMesDyPikaveTeVecanta
```

Kompajleri i gjuhës C++ gjatë formimit të identifikatorëve **i dallon shkronjat e vogla dhe shkronjat e mëdha**. Kjo, d.m.th. se, p.sh., nuk merren si identifikatorë të njëjtë kombinimet:

```
dita
Dita
DITA
ditA
```

Me qëllim të evitimit të gabimeve të mundshme për shkak të dallimit në madhësi të shkronjave, gjatë programimit në gjuhën C++ kryesisht shfrytëzohen shkronjat e vogla.

Variablat

Të dhënat dhe rezultatet që fitohen gjatë llogaritjeve të ndryshme ruhen në hapësirën e memories së kompjuterit si vargje të shifrave binare. Meqë gjatë ekzekutimit të programit në kompjuter, në një lokacion memorues mund të vendosen vlera të ndryshme, përkatësisht vlerat brenda tyre mund të jenë variabile, lokacionet memoruese paraqesin *variabla* (ang. variable). Përmbajtja e një lokacioni njihet edhe si *vlerë e variablës* (ang. variable value), kurse emri simbolik që i shoqërohet e paraqet *identifikatorin e variablës* (ang. variable identifier), ose *emrin e variablës* (ang. variable name).

Vlerat e variablave, varësisht nga tipi i tyre, ruhen në numër të caktuar bajtësh, ashtu siç është treguar në kolonën e fundit të tabelës së dhënë në *Fig.2.1*.

Deklarimi i variablave të zakonshme

Për çdo variabël, para se të shfrytëzohet, duhet të deklarohet tipi i saj. Ky deklarim, p.sh., bëhet kështu:

```
int a;
double x;
float z;
short int koha;
char g;
```

Brenda një deklarimi mund të përfshihen edhe më shumë variabla, duke i ndarë me presje, p.sh., kështu:

```
int x,y;
long int dita,e,f3;
```

Më shumë deklarime të variablave mund të shkruhen në një rresht, p.sh., kështu:

```
double x,h; int g; float a,p;
```

ku, siç shihet, deklarimet e tipeve të veçanta janë ndarë mes vete me pikëpresje.

Deklarimi i tipit të njëjtë brenda një programi mund të paraqitet edhe më shumë herë, p.sh., kështu:

```
int i,j;
int s;
```

gjë që mund të bëhet edhe më shkurt në këtë mënyrë:

```
int i,j,s;
```

Deklarimi i fushave

Vektorët, matricat ose fushat shumëdimensionale deklarohen si variabla të indeksuara, duke i shkruar indekset brenda kllapave të mesme. Në gjuhën C++ indekset fillojnë me vlerën zero.

Deklarimi i vektorëve

Fushat njëdimensionale ndryshe quhen edhe vektorë. Në program ato deklarohen si variabla me një indeks. Kështu, p.sh., deklarimi i vektorit D me 5 anëtarë duket:

```
int D[5];
```

Meqë, siç u tha më sipër, indekset fillojnë me vlerën zero, pas deklarimit të mësipërm, indekset e 5 anëtarëve të vektori D janë: D[0], D[1], ..., D[4].

Shembull

Programi përmes së cilit formohet vektori A(m), duke i marrë vlerat e anëtarëve të veçantë të barabartë me katrorët e indekseve përkatës.

```
// Programi Vektori2
#include <iostream>
using namespace std;
int main()
{
     const m=5;
     int i,A[m];
     cout << "Vektori i formuar\n\n";
          for (i=0;i<m;i++)
          {
               A[i]=i*i;
               cout << "    A["
                    << i
```

```
                    << "]="
                    << A[i]
                    << endl;
          }
          cout << endl;
return 0;
}
```

Këtu, gjatë deklarimit të vektorit A, numri i anëtarëve të tij është përcaktuar përmes konstantes m, së cilës vlera 5 i është shoqëruar përmes deklarimit const.

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.2.7*.

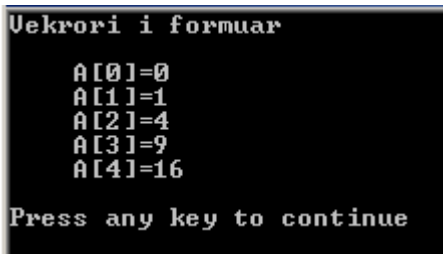


Fig.2.7
Rezultati i programit Vektori2

Deklarimi i matricave

Ngjashëm me vektorët deklarohen edhe matricat. Por, meqë matricat paraqesin fusha dydimensionale, ato deklarohen si variabla me dy indekse. Kështu, p.sh., për matricën Z me 3 rreshta dhe me nga 5 anëtarë në çdo rresht, deklarimi përkatës bëhet kështu:

```
double Z[3][5];
```

Edhe dimensionet e matricave mund të përcaktohen përmes deklarimit const.

Shembull

Programi përmes së cilit formohet matrica A(m,n), duke i llogaritur vlerat e anëtarëve të veçantë përmes prodhimit të indekseve përkatëse, por të rritur për 1.

```
// Programi Matrica1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
```

```
{
     const m=4,n=5;
     int i,j,A[m][n];
          cout << "\nMatrica e formuar A\n"
               << endl;
          for (i=0;i<m;i++)
          {
               for (j=0;j<n;j++)
               {
                    A[i][j]=i*j+1;
                    cout << setw(3)
                         << A[i][j];
               }
               cout << "\n\n";
          }
return 0;
}
```

Siç shihet nga programi i dhënë, edhe te matrica indekset fillojnë me vlerat `i=0` dhe `j=0`. Vlerat e indekseve shkojnë deri te vlerat `m-1` dhe `n-1`, gjë që përcaktohet me shprehjet `i<m` dhe `j<n` te komandat përkatëse `for`. Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.2.8*.

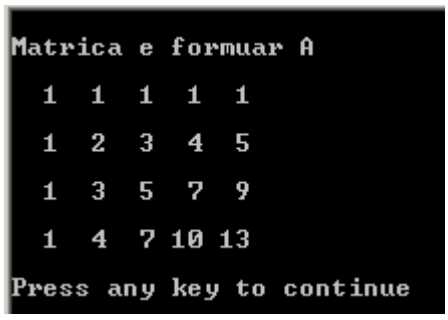


Fig.2.8
Rezultati i programit Matrica1

Deklarimi dhe inicializimi

Gjatë deklarimit të tipeve të variablave, ato mund edhe të inicializohen, përkatësisht atyre mund tu ndahen vlera. Vlerat u ndahen variablave duke e shfrytëzuar shenjën e barazimit, p.sh., kështu:

```
int i=5;
double a=5.3;
```

Shembull Programi përmes së cilit llogaritet shuma `s` e katrorëve të numrave natyrorë mes `1` dhe `m`, si dhe prodhimi `p` i këtyre numrave.

```
// Programi Natyror1
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
     int i,m;
     cout << "Vlera e variablës m: ";
     cin >> m;
     double s=0,p=1;
     for (i=1;i<=m;i++)
     {
          s=s+pow(i,2);
          p=p*i;
     }
     cout << "\nShuma s="
          << s
          << endl;
     cout << "\nProdhimi p="
          << p
          << "\n\n";
return 0;
}
```

Në program, vlerat fillestare të shumës `s` dhe të prodhimit `p` janë përcaktuar gjatë deklarimit të tipit të tyre, përmes deklarimit:

```
double s=0,p=1;
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës si vlerë hyrëse për variablën `m` kompjuterit i jepet vlera `6`, rezultati që fitohet në ekran do të duket si në *Fig.2.9*.

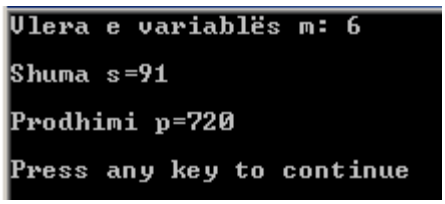


Fig.2.9
Rezultati i programit Natyror1

Vlerat e variablave që inicializohen gjatë deklarimit të tyre mund të shënohen edhe brenda kllapave, p.sh., kështu:

```
char z('b');
int k(175);
```

gjë që është e njëjtë me deklarimin:

```
char z='b';
int k=175;
```

Deklarimi dhe inicializimi i vektorëve

Njëlloj mund të deklarohen dhe të inicializohen edhe vektorët. Kështu, p.sh., deklarimi dhe inicializimi i vektorit R me 5 anëtarë të tipit intexher duket:

```
int R[5]={7,2,4,1,3};
```

Shembull Programi përmes së cilit në ekran shtypet vektori A(m), i cili gjatë deklarimit edhe inicializohet me vlera.

```
// Programi Vektori1
#include <iostream>
using namespace std;
int main()
{
     const m=8;
     int i;
     double A[m]={-5.3,4.77,1.92,-3.8,22.5};
     cout << "Vektori A"
          << "\n\n";
     for (i=0;i<m;i++)
          cout << A[i]
               << "  ";
     cout << "\n\n";
return 0;
}
```

Këtu, edhe pse vektori është deklaruar se formohet prej m=8 anëtarëve, inicializohen vetëm 5 anëtarët e parë të tij. Prandaj, kompjuteri 3 anëtarëve të tjerë u ndan vlera zero. Nëse programi i dhënë ekzekutohet, rezultati i cili shtypet në ekran do të duket si në vijim.

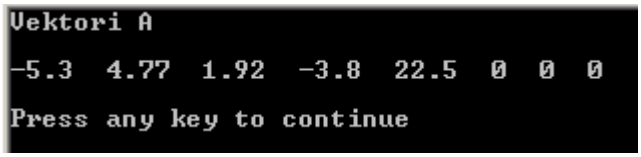


Fig.2.10
Rezultati i programit Vektori1

Deklarimi dhe inicializimi i vektorëve mund të bëhet edhe pa i përcaktuar dimensionet e tyre. Kështu, p.sh., për vektorin R, që u përmend në fillim, mund të shkruajmë:

```
int R[]={7,2,4,1,3};
```

Në këtë rast, duke i pasur parasysh vlerat e shkruara brenda kllapave, kompjuteri për vektorin R automatikisht i rezervon 5 vende, sa edhe janë shënuar anëtarë brenda kllapave.

Nëse gjatë inicializimit rezervohen më shumë vende se sa që ka vlera vektori, siç pamë edhe më lart, kompjuteri anëtarët e painicializuar i mbush me vlerat zero. P.sh., nëse shkruajmë:

```
int F[9]={3,5,2,7,4};
```

përmbajtja e vektorit F në memorien e kompjuterit është:

```
F={3,5,2,7,4,0,0,0,0}
```

Gjendje të njëjtë do të kemi edhe nëse vektori F deklarohet kështu:

```
int F[]={3,5,2,7,4,0,0,0,0};
```

Deklarimi dhe inicializimi i vektorëve, vlerat e anëtarëve të të cilëve janë të tipeve të tjera të mundshme, bëhet plotësisht njëlloj. P.sh., vektori Z me 6 anëtarë të tipit karakter deklarohet dhe inicializohet kështu:

```
char Z[7]={'d','4','*','a','G','$'};
```

Këtu, gjatë deklarimit të vektorit Z janë rezervuar 7 anëtarë, sepse në vendin e 7 të tij vendoset *karakteri zero* (`'\0'`).

Vektori i mësipërm më thjesht mund të inicializohet edhe duke i shfrytëzuar thonjëzat çift, kështu:

```
char Z[7]={"d4*aG$"};
```

Shembull Programi përmes së cilit në ekran shtypet teksti `Libri`, i cili ruhet në vektorin B të tipit karakter.

```
// Programi tekst1
#include <iostream>
using namespace std;
int main()
{
     const k=6;
     int i;
     char B[k]="Libri";
     cout << "\n";
     for (i=0;i<k;i++)
          cout << "   "
               << B[i];
     cout << "\n\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.2.11*.

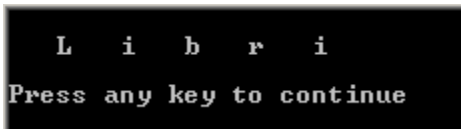


Fig.2.11
Rezultati i programit tekst1

Deklarimi dhe inicializimi i matricave

Matricat deklarohen dhe inicializohen plotësisht njëlloj si edhe vektorët. P.sh., deklarimi dhe inicializimi i matricës `K`, e cila përmban `4` rreshta dhe `3` kolona, mund të bëhet:

```
int K[4][3]={{7,4,1},
             {2,5,8},
             {3,6,2},
             {8,1,3}};
```

Këtu, gjatë inicializimit me vlera është ruajtur edhe struktura fizike e matricës. Por, për kompjuterin kjo nuk ka rëndësi, sepse plotësisht njëlloj e kupton edhe deklarimin:

```
int K[4][3]={{7,4,1},{2,5,8},{3,6,2},{8,1,3}};
```

ose edhe deklarimin te i cili anëtarët e rreshtave të veçant nuk dallohen duke i përfshirë në kllapa plotësuese:

```
int K[4][3]={7,4,1,2,5,8,3,6,2,8,1,3};
```

Shembull Programi përmes së cilit llogaritet shuma e anëtarëve të matricës katrore `R(m,m)`, të cilët gjenden nën diagonalën kryesore të saj.

```
// Programi matrica2
#include <iostream>
using namespace std;
int main()
{
     const m=5;
     int i,j,R[m][m]={{5,8,-2,6,1},
                      {3,-7,4,2,9},
                      {8,1,4,7,-2},
                      {-3,4,7,6,5},
                      {4,2,1,-3,5}};
     double s=0;
          for (i=1;i<m;i++)
               for (j=0;j<m;j++)
                    if (i>j)
                         s=s+R[i][j];
          cout << "\nShuma e kërkuar është s="
               << s
               << "\n\n";
     return 0;
}
```

Në fillim të programit të dhënë është deklaruar dhe është inicializuar matrica katrore `R` (quhet katrore, sepse ka numër të njëjtë rreshtash dhe kolonash). Pastaj, nën të është deklaruar shuma `s` dhe njëkohësisht është inicializuar me vlerën fillestare të saj `0`.

Nëse ekzekutohet programi i dhënë, si rezultat do të shtypet shuma e anëtarëve të matricës `R`, të cilët gjenden nën diagonalen kryesore (për të cilat raporti i indekseve përkatëse është `i>j`), kështu:

Fig.2.12
Rezultati i programit matrica2

```
Shuma e kërkuar është s=24

Press any key to continue
```

Deklarimi dhe inicializimi i matricave tekstuale bëhet ngjashëm.

Shembull Programi përmes së cilit inicializohet matrica `S` në të cilën ruhen

stinët e vitit dhe pastaj edhe shtypet përmbajtja e saj.

```
// Programi tekst2
#include <iostream>
using namespace std;
int main()
{
     const m=4,n=9;
     int i,j;
     char S[m][n]={"Pranvera",
                   "Vera",
                   "Vjeshta",
                   "Dimri"};
     cout << "\nStinët e vitit\n\n";
     for (i=0;i<m;i++)
     {
          cout << "   ";
          for (j=0;j<n;j++)
               cout << S[i][j];
          cout << "\n";
     }
     cout << endl;
return 0;
}
```

Në program, matrica është deklaruar me `m=4` rreshta, kurse në çdo rresht ruhen nga `n=9` karaktere (aq sa është gjatësia e emrit të stinës së parë, përfshirë edhe karakterin zero `'\0'`, i cili vendoset në fund të çdo rreshti).

Pas ekzekutimit të programit të dhënë, në ekran shtypen 4 stinët e vitit, ashtu siç shihet në *Fig.2.13*.

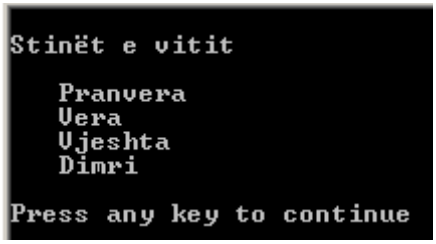


Fig.2.13
Rezultati i programit tekst2

Njëlloj deklarohen edhe fushat shumëdimensionale, me `3` e më shumë indekse.

Konstantet

Vlerat fikse, përkatësisht vlerat të cilat brenda programit nuk ndryshohen, kompjuterit mund t'i jepen duke i deklaruar si konstante. Për ruajtjen e konstanteve në memorien e kompjuterit, gjatë deklarimit të tyre shfrytëzohen identifikatorët e të gjitha tipeve të mundshme si edhe te variablat.
Kështu, p.sh., me deklarimin:

```
const double pi=3.1415926;
```

kompjuteri njoftohet se `pi` duhet të merret si konstante dhe të inicializohet me vlerën `3.1415926`.

Shembull Programi përmes së cilit llogaritet sipërfaqja dhe perimetri i rrethit me rreze r.

```
// Programi rrethi
#include <iostream>
using namespace std;
int main()
{
     const double pi=3.1415926;
     double r,s,p;
     cout << "\nRrezja e rrethit: ";
     cin >> r;
     s=pi*r*r;
     cout << "\nSipërfaqja e rrethit s="
          << s
          << endl;
     p=2*pi*r;
     cout << "\nPerimetri i rrethit p="
          << p
          << "\n\n";
return 0;
}
```

Këtu, konstanta π është deklaruar në fillim të programit si konstante dhe njëkohësisht është inicializuar me vlerën përkatëse.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës, për rrezen e rrethit kompjuterit i jepet vlera 5, rezutltati që shtypet në ekran do të duket si në *Fig.2.14*.

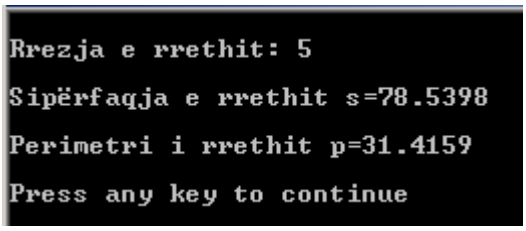


Fig.2.14
Rezultati i programit rrethi

Pas inicializimit, konstantet mund të shfrytëzohen edhe gjatë deklarimit të dimensioneve të vektorëve, p.sh., kështu:

```
const int m=5;
int A[m]={7,2,4,1,3};
```

ose edhe matricave, p.sh., kështu:

```
const int m=4,n=3;
int K[m][n]={{7,4,1},
             {2,5,8},
             {3,6,2},
             {8,1,3}};
```

Si konstante mund të deklarohen edhe vektorët, p.sh., kështu:

```
const double A[6]={2.3, -4.7, 9, -1, 7.3, 5.44};
```

ose edhe matricat, p.sh., kështu:

```
const int K[4][3]={{2,-4,3},
                   {5,6,-7},
                   {-3,9,1},
                   {0,-1,4}};
```

Me deklarimin e vektorëve dhe të matricave si konstante kompjuteri e bllokon çdo ndryshim të vlerave të anëtarëve të tyre brenda programit. Me këtë, jemi të sigurt se në program operohet me vlera fikse të vektorëve dhe të matricave, sepse çdo tentim i ndryshimit të këtyre vlerave bllokohet nga kompjuteri.

Pointerët

Adresat e hapësirës memoruese mund të ruhen duke shfrytëzuar *variabla treguese*, përkatësisht variabla të tipit *pointer* (ang. pointer). Para se të shfrytëzohen variablat e tipit pointer, ato duhet të deklarohen. Deklarimi bëhet njëlloj si edhe deklarimi i variablave, por, për të treguar se kemi të bëjmë me pointer, këtu para

variablës shënohet një yll. Rekomandohet që gjatë deklarimit të variablave të tipit pointer të bëhet edhe inicializimi i tyre me një vlerë, e cila kryesisht merret të jetë zero, p.sh., kështu:

```
int *adr=0;
```

Mosinicializimi i pointerit me një vlerë të caktuar paraqet rrezik dhe prandaj si zgjidhje rekomandohet që ky inicializim të bëhet me vlerën zero.

Adresa e një variable merret nëse para saj shënohet simboli `&`. Kështu, p.sh., për ta marrë adresën e variablës `k` dhe për ta ruajtur atë te pointeri `adr` të deklaruar dhe të inicializuar si më sipër, shkruajmë:

```
adresa=&k;
```

Shembull Programi përmes së cilit tregohet përdorimi i pointerit.

```
// Programi prg5
#include <iostream>
using namespace std;
int main()
{
     int dita;
     int *adita=0;
     dita=25;
     cout << "\nVlera e variablës: "
          << dita
          << endl;
     adita=&dita;
     cout << "Adresa e variablës: "
          << adita
          << "\n\n";
     return 0;
}
```

Në program, fillimisht është deklaruar si intexher variabla `dita` dhe është deklaruar e inicializuar me vlerë zero pointeri `adita`. Pastaj, variablës `dita` i është ndarë vlera `25` dhe përmes komandës `cout` është shtypur vlera e kësaj variable. Në fund, përmes shprehjes:

```
adita=&dita;
```

adresa e variablës `dita` ruhet te pointeri `adita`. Nëse ekzekutohet programi i dhënë, rezulteti që shtypet në ekran mund të duket si në *Fig.2.15*.

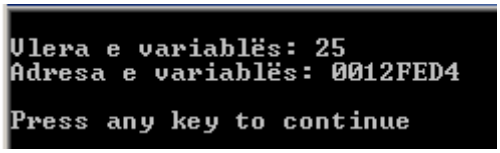


Fig.2.15
Rezultati i programit `prg5`

Adresa e shtypur paraqet një vlerë heksadecimale dhe nuk është e njëjtë nëse programi i dhënë ekzekutohet në kompjuter të ndryshëm.

Operatorët

Gjatë shkruarjes së shprehjeve për operim me të dhëna, në gjuhën `C++` shfrytëzohen operatorë të ndryshëm. Më kryesorët mes tyre janë: operatori i shoqërimit, operatorët aritmetikorë, operatorët relacionalë dhe operatorët logjikë.

Operatori i shoqërimit

Përmes operatorit të shoqërimit `=` (barazimit), *variablës së shkruar në anën e majtë të operatorit i shoqërohet vlera e cila gjendet në anën e djathtë* të këtij operatori. Kështu, p.sh., përmes shprehjeve:

```
z=5;
a=2.384;
beta=-37.4;
```

variablave `z`, `a` dhe `beta` u shoqërohen vlerat e shënuara në anët e djathta të barazimeve përkatëse. Para se të shkruhen në program shprehjet e dhëna më sipër, për variablat që figurojnë në anën e majtë të barazimit duhet patjetër të deklarohen tipet varësisht nga vlerat që u shoqërohen atyre.

Vlera që i është shoqëruar një variable ngel e pandryshuar derisa nuk i shoqërohet një vlerë e re.

Operatorët aritmetikorë

Shprehjet e ndryshme aritmetikore në gjuhën `C++` shkruhen duke i shfrytëzuar operatorët aritmetikorë që shihen në kolonën e parë të tabelës së dhënë në *Fig.2.16*.

Operatori	Operacioni	Shembull	Rezultati
+	Mbledhja	3+4	7
-	Zbritja	9-6	3
*	Shumëzimi	5*4	20
/	Pjesëtimi	8/5.	1.6
%	Moduli	8%5	3

Fig.2.16 Operatorët aritmetikorë

Përdorimi i katër operatorëve të parë (+, -, * dhe /) është i njëjtë me përdorimin e operatorëve përkatës aritmetikorë. Kurse operatori % si rezultat e jep mbetjen, përkatësisht modulin nga pjesëtimi i plotë i dy numrave të plotë. Nga shembulli i marrë në tabelë, për operatorin % shihet se mbetja nga pjesëtimi i plotë i numrit 8 me numrin 5 është 3.

Shembull

Programi përmes së cilit tregohet përdorimi i operatorit %, gjatë pjesëtimit të plotë të numrave x me numrin k, ku x ndryshohet mes vlerave 1 dhe 10, kurse vlera e variablës k kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi moduli
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int k,x,y;
     cout << "\nVlera hyrëse k: ";
     cin >> k;
     cout << "\nTabela e rezultateve\n\n";
     for (x=1;x<=10;x++)
     {
          y=x%k;
          cout << setw(3)
               << x
               << " %"
               << setw(2)
               << k
               << " ="
               << setw(2)
               << y
               << endl;
     }
     cout << endl;
return 0;
```

```
}
```

Nëse pas ekzekutimit të programit, për variablën k kompjuterit i jepet vlera 4, rezultati që paraqitet në ekran do të duket si në *Fig.2.17*.

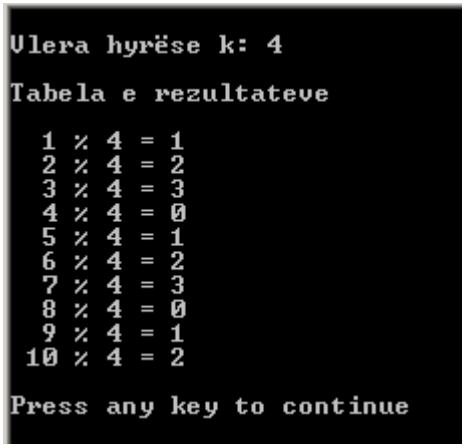


Fig.2.17
Rezultati i programit moduli1

Duhet pasur kujdes gjatë përdorimit të operatorit të pjesëtimit, kur dy vlerat që pjesëtohen janë deklaruar si numra të plotë, sepse edhe rezultati i pjesëtimit do të jetë një numër i plotë.

Shprehjet aritmetikore

Me kombinimin e variablave dhe të operatorëve aritmetikorë mund të shkruhen shprehje të ndryshme aritmetikore, ngjashëm si edhe në matematikë. Kështu, p.sh., shprehja matematikore:

```
3x+4y-5az
```

në gjuhën C++ shkruhet:

```
3*x+4*y-5*a*z
```

Brenda shprehjeve aritmetikore të shkruara në gjuhën C++, sipas nevojës mund të përdoren kllapat e vogla, plotësisht njëlloj si edhe në matematikë. P.sh., shprehja matematikore:

```
3(2x-1)-2(y+3a+b)-x/(y+2)
```

në gjuhën C++ shkruhet:

```
3*(2*x-1)-2*(y+3*a+b)-x/(y+2)
```

Përdorimi i kllapave është i domosdoshëm veçanërisht gjatë pjesëtimit, për ta përcaktuar plotësisht të pjesëtuarin dhe pjesëtuesin, sepse shumë lehtë mund të ndodhin gabime. Kllapat duhet të përdoren edhe në rastet kur paraqiten dy operatorë aritmetikorë njëri pas tjetrit. P.sh., nëse duam ta shumëzojmë me numrin `5` me vlerën negative të variablës `x`, shprehja përkatëse duhet të shkruhet:

```
5*(-x)
```

ku, duke i shfrytëzuar kllapat eliminohet përdorimi i operatorit për shumëzim dhe atij për zbritje njëri pas tjetrit.

Duke e shfrytëzuar operatorin e shoqërimit (barazimin), vlerat që fitohen nga shprehjet aritmetikore mund t'u ndahen variablave të cilat shënohen para operatorit të barazimit. Kështu, p.sh., me shprehjen e shkruar në gjuhën `C++`:

```
y=2*a+3*b-c
```

variablës `y` i shoqërohet vlera e cila llogaritet përmes shprehjes, e cila është shënuar në anën e djathtë të barazimit.

Gjatë shkruarjes së programeve në gjuhën `C++`, si edhe në gjuhët e tjera programuese përdoren edhe shprehje të cilat në matematikë janë të palogjikshme, siç është, p.sh., shprehja:

```
i=i+1
```

me të cilën në gjuhën `C++` nënkuptohet rritja për `1` e vlerës së variablës `i`. Kompjuteri, shprehjet e tilla i kupton drejt, sepse përmes tyre urdhërohet që *vlera numerike e shprehjes në anën e djathtë të barazimit t'i ndahet variablës, e cila figuron në anën e majtë të barazimit*.

Forma të shkurtuara të shprehjeve

Në gjuhën `C++`, për dallim nga gjuhët e tjera programuese, përdoren disa forma të shkurtuara të shkruarjes së shprehjeve, të cilat janë shpjeguar në vijim.

- Nëse duam që vlerën e një variable ta rrisim për `1`, variablën e shkruajmë duke shënuar pas saj `++`. Kështu, p.sh., me shprehjen:

```
c++;
```

vlera e variablës `c` rritet për `1`. Ngjashëm shkruhet edhe për zvogëlimin e variablës për `1`, p.sh., kështu:

```
c--;
```

- Për rritjen ose zvogëlimin e vlerës së një variable `v` mund të përdoren edhe shprehjet të cilat në formë të përgjithshme duken kështu:

```
v += h;
```

ose

```
v -= h;
```

ku h është hapi me të cilin rritet ose zvogëlohet vlera e variablës `v`. Kështu, p.sh., për ta rritur për `1` vlerën e variablës `c` shkruajmë:

```
c += 1;
```

kurse për ta zvogëluar për `1`, ngjashëm shkruajmë:

```
c -= 1;
```

Variablat mund të rriten ose të zvogëlohen edhe me një hap të çfarëdoshëm. P.sh., vlera e variablës `t` do të rritet për `5.5`, nëse shkruajmë:

```
t += 5.5;
```

ose do të zvogëlohet për `3.2`, nëse shkruajmë:

```
t -= 3.2;
```

Plotësisht njëlloj sikurse që përdoren operatorët `+=` (*plus-barazi*) dhe `-=` (*minus barazi*), përdoren edhe kombinimet e `3` operatorëve të tjerë aritmetikor: `*=`, `/=` dhe `%=`.

- Variabla v dhe operatori për rritje (`++`), ose operatori për zvogëlim (`--`), mund të shkruhen në dy forma:

forma prefikse: `++v` ose `--v`
forma postfikse: `v++` ose `v--`

Këtu, te verzioni me prefiks, *vlera e variablës v rritet (zvogëlohet) dhe pastaj përdoret.* Kurse te verzioni me postfiks, *vlera e variablës v përdoret dhe pastaj rritet (zvogëlohet).* Kështu, p.sh., nëse variabla `x` e ka vlerën `5` dhe në program shkruajmë:

```
a = ++x;
```

meqë kemi të bëjmë me formën prefikse të shprehjes, kompjuteri e nënkupton se së pari duhet ta rrisë vlerën e variablës x për 1 dhe atë t'ia shoqërojë variablës a. Përfundimisht, pas kësaj shprehje vlera e variablës a do të jetë 6, sa është edhe vlera e variablës x.

Nëse shprehja e mësipërme shkruhet në formën postfikse, në këtë mënyrë:

```
a = x++;
```

vlera e variablës x së pari do t'i ndahet variablës a dhe pastaj do të rritet për 1. Si rezultat i kësaj shprehje, variablës a i ndahet vlera 5, të cilën fillimisht e ka variabla x, pas së cilës vlera e variablës x rritet për 1 dhe bëhet 6.

Shembull Programi përmes së cilit tregohet përdorimi i formave të shkurtuara të shprehjeve me operatorin e mbledhjes +.

```
// Programi Operator
#include <iostream>
using namespace std;
int main()
{
     int a=5,b=3,c=6,d=4,x=7;

     cout << "\nVlera fillestare a=" << a << endl;
     a++;
     cout << "Vlera e llogaritur a=" << a << endl;

     cout << "\nVlera fillestare b=" << b << endl;
     b += 6;
     cout << "Vlera e llogaritur b=" << b << endl;

     cout << "\nVlera fillestare c=" << c << endl;
     c = ++x;
     cout << "Vlera e llogaritur c=" << c << endl;

     cout << "\nVlera fillestare d=" << d << endl;
     d = x++;
     cout << "Vlera e llogaritur d=" << d << endl;
     cout << endl;
return 0;
}
```

Programi i dhënë është shkruar ashtu që, për çdo variabël, së pari të shtypet vlera me të cilën ajo inicializohet gjatë deklarimit të tipit të saj dhe pastaj edhe vlera që fitohet pas llogaritjes përmes formave të shkurtuara të shkruarjes së shprehjeve. Rezultatet që fitohen në ekran do të duken si në *Fig.2.18*.

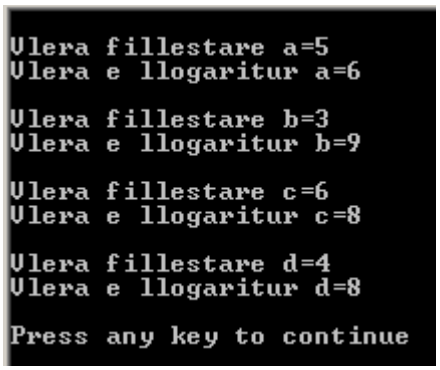


Fig.2.18
Rezultati i programit operator

- Me qëllim që disa variablave t'u shoqërohet një vlerë e caktuar, në gjuhën C++ mund të shfrytëzohen, p.sh., edhe shprehje të formës:

```
a=b=c=d=55;
```

Radha e ekzekutimit të operatorëve

Gjatë llogaritjes së vlerave numerike të shprehjeve aritmetikore, kompjuteri e ka parasysh listën e prioritetit të ekzekutimit të tyre, e cila për operatorët aritmetikorë duket kështu:

```
1.  *   /   %
2.  +   -
```

ku me prioritet më të madh janë 3 operatorët e parë, kurse mbledhja dhe zbritja ekzekutohen pas tyre. Kështu, p.sh., gjatë ekzekutimit të shprehjes:

```
y=3*a+b/c-2;
```

së pari kryhet shumëzimi, pastaj pjesëtimi, mbledhja dhe në fund zbritja. Nëse, p.sh., variablat e veçanta i kanë vlerat: a=5, b=10, c=2.5, llogaritja e vlerës së variablës y në hapat e veçantë do të rrjedhë kështu:

```
1.  3 * a = 3 * 5 = 15
2.  b / c = 10 / 2.5 = 4
3.  1 + 2 = 15 + 4 = 19
4.  3 - 2 = 19 - 2 = 17
```

ku me numra të nxier janë paraqitur hapat e veçantë, kurse vlera që fitohet në hapin e fundit (numri 17) është vlera e llogaritur për variablën y.

Nëse në shprehje paraqiten më shumë operatorë të rangut të njëjtë, ata *ekzekutohen duke shkuar prej anës së majtë kah ana e djathtë e shprehjes*. Kështu, p.sh., nëse janë dhënë vlerat: y=5, a=2.5, b=4, c=9 e d=2, për shprehjen:

```
z=2*y/a-3*b+c/2*d+4
```

radha e ekzekutimit të operatorëve të veçantë dhe vlerat përkatëse janë:

```
1.  2*y=2*5=10
2.  1/a=10/2.5=4
3.  3*b=3*4=12
4.  c/2=9/2=4.5
5.  4*d=4.5*2=9
6.  2-3=4-12=-8
7.  6+5=-8+9=1
8.  7+4=1+4=5
```

Nga shembulli i dhënë shihet se, p.sh., së pari kryhen operacionet e shumëzimit dhe pjesëtimit. Por, në hapin e **1.** kryhet shumëzimi, sepse ai gjendet i pari kur shkohet prej anës së majtë të shprehjes dhe pastaj në hapin e **2.** kryhet pjesëtimi, meqë kur shkohet djathtas takohet menjëherë pas shumëzimit.

Nëse në shprehje paraqiten kllapa, ngjashëm si edhe në matematikë, së pari do të kryhen pjesët e shprehjes të shkruara brenda kllapave. Nëse në shprehje ka më shumë kllapa, për kryerjen e tyre shkohet prej anës së majtë kah ana e djathtë e shprehjes. Gjatë ekzekutimit të shprehjeve brenda kllapave ato merren si shprehje të veçanta dhe për operatorët e rangut të njëjtë shkohet gjithashtu prej anës së majtë kah ana e djathtë e shprehjes brenda kllapave. Kështu, p.sh., nëse na është dhënë shprehja:

```
g=3*(2*a-3*b)+2*(x+6/a)-4*b+1
```

dhe nëse për variablat e veçanta janë dhënë këto vlera: `a=4`, `b=2` dhe `x=4`, ekzekutimi i operatorëve do të rrjedhë në këta hapa:

```
1.  2*a=2*4=8
2.  3*b=3*3=6
3.  1-2=8-6=2

4.  6/a=6/4=1.5
5.  x+4=4+1.5=5.5

6.  3*3=3*2=6
7.  2*5=2*5.5=11
```

```
8.  4*b=4*2=8

9.  6+7=6+11=17
10. 9-8=17-8=9
11. 10+1=9+1=10
```

Vlera `10` që fitohet në hapin e fundit e paraqet vlerën e shprehjes në anën e djathtë të barazimit, e cila i shoqërohet variablës `g`.

Kur kllapat përfshihen brenda kllapave të tjera, përparësi në kryerje kanë pjesët e shprehjeve që gjenden në kllapat e brendshme.

Konvertimi i tipit të të dhënave numerike

Nëse në një shprehje aritmetikore paraqiten të dhëna të tipeve të ndryshme, kompjuteri tenton t'i konvertojë në të dhëna të tipit të njëjtë. Gjatë kësaj, konvertimi do të jetë i suksesshëm, nëse në shprehje paraqiten vetëm të dhëna të natyrës së njëjtë.

Për shprehjet me të dhëna numerike, në procesin e konvertimit merret parasysh rangimi vijues i të dhënave:

```
char
short
int
long
float
double
long double
```

sipas së cilit lejohet konvertimi në të dhëna të rangut më të lartë. Kështu, p.sh., lejohet konvertimi i të dhënave `long`, në të dhëna të tipit `float`, `double` dhe `long double`. Por, gjatë konvertimeve të kundërta, p.sh., prej të dhënave `long`, në të dhëna të tipit `int`, `short`, ose `char`, rezultati do të jetë me gabime. Lejimi ose moslejimi i një konvertimi lidhet me hapësirat memoruese që shfrytëzohen nga të dhënat e tipeve të ndryshme.

Operatorët relacionalë

Në grupin e operatorëve relacionalë bëjnë pjesë operatorët të cilët përdoren për krahasimin e të dhënave, përkatësisht për testimin e raporteve mes tyre. Pas krahasimit përmes operatorëve relacionalë, si rezultat fitohen vlerat logjike `true` ose `false`. Nëse një relacion është i vërtetë, rezultati i tij është `true`, kurse për relacionet e pavërteta rezultati është `false`.

Operatorët relacionalë, të cilët përdoren në gjuhën C++, janë dhënë në tabelën e *Fig.2.19*, ku për variablat a dhe b janë marrë vlerat 4 dhe 7.

Operatori	Domethënia	Shembull	Rezultati
<	Më i vogël se	(a+1) < b	true
<=	Më i vogël se, ose barazi me	(3*a-2) <= (a+2*b)	false
==	Barazi me	(a+3)==7	true
>	Më i madh se	(b+2*a)>(3*a)	true
>=	Më i madh se, ose barazi me	(a+3*b-1) >= (4*b)	false
!=	Jobarazi me	(8*a-2*b) != (3*a+2)	true

Fig.2.19 Operatorët relacionalë

Shembull Programi përmes së cilit krahasohen vlerat numerike të variablave x dhe y, të cilat kompjuterit i jepen përmes tastierës.

```
// Programi relacioni1
#include <iostream>
using namespace std;
int main()
{
     bool a;
     int x,y;
     cout << "\nVlera e variablës x: ";
     cin >> x;
     cout << "\nVlera e variablës y: ";
     cin >> y;
          if (x == y)
               a=true;
          else
               a=false;
     cout << "\nRezultati:\n";
          if (a != true)
               cout << "\nNumrat nuk janë të barabartë\n";
          else
               cout << "\nNumrat janë të barabartë\n";
     cout << "Vlera e variablës logjike është a="
          << a
          << "\n\n";
return 0;
}
```

Në program, fillimisht variabla a është deklaruar si variabël logjike. Pastaj, kompjuterit i urdhërohet që t'i lexojë vlerat numerike të variablave x dhe y, të cilat gjatë ekzekutimit të programit duhet t'i jepen përmes tastierës. Nëse, p.sh.,

kompjuterit i jepen vlerat hyrëse `x=5` dhe `y=7`, rezultati në ekran do të duket si në *Fig.2.20*.

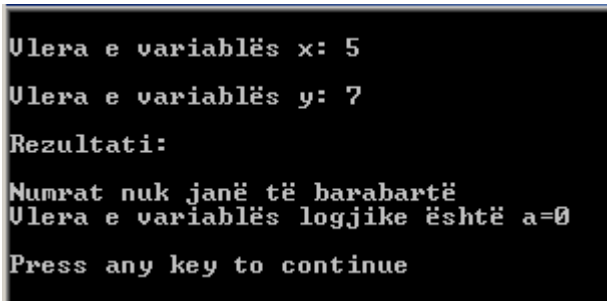


Fig.2.20
Rezultati i programit relacioni1

Këtu, meqë vlerat e variablave `x` dhe `y` janë të ndryshme, pas krahasimit variabla logjike `a` do ta marrë vlerën `false`. Por, kur shtypet, kompjuteri për vlerën logjike `false` e shtyp numrin `0` (kurse për `true` do ta shtypë numrin `1`, ose ndonjë numër tjetër të plotë, por më të madh se `0`).

Operatorët logjikë

Për krahasimin e më shumë shprehjeve njëkohësisht përdoren operatorët logjikë të dhënë në tabelën e *Fig.2.21*.

Operatori	Operacioni	Shembull	Rezultati
&&	Konjuksioni, AND	`(x < 7) && (y ==5)`	`true`
||	Disjunksioni, OR	`(x != 2) || (x > 3)`	`false`
!	Negacioni, NOT	`!(y > 4)`	`false`

Fig.2.21 Operatorët logjikë

Këtu, rezultatet në kolonën e fundit të tabelës fitohen nëse, p.sh., për variablat `x` dhe `y` merren vlerat `2` dhe `5`.

Operatori &&

Për paraqitje të operacionit logjik AND (DHE) shfrytëzohet operatori `&&`. Rezultati i kryerjes së operacionit `AND` mbi dy operandë do të jetë `true`, nëse vlerat e të dy operandëve janë `true`.

Shembull Programi përmes së cilit tregohet përdorimi i operatorit logjik `&&`.

```
// Programi And
#include <iostream>
using namespace std;
```

```
int main()
{
     int a=5, b=3;
     bool x;
     x=(a > b) && (a == (b+2));
     cout << "Vlera e variablës x="
          << x
          << endl;
     return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran si rezultat do të shtypet mesazhi:

```
Vlera e variablës x=1
```

sepse është e saktë (`true`) vlera e shprehjes relacionale:

```
x=(a > b) && (a == (b+2))
```

Gjatë llogaritjes së vlerës së kësaj shprehje, fillimisht llogariten vlerat e dy pjesëve të saj:

```
(a > b)
```

dhe

```
(a == (b+2))
```

të cilat, në bazë të vlerave të variablave `a` dhe `b`, janë të sakta (`true`). Në fund, rezultati i operatorit `==` është `true`, sepse vlerat e të dy operandëve të saj janë `true` (janë të barabarta).

Operatori ||

Për paraqitje të operacionit logjik `OR` (`OSE`) shfrytëzohet operatori `||`. Rezultati i kryerjes së operacionit logjik `OR` mbi dy operandë do të jetë `true`, nëse së paku vlera e njërit operand është `true`.

Shembull Programi përmes së cilit tregohet përdorimi i operatorit logjik `||`.

```
// Programi Or
#include <iostream>
using namespace std;
int main()
{
     int a=5, b=3;
     bool x;
     x=(a < 8) || (a <= (b+1));
     cout << "Vlera e variablës x="
          << x
          << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran si rezultat do të shtypet mesazhi:

```
Vlera e variablës x=1
```

sepse është e saktë vlera e shprehjes relacionale:

```
x=(a < 8) || (a <= (b+1));
```

meqë për vlerat e marra të variablave `a` dhe `b` në fillim të programit është e saktë njëra prej dy shprehjeve relacionale të cilat paraqiten si operand, përkatësisht është e saktë shprehja `(a < 8)`.

Operatori !

Për paraqitje të operacionit logjik `NOT` (JO) shfrytëzohet operatori `!`. Rezultati i kryerjes së operacionit logjik `JO` mbi një operand do të jetë `true`, nëse vlera e operandit është `false`, ose zero.

Shembull Programi përmes së cilit tregohet përdorimi i operatorit logjik `!`.

```
// Programi Not
#include <iostream>
using namespace std;
int main()
{
     int a=5, b=3;
     bool x;
     x=!(a > b);
     cout << "Vlera e variablës x="
          << x
          << endl;
```

```
return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran si rezultat do të shtypet mesazhi:

```
Vlera e variablës x=0
```

sepse vlera e shprehjes logjike:

```
x=!(a > b);
```

është 0 (`false`), meqë operandi i cili është shënuar pas operatorit `!`, përkatësisht shprehja relacionale `(a > b)` është e saktë (`true`).

Operatori i kushtëzuar

Në gjuhën `C++` përdoret një operator i veçantë dypjesësh `?:`, për llogaritje të kushtëzuar. Shprehjet që formohen duke e shfrytëzuar operatorin e kushtëzuar në formë të përgjithshme duken kështu:

```
y = k ? a : b;
```

Kompjuteri, sa herë që i takon shprehjet e kësaj forme, nëse kushti `k` është i saktë, variablës `y` ia ndan vlerën e shprehjes `a`, përndryshe ia ndan vlerën e shprehjes `b`, gjë që në matematik shprehet kështu:

$$y = \begin{cases} a & \text{për } k = true \\ b & \text{për } k = false \end{cases}$$

Shembull Programi përmes së cilit tregohet përdorimi i operatorit të kushtëzuar `?:`.

```
// Programi Kusht
#include <iostream>
using namespace std;
int main()
{
```

```
     int a=8, b=2;
     int y;
     y=(a>b) ? (2*a+1) : (3*b-2);
     cout << "Vlera e variablës y="
          << y
          << endl;
return 0;
}
```

Në program, operatori i kushtëzuar është shfrytëzuar gjatë përcaktimit të vlerës së shprehjes `y`. Përmes kësaj shprehje, merret se `y=2*a+1`, nëse `a>b`, përndryshe merret se `y=3*b-2`. Meqë, në bazë të vlerave të variablave `a` dhe `b` plotësohet kushti `a>b`, kompjuteri si rezultat do të shtypë:

```
Vlera e variablës y=17
```

sepse llogaritet përmes shprehjes `y=2*a+1`.

Komentet

Që programi të jetë i kuptueshëm, qoftë edhe pas një kohe më të gjatë, ose edhe nga shfrytëzues të tjerë, në pjesë të ndryshme të tij mund të shkruhen tekste, të cilat njihen edhe si *komente* (ang. comment).

Në gjuhën `C++` komentet mund të shkruhen në dy mënyra:

komente brenda një rreshti (ang. end-of-line comment) dhe
komente brenda një blloku (ang. block comment).

Gjatë kompajlimit të programit, kompjuteri komentet i eliminon, sepse ato shfrytëzohen vetëm nga përpiluesi i programit.

Komentet brenda një rreshti

Komentet brenda një rreshti fillojnë me dy vija të pjerrëta `//`, dikund pas komandës së shkruar në atë rresht dhe vazhdojnë deri në fund të rreshtit.

Shembull Programi përmes së cilit gjendet vlera e anëtarit më të vogël në vektorin e dhënë `A(m)`.

```
// Programi Koment1
#include <iostream>
using namespace std;
int main()
{
```

```
     const int m=5;
     int i, b;
     int A[m]={7,-3,4,9,-2};
     b=A[0];
     for (i=0;i<m;i++)
          if (A[i] < b)
             b=A[i];
     cout << "Anëtari më i vogël: "
          << b
          << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, si rezultat do të shtypet:

```
Anëtari më i vogël: -3
```

meqë në mesin e vlerave të anëtarëve të vektorit `A`, vlera më e vogël është `-3`.

Në fillim të programit është dhënë një rresht me koment, i cili fillon me dy vijat e pjerrëta:

```
// Programi Koment1
```

ku është shënuar emri `Koment1`, me të cilin emër njëkohësisht programi është ruajtur edhe në disk.

Në program mund të shkruhen edhe komente të tjera, p.sh., ashtu siç shihet në vijim.

```
// Programi Koment2
// përmes këtij programi gjendet anëtari
// më i vogël në vektorin e dhënë A(m)
#include <iostream>
using namespace std;
int main()
{
     const int m=5;
     int i, b;                    // Deklarimi i variablave
     int A[m]={7,-3,4,9,-2};      // Vlerat e vektorit
     b=A[0];                      // Vlera fillestare
     for (i=0;i<m;i++)
          if (A[i] < b)
             b=A[i];              // Ndërrimi i vlerës
     cout << "Anëtari më i vogël: "
          << b
          << endl;
return 0;
}
```

Komente brenda bllokut

Shpesh për komente nevojiten tekste më të gjata. Për këtë qëllim komentet shkruhen brenda bllokut i cili fillon me `/*` dhe përfundon me `*/`. Në këtë mënyrë mund të shkruhen edhe komentet brenda një rreshti. Në vijim është dhënë programi `Koment3`, tek i cili komentet janë shënuar brenda blloqeve.

```
/* Programi Koment3
   përmes këtij programi gjendet anëtari
   më i vogël në vektorin e dhënë A(m) */
#include <iostream>
using namespace std;
int main()
{
     const int m=5;
     int i, b;                    /* Deklarimi i variablave */
     int A[m]={7,-3,4,9,-2};      /* Vlerat e vektorit */
     b=A[0];                      /* Vlera fillestare */
     for (i=0;i<m;i++)
          if (A[i] < b)
             b=A[i];              /* Ndërrimi i vlerës */
     cout << "Anëtari më i vogël: "
          << b
          << endl;
return 0;
}
```

Zbrazësirat brenda programit

Me qëllim të rritjes së dukshmërisë së programit, përkatësisht copëtimit të tij në pjesë, të cilat paraqesin tërësi të veçanta, në program mund të shtohen rreshta të zbrazët.

Shembull

Programi përmes së cilit në vektorin e dhënë `A(m)` numërohen anëtarët negativë (`n`), anëtarët me vlerë zero (`z`) dhe anëtarët pozitivë (`p`).

```
// Programi Numrimi
#include <iostream>
using namespace std;
int main()
{
     const int m=9;
```

```
     int i,n,z,p;

     int A[m]={3,-8,0,-4,5,9,-6,0,2};

     n=0;
     z=0;
     p=0;
     cout << "\nR e z u l t a t i\n\n";
     for (i=0;i<m;i++)
     {
          if (A[i]<0)
             n=n+1;
          else
             if (A[i]==0)
                z=z+1;
             else
                p=p+1;
     }

     cout << "Numri i anëtarëve negativë n="
          << n
          << endl;

     cout << "Numri i anëtarëve zero z="
          << z
          << endl;

     cout << "Numri i anëtarëve pozitivë p="
          << p
          << "\n\n";

return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të shihet si në *Fig.2.22*.

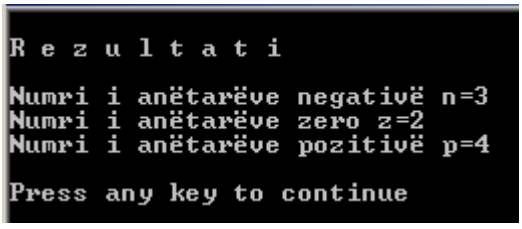


Fig.2.22

Rezultati i programit Numrimi

Edhe komandat e veçanta mund të shkruhen duke lënë zbrazësira brenda tyre, ose mund të shkruhen edhe në më shumë rreshta. Kështu, p.sh., shprehja:

```
y=3*a-2*b+4*(a+2*x-c);
```

mund të shkruhet në këtë mënyrë:

```
y = 3*a
  - 2*b
  + 4*(a+2*x-c);
```

ose edhe kështu:

```
y = 3*a
       - 2*b
            + 4*(a+2*x-c);
```

gjë që, kur kemi të bëjmë me shprehje të gjata, mund ta rrisim kontrollin mbi saktësinë e shkruarjes së tyre.

3

Leximi dhe shtypja

Të dhënat të cilat shfrytëzohen gjatë llogaritjeve në programe të ndryshëm kompjuterit i jepen duke i deklaruar ato si konstante, ose duke ua ndarë variablave përkatëse përmes operatorit të shoqërimit. Por, vendosja e tillë e të dhënave në memorien e kompjuterit është jopraktike, sepse shfrytëzuesi i programit, për t'i ndryshuar të dhënat që përpunohen, duhet të ketë qasje në verzionin burimor të programit. Për këtë qëllim, në rast të përgjithshëm, kompjuterit të dhënat i jepen përmes tastierës, e cila llogaritet si njësi hyrëse standarde. Gjatë kësaj, në program duhet të shkruhen komandat përkatëse për leximin e të dhënave.

Rezultatet e fituara kompjuterit mund t'i urdhërohet t'i shtypë në ekran, duke i vendosur komandat për shtypje brenda programit.

Në fillim të çdo programi të shkruar në C++, në të cilin shfrytëzohen komandat për lexim dhe shtypje, patjetër duhet të vendoset komanda paraprocesorike:

```
#include <iostream>
```

përmes së cilës mundësohet leximi i të dhënave të cilat i jepen kompjuterit, si dhe shtypja e rezultateve.

Leximi dhe shtypja e zakonshme

Forma më e thjeshtë e komandës për leximin e vlerës së variablës v duket kështu:

```
cin >> v;
```

Pas ekzekutimit të kësaj komande, kompjuteri ndalet dhe pret që përmes tastierës t'i jepet vlera e variablës v. Që kompjuteri ta marrë vlerën, pasi ajo të jetë, në tastierë duhet të shtypet edhe tasti Enter.

Komanda për shtypje të vlerës së variablës v, në formën e saj më të thjeshtë, shkruhet kështu:

```
cout << v;
```

Shembull Programi përmes së cilit lexohet vlera numerike e variablës `x` dhe pastaj e njëjta edhe shtypet në ekran.

```
// Programi Lexo1
#include <iostream>
using namespace std;
int main()
{
     int x;
     cin >> x;
     cout << x;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, fillimisht në ekran do të paraqitet kursori në formë të një vize për nënvizim, me çka kompjuteri na njofton se është duke e pritur vlerën e variablës `x` që t'ia shkruajmë përmes tastierës. Pastaj, p.sh., nëse përmes tastierës e shkruajmë numrin `15` dhe e shtypim tastin `Enter`, në ekran do të shtypet numri `15`. Por, në vazhdim të numrit të shtypur do të shkruhet edhe mesazhi të cilin kompjuteri e gjeneron automatikisht, kështu:

```
15Press any key to continue
```

Me qëllim të eliminimit të kësaj pamje, kompjuterit duhet t'i urdhërojmë që pas shtypjes së vlerës së variablës `x` të kalojë në rresht të ri. Për këtë qëllim, në program duhet të shtohet edhe një komandë për shtypje, në këtë mënyrë:

```
     cout << x;
     cout << endl;
```

pas së cilës programi do të duket kështu:

```
// Programi Lexo2
#include <iostream>
using namespace std;
int main()
{
     int x;
     cin >> x;
     cout << x;
     cout << endl;
return 0;
}
```

Në këtë rast, pas shtypjes së vlerës së variablës x, me ekzekutimin e komandës së dytë për shtypje, përmes manipulatorit `endl`, kompjuterit i urdhërohet kalimi në rresht të ri. Rezulati që fitohet pas shtimit të komandës në fjalë, nëse përmes tastierës kompjuterit i jepet vlera `15`, do të duket kështu:

```
15
Press any key to continue
```

Përfshirja e teksteve në rezultate

Rezultatet që shtypen rregullisht shoqërohen edhe me tekste të ndryshme. Tekstet që duhet të shtypen në ekran shkruhen në kuadër të komandës për shtypje `cout`, duke i vendosur ato brenda thonjëzave (`"`). Kështu, p.sh., nëse në programin `lexo1` të dhënë më sipër, shkruhen dy komanda për shtypje:

```
cout << "Vlera e lexuar x=";
cout << x;
```

gjë që në program duket kështu:

```
// Programi Lexo3
#include <iostream>
using namespace std;
int main()
{
     int x;
     cin >> x;
     cout << "Vlera e lexuar x=";
     cout << x;
return 0;
}
```

rezultati që shtypet në ekran do të duket:

```
Vlera e lexuar x=15
```

Në këtë rast, kompjuteri së pari e ekzekuton komandën e parë për shtypje, përmes së cilës teksti i shkruar nën thonjëza përshkruhet në ekran:

Vlera e lexuar x=

Pastaj, me komandën e dytë, në vazhdim të tekstit shtypet vlera e lexuar e variablës `x`.

Dy komandat për shtypje që u dhanë më sipër mund të shkruhen si një komandë, kështu:

```
cout << "Vlera e lexuar x=" << x;
```

ose, me qëllim të pamjes më të mirë edhe në këtë mënyrë:

```
cout << "Vlera e lexuar x="
     << x;
```

gjë që do të praktikohet në vijim të këtij teksti.

Gjatë ekzekutimit të programit të mësipërm, përveç rezultatit i cili shtypet përmes komandave për shtypje, në vazhdim kompjuteri automatikisht e shtyp edhe mesazhin:

```
Press any key to continue
```

Nëse në program komanda e mësipërme për shtypje shkruhet në këtë mënyrë:

```
cout << "Vlera e lexuar x="
     << x
     << endl;
```

përkatësisht nëse programi duket kështu:

```
// Programi Lexo4
#include <iostream>
using namespace std;
int main()
{
     int x;
     cin >> x;
     cout << "Vlera e lexuar x="
          << x
          << endl;
return 0;
}
```

rezultati që shtypet në ekran, si dhe mesazhi i përmednur më sipër, do ta kenë këtë pamje:

```
Vlera e lexuar x=15
Press any key to continue
```

Urdhri për kalim në rresht të ri, me efekt të njëjtë, mund të jepet përmes shtypjes së *karakterit për rresht të ri* (ang. newline character) `\n`, i cili shënohet nën thonjëza:

```
cout << "Vlera e lexuar x="
     << x
     << "\n";
```

Gjatë kësaj, në vend të thonjëzave të zakonshme `"\n"`, karakteri për rresht të ri mund të shkruhet edhe nën thonjëza të njëfishta `'\n'`.

Shtypja e teksteve gjatë leximit

Me qëllim që shfrytëzuesi i programit të informohet për vlerën që kompjuterit duhet t'i jepet përmes tastierës, para komandës për lexim `cin`, përmes komandës `cout` mund të shtypet një tekst përcjellës. Kështu, p.sh., nëse te programi i mësipërm, para komandës për lexim, shtohet komanda:

```
cout << "Vlera hyrëse x: ";
```

programi do të duket si në vijim.

```
// Programi Lexo5
#include <iostream>
using namespace std;
int main()
{
     int x;
     cout << "Vlera hyrëse x: ";
     cin >> x;
     cout << "Vlera e lexuar x="
          << x
          << "\n";
return 0;
}
```

Nëse pas kësaj ekzekutohet programi, në ekran fillimisht do të paraqitet teksti:

```
Vlera hyrëse x:
```

dhe kompjuteri do të presë që përmes tastierës të shkruhet vlera përkatëse e variablës `x`. Nëse, p.sh., në tastierë shkruhet numri `9`, ai do të vendoset në vazhdim të tekstit të sipërm, kështu:

```
Vlera hyrëse x: 9
```

Në fund, pas shtypjes së tastit `Enter` në tastierë, përmes komandës dalëse `cout` do të shtypet edhe rezultati, për ta fituar në ekran pamjen përfundimtare e cila shihet në *Fig.3.1*.

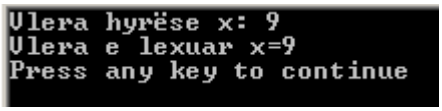


Fig.3.1
Rezultati i programit Lexo5

ku, siç thamë edhe më sipër, rreshti i parë lidhet me komandën e parë `cout` dhe komandën `cin`, kurse rreshti i dytë me komandën e dytë `cout`.

Leximi dhe shtypja e disa vlerave

Nëse njëkohësisht duhet të lexohen vlerat e disa variablave, leximi mund të bëhet me komanda të veçannta për lexim, ose me vetëm një komandë.

Leximi me komanda të veçanta

Kur në program paraqiten më shumë komanda për lexim, kompjuteri pas ekzekutimit të secilës prej tyre do të presë që përmes tastierës t'i jepen vlerat e variablave përkatëse që lexohen.

Shembull Programi përmes së cilit lexohen dhe shtypen vlerat numerike të variablave `a` dhe `b`.

```
// Programi Lexo6
#include <iostream>
using namespace std;
int main()
{
     int a,b;
     cin >> a;
     cout << "Vlera e lexuar a="
          << a
          << "\n";
     cin >> b;
     cout << "Vlera e lexuar b="
          << b
          << "\n";
     return 0;
}
```

Gjatë ekzekutimit të programit të dhënë, kompjuteri do të ndalet pas secilës komande `cin`, për t'ia dhënë përmes tastierës vlerat e variablave të shënuara në vazhdim të këtyre komandave. Nëse, p.sh., kompjuterit i jepen vlerat 5 dhe -3, duke e shtypur pas secilës vlere tastin `Enter`, si rezultat i komandave `cout`, në ekran do të shtypen `2` rreshta me rezultate, kështu:

```
Vlera e lexuar a=5
Vlera e lexuar b=-3
```

Në fakt, duke i pasur parasysh edhe vlerat të cilat ia japim kompjuterit përmes tastierës, në ekran do ta kemi pamjen e dhënë në *Fig.3.2*.

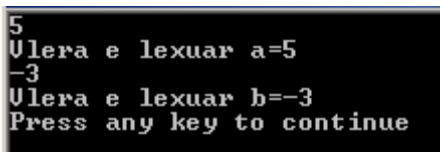


Fig.3.2
Rezultati i programit Lexo6

Gjatë leximit të vlerave për variablat `a` dhe `b`, që të përcaktohet saktë se cilën vlerë është duke e pritur kompjuteri, paraprakisht jepet komanda për shtypje të tekstit përcjellës. Kështu, p.sh., programi i dhënë më sipër mund të shkruhet si në vijim.

```
// Programi Lexo7
#include <iostream>
using namespace std;
int main()
{
     int a,b;
     cout << "Shtype në tastierë numrin a: ";
     cin >> a;
     cout << "Vlera e lexuar a="
          << a
          << "\n";
     cout << "Shtype në tastierë numrin b: ";
     cin >> b;
     cout << "Vlera e lexuar b="
          << b
          << "\n";
     return 0;
}
```

Këtu, para se kompjuterit t'i jepen përmes tastierës vlerat e variablave `a` dhe `b`, në ekran do të paraqiten mesazhet përcjellëse, së pari:

```
Shtype në tastierë numrin a:
```

dhe pastaj edhe mesazhi i dytë:

```
Shtype në tastierë numrin b:
```

Nëse përmes tastierës kompjuterit i jepen vlerat e përmendura më sipër, pas përfundimit të ekzekutimit të programit, në ekran do ta kemi pamjen e dhënë në *Fig.3.3*.

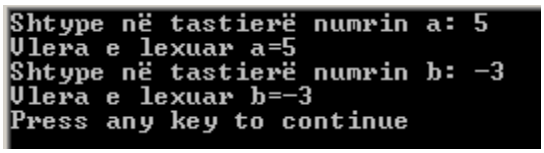


Fig.3.3
Rezultati i programit Lexo7

Leximi me vetëm një komandë

Kompjuteri mund të urdhërohet që me një komandë `cin` t'i lexojë vlerat e më shumë variablave. Kështu, për leximin e vlerave të variablave `v1`, `v2`, ..., `vn`, komanda përkatëse `cin` do ta ketë pamjen:

```
cin >> v1
    >> v2
    .....
    >> vn;
```

Gjatë ekzekutimit të programit, në të cilin paraqitet një komandë e tillë, kompjuterit duhet t'i jepen përmes tastierës njëkohësisht vlerat e të gjitha variablave, duke lënë mes çdo vlere së paku një zbrazësirë.

Shembull

Programi përmes së cilit lexohen vlerat numerike të variablave `a, b` dhe `c`.

```
// Programi Lexo8
#include <iostream>
using namespace std;
int main()
```

```
{
     int a,b,c;
     cin >> a
         >> b
         >> c;
     cout << "Vlera e lexuar a="
          << a
          << "\n";
     cout << "Vlera e lexuar b="
          << b
          << "\n";
     cout << "Vlera e lexuar c="
          << c
          << "\n";
return 0;
}
```

Gjatë ekzekutimit të programit, kompjuteri do të ndalet dhe do të presë që përmes tastierës t'ia shkruajmë vlerat e të tri variablave që lexohen, të ndara mes vete me zbrazësirë, p.sh., kështu:

```
5 -3 9
```

Në fund, kur të shkruhen vlerat dhe të shtypet tasti `Enter`, në ekran do ta kemi pamjen e dhënë në *Fig.3.4*.

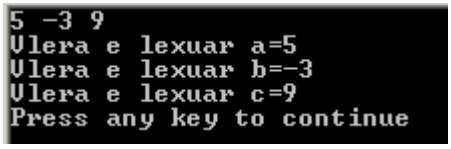


Fig.3.4
Rezultati i programit Lexo8

Shtypja në një hapësirë të caktuar

Gjatë shtypjes së rezultateve shpesh nevojitet që paraprakisht të përcaktohet numri i vendeve që do të shfytëzohen për shtypje të tyre, përkatësisht hapësira në të cilën shtypet rezultati. Për këtë qëllim brenda komandës `cout` mund të shfrytëzohet manipulatori `setw`, komanda `cout.width` ose sekuenca dalëse me karakterin për tabelim horizontal `\t`.

Manipulatori setw

Në formë të përgjithshme, manipulatori `setw` shkruhet kështu:

```
setw(k)
```

ku `k` është numri i vendeve që shfrytëzohen për shtypjen e rezultatit. Kështu, p.sh., për ta shtypur vlerën e variablës `x` në `8` vende, komanda `cout`, bashkë me manipulatorin setw brenda saj, duhet të shkruhet në këtë mënyrë:

```
cout << setw(8)
     << x;
```

Rezultati i cili fitohet pas ekzekutimit të kësaj komande, nëse `x` e ka vlerën `15`, në ekran do të duket kështu:

```
□□□□□□15
```

ku me simbolin □ janë shënuar zbrazësirat të cilat mbesin para numrit gjatë shtypjes së tij në hapësirën e rezervuar përmes manipulatorit `setw`.

Shembull

Programi përmes së cilit gjendet sipërfaqja `s` dhe perimetri `p` i katrorit me brinjën `a`, duke marrë për `a` vlera të ndryshme mes `1` dhe `10`.

```
// Programi Katrori1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     float a,s,p;
     for (a=1;a<=10;a++)
     {
          s=a*a;
          p=4*a;
          cout << setw(5)
               << a
               << setw(7)
               << s
               << setw(6)
               << p
               << endl;
     }
```

```
return 0;
}
```

Këtu, për 3 vlerat që shtypen janë përcaktuar numra të ndryshëm vendesh për shtypjen e tyre. Rezultati që shtypet në ekran pas ekzekutimit të programit do të duket si në *Fig.3.5*.

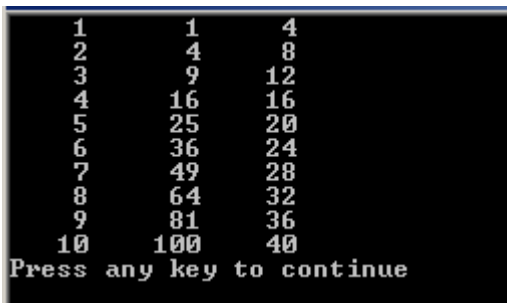


Fig.3.5
Rezultati i programit Katrori1

Siç shihet nga rezultati i dhënë, për shtypjen e vlerave të brinjës `a` janë shfrytëzuar `5` vende, gjë që përcaktohet me manipulatorin `setw(5)`, kurse sipërfaqja `s` dhe perimetri `p` janë shtypur në `7` dhe `6` vende - përkatësisht, duke i shfrytëzuar manipulatorët përkatës.

Në fillim të programit të dhënë më sipër është vendosur komanda paraprocesorike:

```
#include <iomanip>
```

sepse brenda fajllit `iomanip` është përcaktuar manipulatori `setw`.

Manipulatori `setw` shfrytëzohet edhe gjatë shtypjes së teksteve.

Shembull

Programi përmes së cilit në ekran shtypet teksti `Lapsi dhe libri`, duke e shkruar çdo shkronjë në një rresht të ri dhe në një hapësirë, madhësia e së cilës është e barabartë me pozitën e shkronjës në tekst.

```
// Programi Tekst3
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     const m=16;
     int i;
```

```
     char T[m]="Lapsi dhe libri";
     for (i=0;i<m;i++)
          cout << setw(i+1)
               << T[i]
               << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati i cili fitohet në ekran do do të duket si në *Fig.3.6*.

Fig.3.6
Rezultati i programit Tekst3

Në program, hapësira e cila rezervohet për shtypje përmes manipulatorit `setw` është e ndryshueshme, ashtu siç ndryshojnë edhe vlerat e variablës `i`.

Komanda cout.width

Për përcaktimin e vendeve në të cilat do të shtypet një numër, mund të shfrytëzohet edhe komanda:

```
cout.width(k);
```

ku `k` është numri i vendeve që do të shfrytëzohen për shtypje. Kështu, p.sh., nëse variablës `x` i është ndarë vlera `-65` dhe brenda një programi ekzekutohen komandat:

```
cout.width(8);
cout << x;
```

vlera në fjalë do të shtypet në 8 vende, duke i lënë 6 zbrazësira para numrit, në këtë mënyrë:

```
□□□□□□-65
```

ku, siç thamë edhe më parë, me simbolin □ janë shënuar zbrazësirat.

Nga kjo që u dha më sipër shihet se efekti i komandës `cout.width` është i njëjtë me atë të manipulatorit `setw`.

Përmes komandës `cout.width` mund të përcaktohet edhe numri i vendeve në të cilat shtypen tekstet.

Shembull

Programi përmes së cilit në ekran shtypet teksti `Koha e bukur`, duke e vendosur komandën `cout.width` për përcaktimin e numrit të vendeve në të cilat shtypet, ashtu që të ndryshohet madhësia e hapësirës shtypëse.

```
// Programi Shtypja2
#include <iostream>
using namespace std;
int main()
{
     int i;
     for (i=1;i<=10;i++)
     {
          cout.width(11+i);
          cout << "Koha e bukur"
               << endl;
     }
return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran do të shtypet teksti i shënuar nën thonjëza, ashtu siç shihet në *Fig.3.7*.

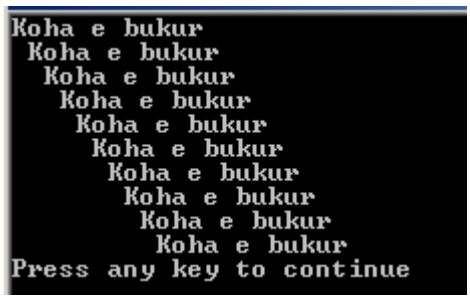


Fig.3.7
Rezultati i programit Shtypja2

Shtypja e këtillë është rezultat i rritjes së hapësirës në të cilën shtypet teksti në fjalë, ashtu siç rritet vlera e variablës `i` në unazë, përmes komandës:

```
cout.width(11+i);
```

Karakteri për tabelim horizontal

Për kapërcim të një hapësire të caktuar vendesh gjatë shtypjes, mund të shfrytëzohet sekuenca dalëse me karakterin për tabelim horizontal `\t`. Sa herë që kompjuteri e takon këtë sekuencë dalëse, i kapërcen 8 vende (kjo është vlerë e nënkuptuar, nëse nuk është ndryshuar nga shfrytëzuesi) në rreshtin që është duke shtypur dhe pastaj e vazhdon shtypjen.

Shembull

Programi përmes së cilit në ekran shtypet fjala `Dita`, me zbrazësira mes shkronjave të veçanta të fjalës, duke e shfrytëzuar manipulatorin për tabelim horizontal `\t`.

```
// Programi Tekst5
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     const m=5;
     int i;
     char F[m]="Dita";
     for (i=0;i<m;i++)
          cout << "\t"
               << F[i];
     cout << endl;
     return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran do të shtypet fjala `Dita` kështu:

```
     D        i        t        a
```

Këtu, me paraqitjen e parë të karakterit për tabelim horizontal `\t` në komandën cout (për `i=0`), kapërcehet fusha prej `8` vendesh (lihen `8` zbrazësira) dhe pastaj shtypet shkronja `D`. Në ciklin e dytë (për `i=1`), me shtypjen e manipulatorit në fjalë jepet komanda për kapërcim të fushës prej 8 vendesh, përfshirë edhe vendin në të cilin është shtypur shkronja `D` dhe pastaj shtypet shkronja e dytë `i`. Në këtë mënyrë procesi

vazhdon derisa të shtypen të gjitha shkronjat e fjalës `Dita`, gjë që në ekran duket si në *Fig.3.8*.

Fig.3.8
Rezultati i programit Tekst5

Karakteri për tabelim horizontal `\t` mund të shfrytëzohet edhe gjatë shtypjes së vlerave numerike.

Shembull

Programi përmes së cilit në ekran shtypen shumat parciale të numrave natyrorë çift mes `2` dhe `n`, në të cilët zbrazësirat mes vlerave të shtypura përcaktohen duke e shfrytëzuar karakterin për tabelim horizontal `\t`.

```
// Programi Shuma1
#include <iostream>
using namespace std;
int main()
{
     const n=9;
     int i,s;
     s=0;
     for (i=2;i<=n;i=i+2)
     {
          s=s+i*i;
          cout << '\t'
               << i
               << '\t'
               << s
               << endl;
     }
return 0;
}
```

Rezultati që fitohet pas ekzekutimit të programit `Shuma1` do të duket si në *Fig.3.9*.

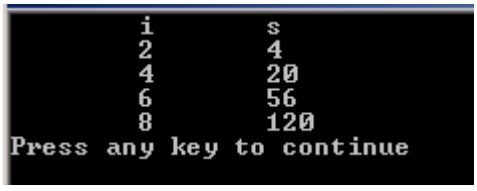


Fig.3.9
Rezultati i programit Shuma1

Siç shihet edhe në *Fig.3.9*, në kolonën e parë janë shtypur numrat natyrorë çift, kurse në kolonën e dytë shumat parciale të tyre. Kështu, p.sh., për numrin natyror `i=4` shuma parciale është:

```
s=2*2 + 4*4
 = 4  +  16 = 20
```

Vendosja e titullit

Gjatë shtypjes së rezultateve në formë të tabelave, në fillim të tabelës duhet të vendoset titulli përkatës. Për këtë qëllim shfrytëzohet komanda `cout`, në vazhdim të së cilës titulli shkruhet nën thonjëza, ashtu që pjesët e tij të përputhen me kolonat përkatëse të tabelës.

Shembull Programi përmes së cilit llogariten katrorët dhe kubet e numrave natyrorë mes `1` dhe `n`.

```
// Programi Titulli1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     const n=5;
     int i;
     cout << " Numri  Katrori  Kubi"
          << endl
          << "------------------------"
          << endl;
     for (i=1;i<=n;i++)
     {
          cout << setw(4)
               << i
               << setw(8)
               << i*i
               << setw(8)
               << i*i*i
               << endl;
     }
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati do të duket si në *Fig.3.10*.

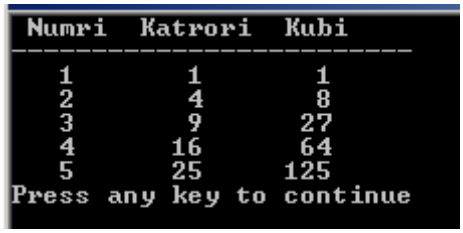


Fig.3.10
Rezultati i programit Titulli1

Siç shihet nga figura e dhënë, në fillim të tabelës janë shkruar dy rreshta. Në rreshtin e parë është titulli i tabelës, kurse rreshti i dytë përmban viza për kufizim të titullit nga vlerat e tabelës.

Rreshti me viza mund të shtypet edhe në fund të tabelës, duke e vendosur një komandë për shtypje para komandës `return 0`, të shkruar në këtë mënyrë:

```
cout << "------------------------"
     << endl;
```

Pas vendosjes së komandës në fjalë, rezultati që fitohet në ekran do të duket si në *Fig.3.11*.

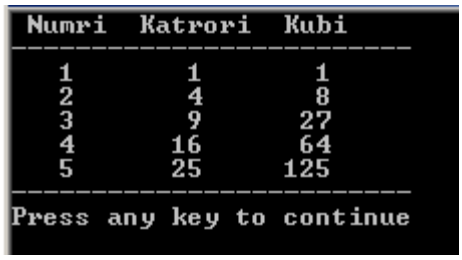


Fig.3.11
Pamja e tabelës pas shtypjes së rreshtit me viza në fund të saj

Mbushja e hapësirës me mostër

Pjesa e zbrazët e hapësirës, e cila është rezervuar për shtypje, përmes komandës:

```
cout fill('s');
```

mund të mbushet me simbolin `s`, i cili zgjidhet sipas dëshirës.

Shembull Programi përmes së cilit në ekran shtypen vlerat numerike të variablave `a` dhe `b`, duke i mbushur hapësirat përkatëse, të rezervuara për shtypje të tyre, me simbolet `*` dhe `^`.

```
// Programi Mostra1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int a=6547,b=-7392;
     cout << "Vlera a=";
     cout.width(8);
     cout.fill('^');
     cout << a
          << endl;
     cout << "Vlera b=";
     cout.width(10);
     cout.fill('*');
     cout << b
          << endl;
return 0;
}
```

Rezultati që fitohet në ekran, pas ekzekutimit të programit të dhënë, duket kështu:

```
Vlera a=^^^^6547
Vlera b=*****-7392
```

Komandat `cout.width` dhe `cout.fill` duhet të vendosen para shtypjes së çdo vlere, përkatësisht nuk vlejnë për të gjitha shtypjet që vijojnë pas tyre.

Gjatë mbushjes me mostër, hapësira për shtypje mund të përcaktohet edhe përmes manipulatorit `setw`.

Shembull Programi përmes së cilit në ekran shtypet vlera numerike e variablës `x`, në hapësirën e cila përcaktohet përmes manipulatorit `setw`, duke e mbushur hapësirën para vlerës që shtypet me simbolin `*`.

```
// Programi Mostra2
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int x=3725;
     cout << "Rezultati x=";
     cout.fill('*');
     cout << setw(8)
          << x
          << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran do të shtypet:

```
Rezultati x=****3725
```

Simboli `s`, i cili shfrytëzohet për mbushje, mund të përcaktohet edhe përmes manipulatorit:

```
setfill('s')
```

i cili shkruhet brenda komandës `cout`. Gjatë shfrytëzimit të këtij manipulatori nuk ka rëndësi si përcaktohet madhësia e hapësirës për shtypje (qoftë përmes komandës `cout.width` ose përmes manipulatorit `setw`).

Shembull

Programi përmes së cilit gjatë shtypjes së vlerës së distancës së kaluar, për mbushjen e hapësirës së zbrazët para numrit me simbolin # shfrytëzohet manipulatori `setfill`.

```
// Programi Mostra3
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     double d=327.56;
     cout << "Distanca e kaluar: ";
     cout << setw(10)
```

```
          << setfill('0')
          << d
          << " Km\n";
return 0;
}
```

Si rezultat, pas ekzekutimit të programit të dhënë, në ekran fitohet:

```
Distanca e kaluar: 0000327.56 Km
```

Mbushjet mund të shfrytëzohen edhe gjatë shtypjes së teksteve.

Shembull

Programi përmes së cilit në ekran shtypet `10` herë teksti `Gjuha C++`, ngjashëm si edhe teksti i shtypur përmes programit `Shtypja2` të dhënë më parë, por këtu zbrazësirat para tekstit plotësohen me pika. Këtu, njëkohësisht shtypet numri i rreshtave `i`, në fillim të çdo rreshti të shtypur.

```
// Programi shtypja3
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int i;
     for (i=1;i<=10;i++)
     {
          cout << left
               << "i="
               << setw(2)
               << setfill(' ')
               << i
               << setw(8+i)
               << right
               << setfill('.')
               << "Gjuha C++"
               << endl;
     }
return 0;
}
```

Rezultati që shtypet në ekran pas ekzekutimit të programit të dhënë do të duket si në *Fig.3.12*.

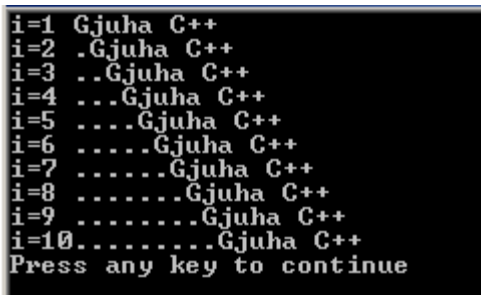


Fig.3.12
Rezultati i programit Shtypja3

Siç mund të shihet edhe në program, për realizimin e shtypjes së vlerave të variablës `i`, si dhe tekstit të zhvendosur `Gjuha C++`, në program janë shfrytëzuar manipulatorët `setw`, `left`, `right` dhe `setfill`.

Shtypja majtas dhe djathtas

Siç shihet edhe nga shembujt e programeve të dhëna më sipër, vlerat që shtypen në hapësirat e rezervuara për shtypje vendosen djathtas. Por, përmes manipulatorit `left` këto vlera mund të shtypen majtas kësaj hapësire.

Shembull

Programi përmes së cilit tregohet përdorimi i manipulatorit `left`, për shtypje të vlerës negative `d` majtas hapësirës së rezervuar për shtypje.

```
// Programi Majt
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     double d=-125.32;
     cout << "Vlera negative: ";
     cout.fill('*');
     cout << setw(10)
          << left
          << d
          << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran duket kështu:

```
Vlera negative: -125.32***
```

Siç shihet edhe nga rezultati, vlera e variablës `d` është shtypur majtas hapësirës prej `10` vendesh, e cila është rezervuar për shtypje. Kurse pjesa e mbetur prej 3 vendesh e kësaj hapësire është mbushur me yje.

Ngjashëm me manipulatorin `left` shfrytëzohet edhe manipulatori `right`, përmes së cilit vlerat shtypen djathtas hapësirës së rezervuar për shtypje.

Shembull

Programi përmes së cilit në ekran shtypen vlerat e variablave `x` dhe `y`, njëra majtas e tjetra djathtas hapësirës së përcaktuar për shtypje.

```
// Programi Shtypja1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     double x=532.74,y=49.6;
     cout.fill('=');
     cout.width(10);
     cout << left
          << x
          << endl;
     cout.width(10);
     cout << right
          << y
          << endl;
return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati që shtypet në ekran duket kështu:

```
532.74====
======49.6
```

Këtu, përmes komandave `cout.width(10)` është përcaktuar që numrat të shtypen në `10` vende, kurse zbrazësirat janë mbushur me simbolin `=`, për çka është shfrytëzuar komanda `cout.fill('=')`. Për t'i shtypur vlerat numerike majtas dhe pastaj djathtas hapësirës së rezervuar për shtypje janë shfrytëzuar manipulatorët `left` dhe `right`, të përfshirë brenda komandave përkatëse `cout`.

Shtypja e parashenjës

Parashenja e numrave me vlerë negative shtypet patjetër. Por, për shtypje të parashenjës së numrave pozitivë duhet të shfrytëzohet manipulatori `showpos`.

Shembull Programi përmes së cilit tregohet përdorimi i manipulatorit `showpos`, për shtypje të parashenjës së vlerave të variablave `x` dhe `y`, të cilat ndryshohen brenda një unaze.

```
// Programi Parashenja
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     double x=-5.3,y;
     cout << "  x       y"
          << endl;
     while (x<=2.5)
     {
          y=x*x;
          cout << showpos
               << "   "
               << x
               << "   "
               << y
               << endl;
          x=x+1;
     }
return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati në ekran duket si në *Fig.3.13*.

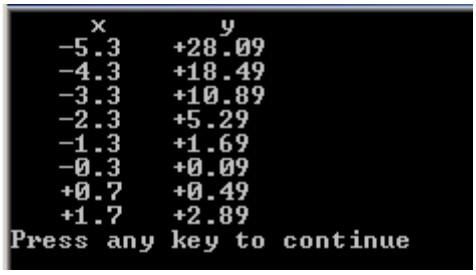


Fig.3.13
Rezultati i programit Parashenja

Siç shihet edhe nga rezultatet, pavarësisht se çfarë parashenje kanë vlerat, me përdorimin e manipulatorit `showpos`, përveç parashenjës së numrave negativë, në ekran shtypen edhe parashenjat e numrave pozitivë.

Efekti i manipulatorit `showpos` eliminohet duke e shfrytëzuar manipulatorin `noshowpos`.

Mundësi të tjera të shtypjes së parashenjës

Parashenja e numrit mund të shtypet në fillim të hapësirës të rezervuar për shtypje, vlera numerike djathtas kësaj hapësire, kurse hapësira para numrit mund të mbushet me mostër të caktuar. Për këtë qëllim shfrytëzohet manipulatori `internal`.

Shembull

Programi përmes së cilit tregohet përdorimi i manipulatorit `internal`, për shtypje të parashenjës negative të numrit `d` majtas hapësirës së rezervuar për shtypje.

```
// Programi Internal
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     double d=-125.32;
     cout << "Vlera negative: ";
     cout.fill('*');
     cout << setw(10)
          << internal
          << d
          << endl;
     return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati që shtypet në ekran do ta ketë këtë pamje:

```
Vlera negative: -***125.32
```

Këtu, për mbushje me yje të pjesës së hapësirës së rezervuar, shfrytëzohet komanda:

```
     cout.fill('*');
```

Në praktikë, manipulatori `internal` mund të shfrytëzohet edhe në rastet kur duam që hapësirën e zbrazët para numrit me parashenjë negative ta plotësojmë me zero. P.sh., për këtë qëllim, në programin e mësipërm, komanda `cout.fill` duhet të shkruhet kështu:

```
cout.fill('0');
```

Rezultati i cili fitohet, nëse ekzekutohet programi `Internal`, pas këtij ndryshimi, është:

```
Vlera negative: -000125.32
```

ku, siç shihet, hapësira e zbrazët para numrit është plotësuar me zero.

Nëse nga programi eliminohet komanda `cout.fill`, rezultati do të shtypet ashtu që parashenja minus shtypet në fillim të hapësirës së rezervuar, kurse numri djathtas kësaj hapësire, kështu:

```
Vlera negative: -□□□125.32
```

duke lënë mes parashenjës dhe numrit një zbrazësirë prej 3 vendesh, të cilat këtu, për t'i dalluar më mirë, janë plotësuar me simbolin □.

Shtypja me precizitet të caktuar

Gjatë shtypjes së numrave, të cilët e kanë edhe pjesën pas pikës dhjetore, (pjesën decimale), kompjuteri nuk i shtyp zerot në fund të kësaj pjese. Gjithashtu, nëse numri nuk ka asnjë shifër në pjesën dhjetore të tij, përkatësisht nëse ato janë shifra zero, kompjuteri nuk i shtyp as zerot dhe as pikën.

Nëse duam që vlera që shtypet të merret me një precizitet të caktuar, përkatësisht ta përcaktojmë se me sa shifra pas pikës dhjetore duam të punojmë, para shtypjes duhet të përdoret komanda:

```
cout.precision(k);
```

ose manipulatori:

```
setprecision(k)
```

ku k është numri i shifrave dhjetore, në të cilin numër përfshihet edhe pika dhjetore.

Gjatë përcaktimit të numrit të shifrave me të cilat duam të punojmë, kompjuteri njëkohësisht e rrumbullakëson shifrën e fundit të numrit, në bazë të rregullave për rrumbullakësim të dhëna në vijim.

- *Nëse shifra e parë që nuk shtypet është 5, ose më e madhe se 5, shifra e fundit që shtypet do të rritet për 1.*
- *Nëse shifra e parë që nuk shtypet është më e vogël se 5, shifra e fundit që shtypet nuk e ndryshon vlerën.*

Rrumbullakësimi nuk i ndryshon vlerat numerike përkatëse në memorien e kompjuterit.

Shembull Programi përmes së cilit llogaritet me precizitet të caktuar vlera numerike e shprehjes y=27/x, nëse variabla x nuk merr vlerë zero dhe kompjuterit i jepet përmes tastierës.

```
// Programi Sakt1
#include <iostream>
using namespace std;
int main()
{
     double x,y;
     cout << "Vlera hyrëse: ";
     cin >> x;
     y=27/x;
     cout << "Vlera y=";
     cout.precision(5);
     cout << y
          << endl;
return 0;
}
```

Nëse pas ekzekutimit të programit, përmes tastierës kompjuterit i jepet vlera 7, rezultati që shtypet në ekran do të jetë:

```
Vlera y=3.8571
```

Ky rezultat shtypet sepse vlera e pjesëtimit 27/7 është 3.8571428, kurse me komandën:

```
cout.precision(5);
```

nga kompjuteri është kërkuar të punohet me precizitet prej 5 vendesh dhjetore, ku si vend numërohet edhe pika.

Me komandën `cout.precision` nuk përcaktohet edhe numri i shifrave pas pikës dhjetore, të cilat shtypen. Kështu, p.sh., nëse gjatë ekzekutimit të programit të mësipërm, përmes tastierës kompjuterit si vlerë hyrëse për variablën x ia japim vlerën 4, rezultati që shtypet në ekran është:

```
Vlera y=6.75
```

vlerë e cila fitohet edhe nëse llogarisim me kalkulator të dorës. Për precizitetin e kërkuar, 3 vendet të tjera të numrit janë 0 dhe prandaj si të panevojshme kompjuteri nuk i paraqet në ekran.

Nëse duam që në ekran të shtypen të gjitha shifrat pas pikës dhjetore, për precizitetin e përcaktuar, duhet ta shfrytëzojmë manipulatorin `showpoint`. Kështu, p.sh., nëse te programi i mësipërm, në fillim të urdhrit të fundit për shtypje e shfrytëzojmë manipulatorin në fjalë:

```
cout << showpoint
     << y
     << endl;
```

rezultati, i cili do të shtypet për vlerën hyrëse 4, do të duket kështu:

```
Vlera y=6.7500
```

Efekti i manipulatorit `showpoint` eliminohet duke e shfrytëzuar manipulatorin `noshowpoint`.

Shembull

Programi përmes së cilit shihet rrumbullakësimi i shifrave dhjetore gjatë shtypjes së vlerës së variablës `x`, me precizitet që ndryshon me ndryshimin e vlerave të variablës `i`.

```
// Programi Sakt2
#include <iostream>
using namespace std;
int main()
{
     int i;
     double x;
     x=5.123456789;
     for (i=1; i<=10; i++)
     {
          cout << "i="
               << i
```

```
                << "\t";
          cout << "x=";
          cout.precision(i);
          cout << x
               << endl;
     }
return 0;
}
```

Rezultatet që shtypen në ekran pas ekzekutimit të programit `Sakt2` do të duken si në *Fig.3.14*.

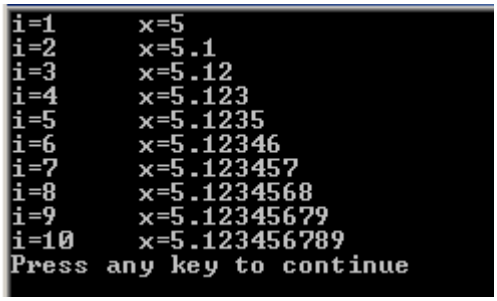


Fig.3.14
Rezultati i programit Sakt2

Në program, përmes komandës:

```
cout.precision(i);
```

përcaktohet numri i shifrave në pjesën dhjetore të vlerës së variablës `x`, të cilat shtypen, duke e përfshirë në atë numër edhe pikën. Kështu, p.sh., për `i=4` pjesa dhjetore që shtypet (`.123`) e përmban pikën dhe `3` shifra.

Nga rezultatet e fituara shihet se prej vlerës `i=5` e deri në `i=9` *shifrat e fundit të numrave janë të rrumbullakësuara*, në bazë të rregullave për rrumbullakësim, të cilat u përmendën më sipër. Kështu, p.sh., për `i=6`, nëse do të shtypej numri x i pa rrumbullakësuar dhe me precizitet 6, në ekran do të kishim:

```
i=6      x=5.12345
```

kurse shifrat dhjetore që nuk shtypen janë `6789`. Meqë shifra e parë që nuk shtypet është `6`, në bazë të rregullave për rrumbullakësim, shifra e fundit dhjetore që shtypet rritet për `1`, përkatësisht shifra `5` bëhet `6`.

Në vend të komandës:

```
cout.precision(i);
```

në programin e dhënë më sipër mund të shfrytëzohet manipulatori:

```
     setprecision(i)
```

ashtu siç shihet në vijim.

```
// Programi Sakt3
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int i;
     double x;
     x=5.123456789;
     for (i=1; i<=10; i++)
     {
          cout << "i="
               << i
               << "\t";
          cout << "x=";
          cout << setprecision(i)
               << x
               << endl;
     }
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran nuk do të dallohet nga ai që është dhënë në *Fig.3.14*.

Shtypja në sisteme të tjera numerike

Vlera e variablave të tipit intexher, përveç në sistemin *decimal* të numrave, mund të shtypen edhe në sistemin *oktal* dhe *heksadecimal*.

Shembull

Programi përmes së cilit vlera e variablës a shtypet edhe në sistemin numerik oktal dhe heksadecimal.

```
// Programi Heksadec
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int a;
```

```
    cout << "Vlera e variablës a: ";
    cin >> a;
    cout << "\nVlera decimale: "
         << dec
         << a;
    cout << "\nVlera oktale: "
         << oct
         << a;
    cout << "\nVlera heksadecimale: "
         << hex
         << a
         << endl;
    return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse për variablën `a` kompjuterit përmes tastierës i jepet vlera `185`, rezultati që shtypet në ekran është:

```
Vlera decimale: 185
Vlera oktale: 271
Vlera heksadecimale: b9
```

Gjatë shtypjes së vlerave në sistemin heksadecimal të numrave, që gjatë shtypjes së shifrave heksadecimale kompjuteri t'i shfrytëzojë shkronjat e mëdha, para shtypjes së vlerave duhet të shfrytëzohet manipulatori `uppercase`. Kështu, p.sh., nëse në programin e mësipërm komanda përkatëse shkruhet kështu:

```
    cout << uppercase
         << "\nVlera heksadecimale: "
         << hex
         << a
         << endl;
```

vlera heksadecimale në rreshtin e fundit të rezultatit të mësipërm është:

```
Vlera heksadecimale: B9
```

Në të kundërtën, për kalim në shtypje me shkronja të vogla të vlerave heksadecimale duhet të shfrytëzohet manipulatori `nouppercase`.

Shtypja decimale dhe eksponenciale

Numrat me pikë dhjetore në ekran zakonisht shtypen në formën e tyre decimale. Por, duke e shfrytëzuar manipulatorin `scientific`, vlerat numerike në ekran mund të paraqiten edhe në formën e tyre eksponenciale. Pas kësaj, për

kthim në shtypjen në formën decimale duhet të shfrytëzohet manipulatori `fixed`.

Leximi dhe shtypja e vektorëve

Vlerat e anëtarëve të vektorit mund të përcaktohen gjatë deklarimit të tipit të tij, përmes deklarimit si konstante, ose ato kompjuterit mund t'i jepen edhe përmes tastierës si vlera hyrëse. Gjithashtu, vlerat e vektorit mund të shtypen, duke e vendosur komandën përkatëse për shtypje brenda një unaze.

Përcaktimi gjatë deklarimit të tipit

Vlerat e anëtarëve të vektorit kompjuterit mund t'i jepen gjatë deklarimit të tipit të tij, duke i shënuar këto vlera brenda kllapave.

Shembull

Programi përmes së cilit kompjuterit i jepen vlerat numerike të vektorit `A(m)` gjatë deklarimit të tipit të tyre dhe pastaj këto vlera shtypen në ekran.

```
// Programi Vektor1a
#include <iostream>
using namespace std;
int main()
{
     const int m=5;
     int i;
     int A[m]={7,2,-4,9,3};
     cout << "Vektori A"
          << "\n";
     for (i=0;i<m;i++)
          cout << A[i]
               << "  ";
     cout << "\n";
return 0;
}
```

Në program, vlerat e anëtarëve të vektorit kompjuterit i janë dhënë gjatë deklarimit të tipit të tij, përmes komandës:

```
int A[m]={7,2,-4,9,3};
```

Numri i anëtarëve të vektorit `m` është deklaruar si konstante e tipit `int`, përmes komandës:

const int m=5;

Përfshirja e tipit gjatë këtij deklarimi nuk është e domosdoshme, por, ashtu siç është dhënë edhe më parë, kjo komandë mund të shkruhet thjesht:

const m=5;

Shtypja e anëtarëve të vektorit

Siç shihet në programin `Vektor1` të dhënë më sipër, anëtarët e vektorit shtypen duke e shfrytëzuar unazën:

```
for (i=0;i<m;i++)
```

përmes së cilës vlerat e indeksit ndryshohen mes vlerës `i=0` dhe `i=m-1`, me hap `1`, gjë që më hollësisht do të shpjegohet në kapitullin vijues.

Komanda për shtypje:

```
cout << A[i]
     << "  ";
```

përmes së cilës shtypet anëtari `A[i]` i vektorit dhe pastaj edhe dy zbrazësira, është vendosur brenda unazës. Rezultati që fitohet në ekran pas ekzekutimit të programit në fjalë do të duket kështu:

```
Vektori A
7  2  -4  9  3
```

ku mes çdo numri të shtypur ka dy zbrazësira. Me komandën për shtypje:

```
cout << "\n";
```

kompjuterit i urdhërohet që pas shtypjes së vektorit, mesazhi përfundimtar, që e përmendëm më parë, të shtypet në rreshtin vijues.

Titulli `Vektori A` është shtypur duke e shfrytëzuar komandën:

```
cout << "Vektori A"
     << "\n";
```

e cila është vendosur menjëherë në fillim, para se të shtypen vlerat e vektorit.

Për futjen e zbrazësirave, pas çdo vlere të shtypur, mund të përdoret edhe karakteri për tabelim horizontal `\t`, përmes së cilit vlerat numerike shtypen në

fusha me nga 8 vende (duke filluar prej vendit të parë në fushë). P.sh., nëse te programi i dhënë më sipër, komanda për shtypje, e cila gjendet brenda unazës, shkruhet:

```
cout << A[i]
     << "\t";
```

anëtarët e vektorit shtypen kështu:

```
Vektori A
7       2       -4      9       3
```

Anëtarët e vektorit mund të shtypen edhe në këtë mënyrë:

```
A={  7  2  -4  9  3  }
```

Për ta realizuar shtypjen e tillë, programi në fjalë duhet të shkruhet si në vijim.

```
// Programi Vektor1b
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i;
    int A[m]={7,2,-4,9,3};
    cout << "A={  ";
    for (i=0;i<m;i++)
        cout << A[i]
             << "  ";
    cout << "}\n";
return 0;
}
```

Anëtarët e vektorit mund të shtypen edhe vertikalisht. Për këtë qëllim, në komandën për shtypje të anëtarëve të vektorit, pas çdo shtypje, duhet të urdhërohet kalimi në një rresht të ri, përmes shtypjes së karakterit për rresht të ri \n (shih programin vijues).

```
// Programi Vektor1c
#include <iostream>
```

```
using namespace std;
int main()
{
    const int m=5;
    int i;
    int A[m]={7,2,-4,9,3};
    cout << "Vektori A" << "\n";
    for (i=0;i<m;i++)
        cout << A[i]
             << "\n";
    return 0;
}
```

Rezultati i cili fitohet në ekran pas ekzekutimit të programit do të duket kështu:

```
Vektori A
7
2
-4
9
3
```

Që rezultati të ketë një formë më të qartë, anëtarët e vektorit mund të shtypen edhe në këtë mënyrë:

```
A[1]=7
A[2]=2
A[3]=-4
A[4]=9
A[5]=3
```

Për shtypje të këtillë, programi në fjalë duhet të shkruhet si në vijim.

```
// Programi Vektor2
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i;
    int A[m]={7,2,-4,9,3};
    for (i=0;i<m;i++)
        cout << "A["
             << i
```

```
            << "]="
            << A[i]
            << "\n";
return 0;
}
```

Hapësira në të cilën shtypen anëtarët e veçantë të vektorit mund të përcaktohet përmes komandës `cout.width`, përdorimi i së cilës është shpjeguar më parë. Kështu, p.sh., programi për shtypje të anëtarëve të vektorit të marrë si shembull në programet e mësipërme, duke e shfrytëzuar këtë komandë, mund të duket si në vijim.

```
// Programi Vektor3
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i;
    int A[m]={7,2,-4,9,3};
    cout << "Vektori A"
         << "\n";
    for (i=0;i<m;i++)
    {
        cout.width(5);
        cout << A[i];
    }
    cout << "\n";
return 0;
}
```

Në program, duke e vendosur komandën `cout.width(5)` brenda unazës për shtypje, është paraparë që çdo anëtar i vektorit të shtypet në `5` vende. Rezultati që fitohet pas ekzekutimit të programit në fjalë do të duket kështu:

```
Vektori A
    7    2   -4    9    3
```

Këtu, vlerat e veçanta janë shkruar duke filluar prej fundit të 5 vendeve të rezervuara për secilin anëtar të vektorit.

Për përcaktimin e vendeve në të cilat shtypen anëtarët e veçantë të vektorit mund të shfrytëzohet edhe manipulatori `setw`, ashtu siç është shpjeguar përdorimi i tij gjatë shtypjes së vlerave të ndryshme.

Përcaktimi përmes deklarimit si konstante

Anëtarët e vektorit mund të deklarohen edhe si konstante, ngjashëm me atë që u shpjegua më sipër, duke e shënuar para fjalën `const`.

Shembull Programi përmes së cilit llogaritet prodhimi p i anëtarëve pozitivë të vektorit të dhënë `A(m)`.

```
// Programi Vektor4
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     const m=5, A[m]={4,-9,-6,2,7};
     int i,p;
     cout << "\nAnëtarët pozitivë\n\n";
     p=1;
     for (i=0;i<m;i++)
          if (A[i]>0)
          {
               cout << setw(10)
                    << A[i]
                    << "\n";
               p=p*A[i];
          }
     cout << "\nProdhimi p="
          << p
          << "\n\n";
return 0;
}
```

Në program, përmes komandës:

```
     const m=5, A[m]={4,-9,-6,2,7};
```

kompjuterit i jepen vlerat e vektorit `A(m)`, si dhe numri i anëtarëve të tij `m`. Kurse për shtypje të anëtarëve të vektorit, brenda unazës, është vendosur komanda:

```
     cout << setw(10)
          << A[i]
          << "\n";
```

Në fund, jashtë unazës është vendosur edhe komanda për shtypjen e prodhimit të anëtarëve pozitivë të vektorit:

```
     cout << "Prodhimi p="
```

```
          << p
          << "\n";
```

Rezultati që fitohet në ekran, pas ekzekutimit të programit `Vektor4`, do të duket si në *Fig.3.15*.

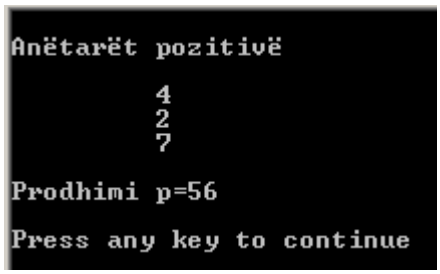


Fig.3.15
Rezultati i programit Vektor4

Përcaktimi përmes leximit

Vlerat e anëtarëve të vektorit kompjuterit mund t'i jepen edhe duke i shkruar përmes tastierës. Për këtë qëllim, komanda për leximin e tyre duhet të vendoset brenda unazës, përmes së cilës ndryshohen indekset e anëtarëve të veçantë.

Shembull Programi përmes së cilit fillimisht lexohen anëtarët e vektorit `A(m)` dhe pastaj ato shtypen në ekran.

```
// Programi Vektor3
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i, A[m];

/* Leximi */

    for (i=0;i<m;i++)
    {
        cout << "Anëtari i "
             << i+1
             << ": ";
        cin >> A[i];
        cout << "\n";
```

```
    }

/* Shtypja */

    cout << "A={  ";
    for (i=0;i<m;i++)
        cout << A[i]
             << "  ";
    cout << "}\n";

    return 0;
}
```

Këtu, brenda unazës për lexim janë vendosur 3 komanda. Me komandën e parë:

```
    cout << "Anëtari i "
         << i+1
         << ": ";
```

fillimisht shtypet teksti i shënuar nën thonjëza:

```
Anëtari i
```

pastaj vlera aktuale e indeksit `i`, i cili është i rritur për `1`, me qëllim që numërimi i anëtarëve të mos fillojë prej zeros, dhe në fund shtypen dy pika dhe një zbrazësirë, për ta fituar, p.sh., mesazhin e parë kështu:

```
Anëtari i 1:
```

Pas komandës `cout`, brenda unazës vjen komanda e dytë:

```
    cin >> A[i];
```

përmes së cilës urdhërohet leximi i vlerës së anëtarit `A[i]`.

Me komandën e tretë në unazë:

```
    cout << "\n";
```

urdhërohet kalimi në rresht të ri, për të vazhduar me shtypjen e mesazhit për leximin e anëtarit të dytë.

Nëse, p.sh., vlerat e anëtarëve të vektorit shtypen në tastierë, duke i zgjedhur lirisht, pas shkruarjes së anëtarit të fundit, në ekran do ta kemi pamjen e dhënë në *Fig.3.16*.

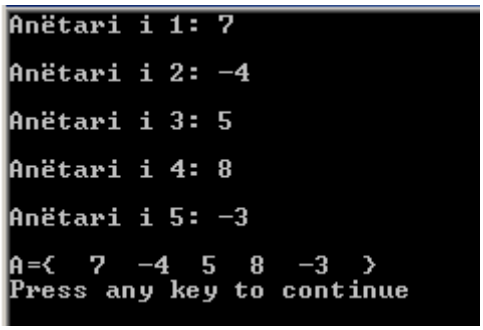


Fig.3.16
Pamja e ekranit pas ekzekutimit të programit dhe shkruarjes në tastierë të vlerave të anëtarëve të vektorit

Këtu, në rreshtin e fundit janë shtypur anëtarët e vektorit përmes komandave të përfshira në pjesën e programit për shtypje.

Leximi dhe shtypja e matricave

Si te vektorët, edhe vlerat e anëtarëve të matricave mund të përcaktohen gjatë deklarimit të tipit të tyre, përmes deklarimit si konstante, ose këto vlera kompjuterit mund t'i jepen si vlera hyrëse përmes tastierës. Për shtypje të anëtarëve të matricave, komandat përkatëse duhet të vendosen brenda dy unazave, me të cilat zgjedhen rreshtat dhe kolonat e tyre.

Plotësisht njëlloj si te matricat veprohet edhe gjatë leximit, ose shtypjes së fushave shumëdimensionale.

Përcaktimi gjatë deklarimit të tipit

Vlerat e anëtarëve të matricës kompjuterit mund t'i jepen gjatë deklarimit të tipit të saj, duke i shënuar këto vlera brenda kllapave.

Shembull Programi përmes së cilit kompjuterit i jepen vlerat numerike të matricës `A(m,n)` gjatë deklarimit të tipit të saj.

```
// Programi Matrica1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     const int m=4, n=5;
     int i,j;
     int A[m][n]={{5,3,-4,8,2},
                  {1,5,2,7,-8},
                  {-3,9,5,2,4},
                  {4,-1,3,6,8}};
```

```
     cout << "Matrica A"
          << "\n";
     for (i=0;i<m;i++)
     {
          for (j=0;j<n;j++)
               cout << setw(7)
                    << A[i][j] ;
          cout << "\n";
     }
return 0;
}
```

Në program, vlerat numerike të matricës janë përcaktuar përmes komandës:

```
int A[m][n]={{5,3,-4,8,2},
             {1,5,2,7,-8},
             {-3,9,5,2,4},
             {4,-1,3,6,8}};
```

Për shtypjen e anëtarëve të matricës shfrytëzohet komanda:

```
cout << setw(7)
     << A[i][j] ;
```

e cila është vendosur brenda dy unazave të realizuara me komandat `for`, përmes së cilave përcaktohen vlerat e dy indekseve (indeksit `i` për zgjedhje të rreshtave të matricës dhe indeksit `j` për zgjedhje të kolonave të saj).

Komanda:

```
cout << "\n";
```

është vendosur në pjesën ku dilet nga unaza e dytë, me qëllim të kalimit në një rresht të ri (pasi të përfundojë shkruarja e rreshtit paraprak të matricës).

Rezultati që shtypet në ekran pas ekzekutimit të programit `Matrica1` do të duket si në *Fig.3.17*.

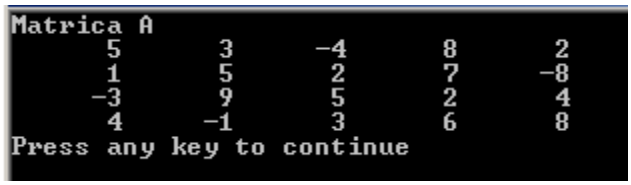


Fig.3.17
Rezultati i programit Matrica1

Gjatë shtypjes së matricave, zbrazësirat mes elementeve të veçanta që shtypen mund të lihen edhe duke e shfrytëzuar karakterin për tabelim horizontal `\t`. Kështu, p.sh., pjesa e programit të mësipërm, përmes së cilës realizohet shtypja e matricës, mund të duket si në vijim.

```
for (i=0;i<m;i++)
{
     for (j=0;j<n;j++)
          cout << A[i][j]
               << "\t";
     cout << "\n";
}
```

Rezultati që fitohet pas ekzekutimit të programit me modifikimin në fjalë do të duket si në *Fig.3.18*.

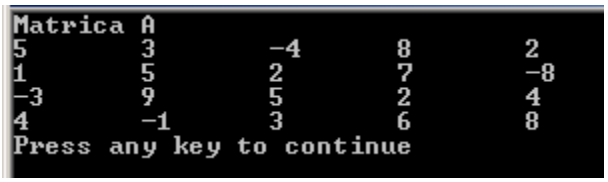


Fig.3.18
Rezultati që fitohet, nëse gjatë shtypjes shfrytëzohet karakteri për tabelim horizontal

Këtu, anëtarët e veçantë të matricës janë shtypur në hapësira me nga `8` vende, duke e filluar shkruarjen e tyre prej fillimit të hapësirave përkatëse.

Përcaktimi përmes deklarimit si konstante

Anëtarët e matricës mund të deklarohen edhe si konstante, ngjashëm me atë që u shpjegua për vektorët.

Shembull

Programi përmes së cilit gjendet numri i anëtarëve negativë `k` të matricës `A(m,n)`, e cila deklarohet në fillim të programit si konstante.

```
// Programi Matrica2
#include <iostream>
#include <iomanip>
using namespace std;
int main()
```

```
{
     const int m=4, n=5;
     const A[m][n]={{5,3,-4,8,2},
                    {1,5,2,7,-8},
                    {-3,9,5,2,4},
                    {4,-1,3,6,8}};
     int i,j,k=0;
     cout << "\n       Matrica A\n"
          << endl;
     for (i=0;i<m;i++)
     {
          for (j=0;j<n;j++)
          {
               if (A[i][j]<0)
                    k=k+1;
               cout << setw(4)
                    << A[i][j];
          }
          cout << endl;
     }
     cout << "\nNumri i anëtarëve negativë k="
          << k
          << "\n\n";
     return 0;
}
```

Në program, përmes deklarimeve `const`:

```
const int m=4, n=5;
const A[m][n]={{5,3,-4,8,2},
               {1,5,2,7,-8},
               {-3,9,5,2,4},
               {4,-1,3,6,8}};
```

kompjuterit i jepen vlerat e dimensioneve të matricës dhe të anëtarëve të saj.

Për numërimin e anëtarëve negativë të matricës, brenda unazave, janë vendosur komandat:

```
if (A[i][j]<0)
     k=k+1;
```

Kurse komandat për shtypje të matricës janë të njëjta me ato që u shfrytëzuan edhe te programi `Matrica1`. Por, për kalim në rresht të ri, në vend të karakterit `\n` këtu shfrytëzohet manipulatori `endl`, me efekt plotësisht të njëjtë.

Rezultati që fitohet në ekran pas ekzekutimit të programit `Matrica2` do të duket si në *Fig.3.19*.

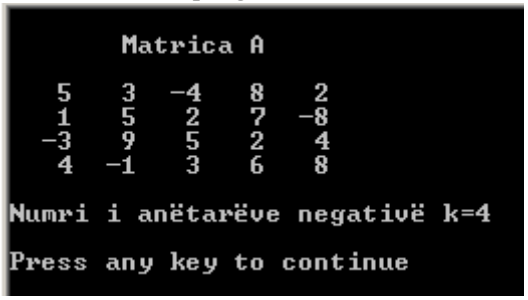


Fig.3.19
Rezultati i programit Matrica2

Përcaktimi përmes leximit

Vlerat e anëtarëve të matricës kompjuterit mund t'i jepen edhe duke i shkruar përmes tastierës, plotësisht njëlloj si edhe gjatë leximit të anëtarëve të vektorit. Por, këtu duhet pasur më shumë kujdes, sepse kemi të bëjmë me dy indekse, njëri për rreshtat dhe tjetri për kolonat.

Leximi dhe shtypja e teksteve

Gjatë punës me kompjuter, shpesh përpunohen edhe tekste. Komandat për leximin dhe shtypjen e teksteve janë të ngjashme me ato që u përmendën gjatë leximit dhe shtypjes së vlerave numerike.

Shembull Programi përmes së cilit lexohet dhe shtypet teksti të cilin kompjuterit ia japim si vlerë hyrëse përmes tastierës.

```
// Programi teksti
#include <iostream>
using namespace std;
int main()
{
     char a[15];
     cout << "Teksti hyrës: ";
     cin >> a ;
     cout << "\nTeksti i lexuar: "
          << a
          << "\n";
     return 0;
}
```

Nëse gjatë ekzekutimit të programit, pas mesazhit:

```
Teksti që lexohet:
```

përmes tastierës shkruhet fjala:

```
Universiteti
```

kompjuteri si rezultat në ekran do të shtypë:

```
Teksti i lexuar: Universiteti
```

Gjatë leximit të zakonshëm, nëse teksti që ia japim kompjuterit si vlerë hyrëse përmes tastierës përmban zbrazësira, kompjuteri e lexon vetëm pjesën e tekstit deri te zbrazësira e parë, duke e injoruar plotësisht pjesën tjetër të tekstit.

Nëse gjatë ekzekutimit të programit të mësipërm, kompjuterit si vlerë hyrëse i jepet fjalia:

```
Koha e bukur
```

si rezultat do të shtypet:

```
Teksti i lexuar: Koha
```

sepse kompjuteri e ndërpret leximin me paraqitjen e zbrazësirës së parë në tekstin hyrës.

Përcaktimi i gjatësisë së tekstit që lexohet

Duke e zgjedhur gjërësinë e fushës ku do të shkruhet teksti, përcaktohet gjatësia e tekstit që lexohet. Për këtë qëllim shfrytëzohet komanda:

```
cin.width(k);
```

me të cilën kompjuteri njoftohet që ta lexojë vetëm tekstin me `k-1` karaktere, duke e kapërcyer pa lexuar pjesën e tekstit që mund të jetë shkruar përmes tastierës.

Shembull

Programi përmes së cilit lexohen vetëm `4` karaktere të tekstit `a`, që kompjuterit i jepet përmes tastierës, i cili është deklaruar me gjatësi prej `10` karakteresh.

```
// Programi Shtypja6
#include <iostream>
using namespace std;
```

```
int main()
{
     char a[10];
     cout << "Teksti hyrës: ";
     cin.width(5);
     cin >> a ;
     cout << "\nTeksti i lexuar: "
          << a
          << "\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe nëse pas mesazhit:

```
Teksti hyrës:
```

si vlerë hyrëse përmes tastierës shkruhet fjala `Prishtina`, rezultati në ekran do të shtypet kështu:

```
Teksti i lexuar: Pris
```

Shkurtimi i fjalës `Prishtina` në `4` shkronja është rezultat i përdorimit të komandës `cin.width(5)`, përmes së cilës leximi kufizohet në `5` karaktere, por kompjuteri prej karaktereve të shkruara përmes tastierës i merr vetëm `4`, sepse si karakter të `5` ia shton *karakterin zero* (`'\0'`).

Në program, te komanda për shtypje:

```
cout << "\nTeksti i lexuar: "
     << a
     << "\n";
```

karakteri për kalim në rresht të ri `\n` paraqitet në fillim dhe në fund të saj. Në këtë mënyrë, kompjuterit i urdhërojmë që, para se ta shtypë fjalinë `Teksti i lexuar`, të kalojë në rresht të ri, pastaj ta shtypë vlerën e variablës `a` (tekstin hyrës) dhe përsëri të kalojë në rresht të ri.

Edhe në këtë rast, nëse teksti të cilin kompjuterit ia japim përmes tastierës ka zbrazësira, kompjuteri e lexon vetëm pjesën e tekstit deri te zbrazësira e parë.

Leximi i fjalive

Kompjuterit mund t'i jepen si vlera hyrëse edhe tekste të çfarëdoshme, siç janë, p.sh., fjalitë. Por, leximi nuk mund të bëhet përmes komandës standarde për lexim cin, sepse kompjuteri e merr vetëm pjesën e fjalisë të cilën ia kemi dhënë përmes tastierës deri në paraqitjen e zbrazësirës së parë.

Për leximin e teksteve të cilat përmbajnë edhe zbrazësira, përdoret komanda e veçantë për lexim:

```
cin.get(a,k);
```

ku a është variabla në të cilën vendoset teksti i lexuar, kurse k është gjatësia e tekstit që lexohet, e shprehur në karaktere.

Shembull Programi përmes së cilit tregohet problemi i leximit të fjalisë brenda së cilës ka zbrazësira si dhe versioni i tij me zgjidhje të problemit në fjalë.

```
// Programi Lexo5
#include <iostream>
using namespace std;
int main()
{
     char A[20];
     cout << "Fjalia që lexohet: ";
     cin >> A;
     cout << "Fjalia që u lexua: "
          << A
          << "\n";
     return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse përmes tastierës kompjuterit i jepet fjalia: Koha e bukur, në ekran do ta kemi pamjen:

```
Fjalia që lexohet: Koha e bukur
Fjalia që u lexua: Koha
```

Prej këtu shihet se kompjuteri si rezultat e ka shtypur vetëm fjalën Koha, kurse pjesën tjetër të fjalisë nuk e ka lexuar fare, sepse leximin e ka ndërprerë me paraqitjen e zbrazësirës së parë.

Për ta eliminuar problemin në fjalë, për lexim duhet të shfrytëzohet komanda e përmendur më sipër cin.get, ashtu siç është shkruar në programin vijues.

```
// Programi Lexo6
#include <iostream>
using namespace std;
```

```
int main()
{
     const m=20;
     char A[m];
     cout << "Fjalia që lexohet: ";
     cin.get(A,m);
     cout << "Fjalia që u lexua: "
          << A
          << "\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepet fjalia e përmendur më sipër, në ekran do ta kemi pamjen:

```
Fjalia që lexohet: Koha e bukur
Fjalia që u lexua: Koha e bukur
```

Prej këtu shihet se kompjuteri e ka lexuar dhe pastaj edhe e ka shtypur komplet fjalinë edhe përkundër asaj se brenda saj ka zbrazësira.

Gjatësia e tekstit që lexohet përmes komandës `cin.get`, kufizohet në një karakter më pak se sa që janë rezervuar vende përmes komandës `char`, sepse si karakter të fundit në tekst kompjuteri vetë e shton *karakterin zero* (`'\0'`), ashtu siç e kemi përmendur edhe më parë.

Leximi i komplet rreshtit

Për ta lexuar si vlerë hyrëse tekstin e shkruar përmes tastierës brenda një rreshti, mund të përdoret edhe komanda:

```
cin.getline(a,k);
```

ku `a` është vektori i karaktereve ku vendoset teksti prej `k`-karakteresh i cili lexohet.

Kompjuteri, si ta takojë këtë komandë, e lexon përmbajtjen e rreshtit aktual, pavarësisht nga përmbajtja e tij.

Shembull Programi përmes së cilit tregohet përdorimi i komandës `cin.getline` për leximin e tekstit të shkruar në një rresht.

```
// Programi lexo7
```

```
#include <iostream>
using namespace std;
int main()
{
     const m=20;
     char A[m];
     cout << "Fjalia që lexohet: ";
     cin.getline(A,m);
     cout << "Fjalia që u lexua: "
          << A
          << "\n";
     return 0;
}
```

Nëse ekzekutohet programi dhe si fjali hyrëse përmes tastierës jepet ajo që u përmend në programin paraprak Lexo6, rezultati në ekran do të duket plotësisht njëlloj.

Shtypja e kushtëzuar

Brenda komandës cout mund të përdoret *operatori i kushtëzuar*, përmes së cilit mund të realizohet shtypja e kushtëzuar.

Shembull

Programi përmes së cilit tregohet shtypja e kushtëzuar duke e shfrytëzuar operatorin e kushtëzuar.

```
// Programi Shtypja5
#include <iostream>
using namespace std;
int main()
{
     int x,y;
     cout << "Vlera për x: ";
     cin >> x;
     cout << "Vlera për y: ";
     cin >> y;
     cout << x
          << (x==y ? " është" : " nuk është")
          << " barazi me "
          << y
          << "\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, kompjuterit së pari duhet përmes tastierës t'i jepen vlerat numerike për variablat `x` dhe `y`. Pastaj, si rezultat në ekran shtypet vlera e variablës `x` dhe varësisht nga ajo se a është ose jo `x` i barabartë me `y`, shtypet mesazhi `është`, ose mesazhi `nuk është`. Në fund, shtypja kompletohet me shtypjen së pari të mesazhit `barazi me` si dhe vlerës së variablës `y`. Kështu, p.sh., nëse kompjuterit si vlera hyrëse i jepen vlerat x=5 dhe y=7, mesazhi që shtypet në ekran në formën e tij përfundimtare do të jetë:

```
5 nuk është barazi me 7
```

Kushti i cili shkruhet para pikëpyetjes `?` mund të jetë i çfarëdoshëm, siç mund të jenë të çfarëdoshme edhe ato që shtypen para dhe pas dy pikave.

4

Degëzimet

Komandat, të cilat gjenden brenda një programi, kompjuteri i ekzekuton një nga një, sipas radhës me të cilën ato paraqiten. Por, ekzekutimi i tillë sekuencial i komandave ndërpritet, nëse në program paraqiten komandat për degëzim. Në gjuhën C++, degëzimet realizohen duke i shfrytëzuar komandat `if`, `goto` dhe `switch`.

Komanda për degëzim if

Komanda themelore përmes së cilës realizohen degëzimet e ndryshme në programe është komanda `if`. Gjatë shkruarjes së komandës `if` mund të paraqiten disa raste të degëzimeve: *të zakonshme*, *të përbëra*, *të shumëfishta* dhe *të ndërthurura*.

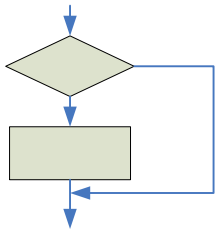
Degëzime të zakonshme

Nëse në degët e komandës `if` ka vetëm nga një komandë thuhet se komanda `if` është e zakonshme.

Me një degë

Pjesa e bllok-diagramit me një degë (shih djathtas), në formë të përgjithshme në program shkruhet kështu:

```
if (k)
    a;
```



ku janë:

`k` - kushti për degëzim.

`a` - komandë e cila ekzekutohet, nëse plotësohet kushti `k`.

Shembull

Programi përmes së cilit krahasohen dy vlera hyrëse `x` dhe `y`. Nëse është `x<y`, llogaritet vlera `z` si shumë e tyre, përndryshe fillimisht merret se `z=-5`.

```
// Programi if1
#include <iostream>
using namespace std;
int main()
{
      double x,y,z;
      cout << "Vlera x=";
      cin >> x;
      cout << "\nVlera y=";
      cin >> y;
      z=-5;
      if (x<y)
            z=x+y;
      cout << "\nRezultati është z=";
      cout << z
           << "\n";
return 0;
}
```

Në program, në fakt llogaritet vlera e funksionit z përmes shprehjes vijuese:

$$z = \begin{cases} -5 & \text{për } x \geq y \\ x + y & \text{për } x < y \end{cases}$$

Nëse ekzekutohet programi, kompjuteri së pari do të kërkojë që përmes tastierës t'ia japim vlerat hyrëse `x` dhe `y`. Kështu, p.sh., nëse pas mesazheve përkatëse në tastierë shtypen vlerat `x=3` dhe `y=7`, meqë plotësohet kushti `x<y`, për variablën `z` do të llogaritet vlera `z=3+7=10`. Si rezultat i veprimeve të përmendura më sipër, në ekran do ta kemi pamjen e dhënë në *Fig.4.1*.

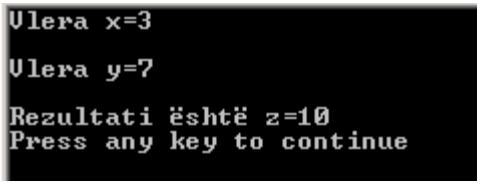


Fig.4.1
Rezultati i programit if1

Komanda për degëzim `if` mund të paraqitet brenda unazës, përmes së cilës përsëritet një pjesë e caktuar e programit.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y = 2x - 3\sum_{\substack{i=1 \\ (i\neq 4)}}^{n+1}(i+2)$$

nëse vlerat e variablave `x` dhe `n` kompjuterit i jepen përmes tastierës.

```
// Programi if8
#include <iostream>
using namespace std;
int main()
{
      double x,s,y;
      int i,n;
      cout << "Vlera x=";
      cin >> x;
      cout << "\nVlera n=";
      cin >> n;
      s=0;
      for (i=1;i<=n+1;i++)
            if (i!=4)
                  s=s+(i+2);
      y=2*x-3*s;
      cout << "\nVlera e funksionit y=";
      cout << y
           << "\n";
return 0;
}
```

Në program, komanda për degëzim:

```
if (i!=4)
```

është vendosur brenda unazës që realizohet me komandën `for`. Përmes saj, nëse vlera e variablës `i` është e ndryshme nga numri `4`, ekzekutohet komanda:

```
s=s+(i+2);
```

përmes së cilës shumës i shtohet anëtari vijues.

Pamja e ekranit pas përfundimit të ekzekutimit të programit `if8` do të duket si në *Fig.4.2*, ku, siç shihet, përmes tastierës kompjuterit i janë dhënë vlerat hyrëse `x=7` dhe `n=5`.

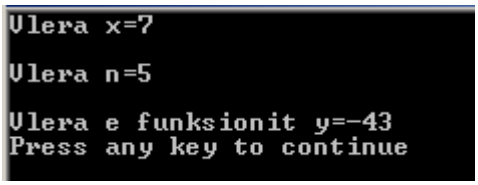


Fig.4.2
Rezultati i programit if8

Rezultati i fituar mund të vërtetohet nëse në shprehjen e dhënë zëvendësohen vlerat hyrëse, kështu:

```
y=2*7-3*{(1+2)+(2+2)+(3+2)+(5+2)}
 =14-3*19
 =-43
```

Kushti i cili përfshihet brenda komandës `if` mund të përmbajë edhe vlera të anëtarëve të vektorëve.

Shembull

Programi përmes së cilit gjendet se sa anëtarë (z) të vektorit `F[m]` janë numra më të mëdhenj se vlera e variablës `x`, e cila kompjuterit i jepet përmes tastierës si numër i plotë.

```
// Programi if14
#include <iostream>
using namespace std;
int main()
{
      const m=8, F[m]={7,3,4,9,-2,5,11,6};
      int i,x,z;
      cout << "\nVlera e variablës x:";
      cin >> x;
      z=0;
      for (i=0;i<m;i++)
            if (F[i]>x)
                  z=z+1;
      cout << "\nMë të mëdhenj se x="
           << x
           << " janë "
           << z
           << " anëtar"
           << "\n\n";
return 0;
}
```

Këtu, komanda `if` është vendosur brenda unazës me të cilën ndryshohen indekset `i` të anëtarëve të vektorit. Gjatë ekzekutimit të kësaj komande, kur gjendet se anëtari i caktuar `F[i]` i vektorit është më i madh se vlera hyrëse `x`, ekzekutohet shprehja:

```
z=z+1;
```

me të cilën numëratori `z` rritet për `1`.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit si vlerë hyrëse për variablën `x` i jepet vlera `5`, rezultati që fitohet në ekran do të duket si në *Fig.4.3*.

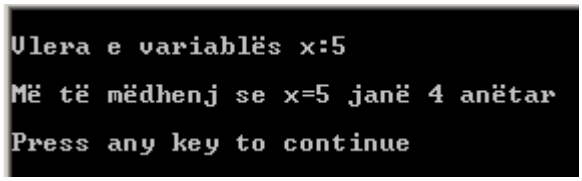


Fig.4.3
Rezultati i programit if14

Në degën e komandës `if` mund të vendoset edhe komanda për shtypje.

Shembull

Programi përmes së cilit shtypen anëtarët negativë të vektorit të dhënë `G[n]`. Njëkohësisht, pranë çdo anëtari duhet të shtypet edhe indeksi përkatës. Anëtarët e vektorit të deklarohen si konstante në fillim të programit.

```
// Programi if15
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const n=8, G[n]={6,-3,-9,5,-8,2,-4,7};
      int i;
      char t[]="------------------";
      cout << "\nAnëtarët  negativë\n\n"
           << "Indeksi   Anëtari"
           << endl
           << t
           << endl;
      for (i=0;i<n;i++)
            if (G[i]<0)
                  cout << setw(4)
                       << i
                       << setw(10)
                       << G[i]
```

```
                        << endl;
        cout << t
             << endl;
return 0;
}
```

Këtu, në degën e komandës `if` është paraparë shtypja e anëtarëve negativë të vektorit dhe të indekseve përkatëse. Rezultati që fitohet në ekran, pas ekzekutimit të programit, do të duket si në *Fig.4.4*.

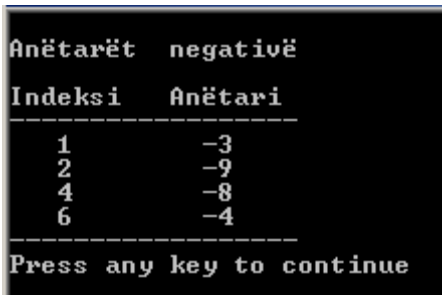
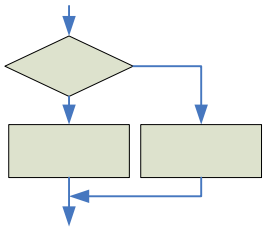


Fig.4.4
Rezultati i programit if15

Me dy degë

Pjesa e bllok-diagramit me dy degë (shih djathtas), në formë të përgjithshme në program shkruhet kështu:

```
if (k)
    a;
else
    b;
```



ku janë:

`k` - kushti për degëzim.
`a` - komandë e cila ekzekutohet, nëse plotësohet kushti `k`.
`b` - komandë e cila ekzekutohet, nëse nuk plotësohet kushti `k`.

Shembull

Programi në të cilin përmes komandës `if` gjendet raporti mes vlerave hyrëse `x` dhe `y`. Pastaj, nëse `x` është më i madh ose barazi me `y`, llogaritet vlera e funksionit `z=2*x+1`, përndryshe kjo vlerë llogaritet me shprehjen `z=x+y`.

```
// Programi if2
#include <iostream>
using namespace std;
```

```
int main()
{
      double x,y,z;
      cout << "Numri i parë x=";
      cin >> x;
      cout << "\nNumri i dytë y=";
      cin >> y;
      if (x>=y)
         z=2*x+1;
      else
         z=x+y;
      cout << "\nVlera e llogaritur është z=";
      cout << z
           << "\n";
return 0;
}
```

Përmes programit të dhënë në fakt llogaritet vlera e funksionit:

$$z=\begin{cases}2x+1 & \text{për } x \geq y \\ x+y & \text{për } x < y\end{cases}$$

Nëse ekzekutohet programi i dhënë dhe përmes tastierës, si vlera hyrëse për variablat `x` dhe `y`, kompjuterit i jepen numrat `7` dhe `3`, në ekran do ta kemi pamjen e dhënë në *Fig.4.5*.

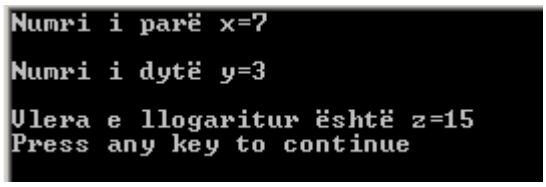


Fig.4.5
Rezultati i programit if2

Në program, meqë plotësohet kushti `x>y`, vlera për `z` është llogaritur përmes shprehjes së parë `z=2*x+1`.

Në degët e komandës për degëzim `if` mund të vendosen komanda për shtypje.

Shembull

Programi në të cilin përmes komandës `if` gjenerohen mesazhet:

```
x barazi me y
```

ose

```
x jobarazi me y
```

varësisht nga raporti mes dy vlerave hyrëse `x` dhe `y`, të cilat kompjuterit i jepen përmes tastierës.

```
// Programi if9
#include <iostream>
using namespace std;
int main()
{
      double x,y;
      cout << "Vlera x=";
      cin >> x;
      cout << "\nVlera y=";
      cin >> y;
      if (x==y)
            cout << "\nx barazi me y";
      else
            cout << "\nx jobarazi me y";
      cout << "\n";
return 0;
}
```

Në program, përmes komandës:

```
if (x==y)
```

gjendet se a është `x=y`. Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepen vlerat hyrëse `5` dhe `7`, në ekran do të paraqitet mesazhi përkatës, ashtu siç shihet në *Fig.4.6*.

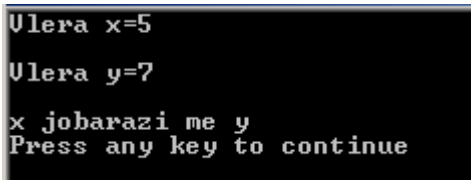


Fig.4.6
Rezultati i programit if9

Në dy degët e komandës if mund të vendosen dy numëratorë të ndryshëm.

Shembull

Programi përmes së cilit numërohen anëtarët pozitivë (`p`) dhe anëtarët negativë (n) të vektorit të dhënë `A(m)`.

```
// Programi if13
#include <iostream>
using namespace std;
int main()
{
      const m=7, A[m]={6,-2,-9,4,7,3,-5};
      int i,p,n;
      p=0;
      n=0;
      for (i=0;i<m;i++)
            if (A[i]<0)
                  n=n+1;
            else
                  p=p+1;
    cout << "\nNumri i anëtarëve pozitivë p="
         << p
         <<"\n";
    cout << "\nNumri i anëtarëve negativë n="
         << n
         <<"\n";
return 0;
}
```

Këtu, komanda:

```
            if (A[i]<0)
                  n=n+1;
            else
                  p=p+1;
```

është vendosur brenda unazës dhe përmes saj çdo anëtar i vektorit `A[i]` testohet se a është numër negativ. Nëse konstatohet se një anëtar i vektorit është numër negativ, përmes shprehjes `n=n+1` rritet për një numri i anëtarëve negativë. Përndryshe, nëse anëtari i vektorit nuk është numër negativ, rritet për një numëratori i numrave pozitivë përmes shprehjes `p=p+1`.

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran është:

```
Numri i anëtarëve pozitivë p=4
Numri i anëtarëve negativë n=3
```

Natyra e komandave në dy degët e komandës `if` mund të jetë e ndryshme.

Shembull

Programi përmes së cilit numërohen anëtarët pozitivë (`p`) dhe shtypen anëtarët negativë të matricës së dhënë `R(m,n)`.

```
// Programi if16
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=4,n=3, R[m][n]={{18,-3,5},
                              {-6,4,-2},
                              {7,-9,14},
                              {5,11,-8}};
      int i,j,p;
      char t[]="------------------------------";
      cout << "\nAnëtarët  n e g a t i v ë\n\n"
           << "     i     j     Anëtari"
           << endl
           << t
           << endl;
      p=0;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                if (R[i][j]<0)
                        cout << setw(6)
                             << i
                             << setw(6)
                             << j
                             << setw(10)
                             << R[i][j]
                             << endl;
                else
                        p=p+1;
      cout << t
           << endl;
      cout << "Numri i anëtarëve pozitivë p="
           << p
           << "\n\n";
return 0;
}
```

Këtu, meqë kemi të bëjmë me matricë, për t'i marrë në shqyrtim të gjithë anëtarët e saj, shfrytëzohen dy unaza: unaza e jashtme me indeksin e parë `i` dhe unaza e brendshme me indeksin e dytë `j`.

Në program, në degën e parë të komandës `if` shtypen indekset dhe vlerat e anëtarëve negativë të matricës, përmes komandës përkatëse `cout`. Kurse nëse anëtari i matricës është pozitiv, ekzekutohet dega e dytë, në të cilën rritet për `1` numëratori i numrave pozitivë (`p=p+1`).

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.4.7*.

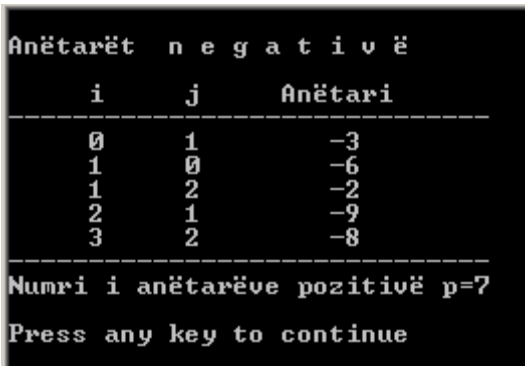


Fig.4.7
Rezultati i programit if16

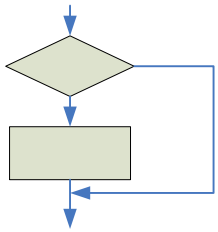
Me komanda të përbëra

Shpesh herë nevojitet që në njërën, ose edhe në të dy degët e komandës `if`, të ketë më shumë komanda, të cilat këtu do t'i quajmë *komanda të përbëra*. Me qëllim të përcaktimit të plotë të komandave të përbëra të një dege, ato përfshihen brenda kllapave.

Me një degë

Nëse komanda `if` përmban vetëm një degë me komanda të përbëra, përmes së cilës programohet pjesa e bllok-diagramit që shihet djathtas, në formë të përgjithshme shkruhet:

```
if (k)
  {
   x;
  }
```



ku janë:

`k` - kushti për degëzim.
`x` - komanda e përbërë, e cila ekzekutohet, nëse plotësohet kushti `k`.

Shembull

Programi përmes së cilit llogariten vlerat numerike të variablave `x` dhe `y`, varësisht nga ajo se a është ose nuk është vlera hyrëse `d` më e madhe se `7`. Nëse plotësohet kushti në fjalë, për llogaritje shfrytëzohen shprehjet:

```
x=3d+2
y=d-4
```

përndryshe vlerat e këtyre dy variablave fillimisht merren zero.

```
// Programi if3
#include <iostream>
using namespace std;
int main()
{
      int d,x,y;
      cout << "Vlera hyrëse d=";
      cin >> d;
      x=0;
      y=0;
      if (d>7)
      {
            x=3*d+2;
            y=d-4;
      }
      cout << "\nVlera x=";
      cout << x;
      cout << "\nVlera y=";
      cout << y
           <<"\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepet vlera hyrëse d=9, për variablat x dhe y do të llogariten vlerat:

```
x=3*9+2=29
y=9-4=5
```

kurse në ekran do ta kemi pamjen e dhënë në *Fig.4.8*.

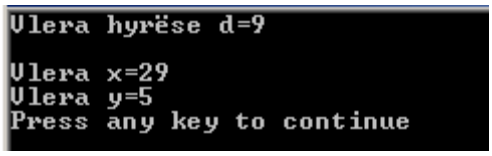


Fig.4.8
Rezultati i programit if3

Në degën e komandës `if` mund të përfshihen edhe llogaritje përmes unazës.

Shembull

Programi përmes së cilit llogaritet vlera numerike e funksionit g përmes shprehjes:

$$g=\begin{cases}3 & \text{për } x=5\\ 2x+4\sum_{i=2}^{n}(x+i) & \text{për } x\neq 5\end{cases}$$

Për variablat x dhe n kompjuterit i jepen vlerat numerike përkatëse përmes tastierës.

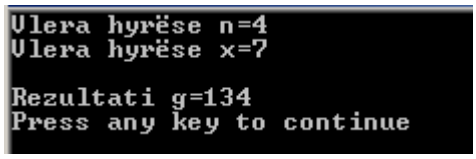
```
// Programi if10
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      float x,s,g;
      cout << "Vlera hyrëse n=";
      cin >> n;
      cout << "Vlera hyrëse x=";
      cin >> x;
      g=3;
      if (x!=5)
         {
             s=0;
             for (i=2;i<=n;i++)
                   s=s+(x+i);
             g=2*x+4*s;
         }
      cout << "\nRezultati g=";
      cout << g
           <<"\n";
return 0;
}
```

Në program, si vlerë fillestare e funksionit është marrë vlera g=3. Por, pastaj, përmes pjesës së programit:

```
      if (x!=5)
         {
             s=0;
             for (i=2;i<=n;i++)
                   s=s+(x+i);
             g=2*x+4*s;
         }
```

nëse x nuk është 5, në vend të vlerës fillestare të funksionit g llogaritet vlera e re e tij, duke e llogaritur fillimisht vlerën e shumës s. Kështu, p.sh., nëse përmes tastierës për variablat n dhe x kompjuterit i jepen vlerat 4 dhe 7, në ekran do ta kemi vlerën e llogaritur për funksionin g, ashtu siç shihet në *Fig.4.9*.

Fig.4.9
Rezultati i programit if10



Dega e përbërë e komandës if njëkohësisht mund të përmbajë komanda për llogaritje dhe shtypje.

Shembull

Programi përmes së cilit numërohen dhe shtypen anëtarët e vektorit B(m), të cilët janë numra më të mëdhenj se numri pozitiv x. Vlera e numrit x kompjuterit t'i jepet si vlerë hyrëse përmes tastierës.

```
// Programi if17
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=7, B[m]={9,-7,2,-5,8,11,3};
      int i,n;
      float x;
      char t[]="----------------";
      cout << "\nVlera e numrit pozitiv x: ";
      cin >> x;
      cout << "\nNumrat më të mëdhenj se "
           << x
           << endl
           << "\n    i    Vlera"
           << endl
           << t
           << endl;
      n=0;
      for (i=0;i<m;i++)
            if (B[i]>x)
                  {
                        n=n+1;
                        cout << setw(5)
                             << i
                             << setw(7)
```

```
                              << B[i]
                              << endl;
                    }
      cout << t
           << "\n\nNumri i anëtarëve më të mëdhenj se "
           << x
           << " është "
           << n
           << "\n\n";
return 0;
}
```

Këtu, në degën e komandës `if` gjenden më shumë komanda, përmes së cilave numërohen anëtarët më të mëdhenj se numri x dhe njëkohësisht shtypen anëtarët e numëruar dhe indekset e tyre.

Pas ekzekutimit të programit, nëse si vlerë hyrëse përmes tastierës kompjuterit i jepet vlera 5, në ekran do ta kemi pamjen e dhënë në *Fig.4.10*.

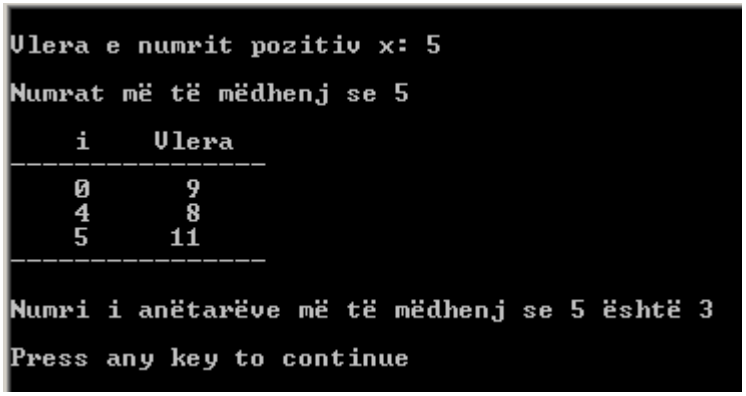


Fig.4.10 Rezultati i programit if17

Këtu, për shtypje të vizave në fillim dhe në fund të tabelës së krijuar është shfrytëzuar variabla `T` e tipit `char`, përmes së cilës ruhen vizat. Pastaj, brenda dy komandave `cout` urdhërohet shtypja e vlerës tekstuale të kësaj variable.

Në program, për ta shtypur fjalinë e cila paraqitet në fund të tabelës:

```
Numri i anëtarëve më të mëdhenj se 5 është 3
```

shfrytëzohet komanda:

```
      cout << T
           << endl
           << "Numri i anëtarëve më të mëdhenj se "
           << x
```

```
        << " është "
        << n
        << endl;
```

Me këtë komandë së pari shtypet rreshti me viza në fund të tabelës (variabla `t`), pastaj pasi të kalohet në rresht të ri përmes manipulatorit `endl`, shtypet teksti:

```
Numri i anëtarëve më të mëdhenj se
```

me një zbrazësirë në fund të tij. Në vazhdim të këtij teksti shtypet vlera e variablës `x`, për ta fituar fjalinë:

```
Numri i anëtarëve më të mëdhenj se 5
```

Në të njëjtin rresht më tutje shtypet fjala `është` me nga një zbrazësirë para dhe pas saj, kështu:

```
Numri i anëtarëve më të mëdhenj se 5 është
```

Në fund të fjalisë së shtypur në këtë mënyrë, shtypet edhe vlera e variablës n, për ta fituar formën përfundimtare të saj:

```
Numri i anëtarëve më të mëdhenj se 5 është 3
```

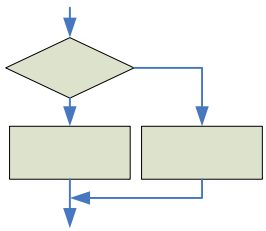
Me dy degë

Kur komanda if ka dy degë, mund të paraqiten tre verzione të saj: komanda e përbërë të paraqitet në degën e parë, në degën e dytë, ose në të dy degët.

Dega e parë e përbërë

Pjesa e bllok-diagramit me dy degë (shih djathtas), njëra prej të cilave përmban më shëmë komanda, në formë të përgjithshme në program shkruhet kështu:

```
if (k)
  {
    x;
  }
else
    b;
```



ku janë:

k - kushti për degëzim.
x - komanda e përbërë e cila ekzekutohen nëse plotësohet kushti k.
b - komandë e cila ekzekutohet nëse nuk plotësohet kushti k.

Shembull

Programi përmes së cilit llogaritet vlera e variablës d në bazë të shprehjeve në dy degët e veçanta të komandës if, e cila ka një komandë të përbërë në degën e parë.

```
// Programi if4
#include <iostream>
using namespace std;
int main()
{
      int k=5,g=9,x,y,h,d;
      h=4*k-g;
      if (h>g)
      {
            x=2*k-1;
            y=3;
            d=3*x+y;
      }
      else
            d=h+2;
     cout << "\nRezultati d=";
     cout << d
          <<"\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, së pari do të llogaritet vlera:

```
h=4*k-g
 =4*5-9=11
```

Pastaj, meqë në komandën if plotësohet kushti h>g, llogariten vlerat:

```
x=2*k-1
 =2*5-1=9
y=3
d=3*x+y
 =3*9+3=30
```

dhe në fund, në ekran shtypet:

```
Rezultati d=30
```

Dega e përbërë e komandës `if` mund të përmbajë edhe unaza për llogaritje të ndryshme.

Shembull

Programi përmes së cilit llogaritet vlera numerike e funksionit `y` përmes shprehjes:

$$y = \begin{cases} x + 2\prod_{i=1}^{n+1}(i+1) & \text{për } x < 3 \\ 2x - 1 & \text{për } x >= 3 \end{cases}$$

Vlerat hyrëse të variablave `x` dhe `n` kompjuterit t'i jepen përmes tastierës.

```
// Programi if11
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      float x,y,p;
      cout << "Vlera hyrëse n=";
      cin >> n;
      cout << "Vlera hyrëse x=";
      cin >> x;
      if (x<3)
            {
               p=1;
               for (i=1;i<=n+1;i++)
                   p=p*(i+1);
               y=x+2*p;
            }
      else
            y=2*x-1;
      cout << "\nRezultati y=";
      cout << y
           <<"\n";
return 0;
}
```

Nëse ekzekutohet programi dhe përmes tastierës, për variablat hyrëse n dhe x, kompjuterit ia japim vlerat 2 dhe 1.5, në ekran do ta kemi pamjen e dhënë në *Fig.4.11*.

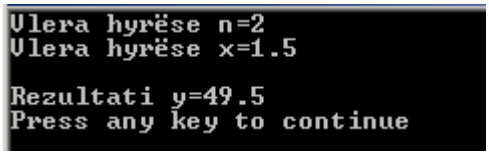


Fig.4.11
Rezultati i programit if11

Te programi if11, vlera e variablës n nevojitet vetëm te dega e parë e komandës if, përkatësisht vetëm nëse është x<3. Për këtë arsye, komandën për lexim të vlerës së kësaj variable mund ta vendosim në degën përkatëse, ashtu siç shihet në verzionin if11a të programit, i cili është dhënë në vijim.

```
// Programi if11a
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      float x,y,p;
      cout << "Vlera hyrëse x=";
      cin >> x;
      if (x<3)
            {
                  cout << "Vlera hyrëse n=";
                  cin >> n;
                  p=1;
                for (i=1;i<=n+1;i++)
                      p=p*(i+1);
                y=x+2*p;
            }
      else
            y=2*x-1;
      cout << "\nRezultati y=";
      cout << y
           <<"\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës si vlerë hyrëse për variablën x ia japim vlerën 5, meqë nuk është x<3, kompjuteri nuk do ta kërkojë vlerën hyrëse të variablës n. Në këtë rast, vlera e funksionit y do të llogaritet me shprehjen y=2*x-1, e cila gjendet në degën e dytë të komandës if dhe në ekran do ta ketë pamjen e dhënë në *Fig.4.12*.

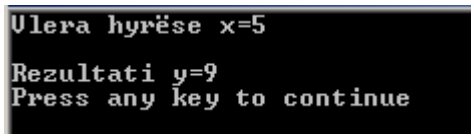


Fig.4.12
Rezultati i programit if11a, për x>3

Por, nëse pas ekzekutimit të programit në fjalë, si vlerë hyrëse për variablën `x` jepet një vlerë më e vogël se `3`, kompjuteri do ta kërkojë edhe vlerën e variablës `n`. Kjo është si rezultat i kalimit në degën e parë të komandës `if`, ku edhe gjendet komanda për leximin e vlerës së variablës `n`. Pas përfundimit të ekzekutimit të programit, në këtë rast do ta kemi pamjen e dhënë në *Fig.4.13*, gjë që është e ngjashme me pamjen e dhënë në *Fig.4.11*.

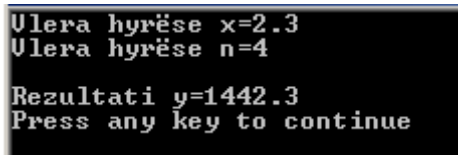


Fig.4.13
Rezultati i programit if11a, për x<3

Në degën e përbërë të komandës `if` mund të paraqitet edhe komandë tjetër për degëzim.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$g = \begin{cases} 3x + z & \text{për } x \geq 5 \\ x - 1 & \text{për } x < 5 \end{cases}$$

ku `z` është anëtari më i madh në vektorin e dhënë `R(n)`. Vlera e variablës `x` kompjuterit t'i jepet përmes tastierës.

```
// Programi if19
#include <iostream>
using namespace std;
int main()
{
      const n=9, R[n]={4,7,2,5,12,9,1,15,3};
      int i,z;
      float x,y;
      cout << "\nVlera hyrëse x=";
      cin >> x;
      if (x>=5)
            {
                  z=R[0];
                  for (i=1;i<n;i++)
                        if (z<R[i])
```

```
                        z=R[i];
                cout << "\nAnëtari më i madh z="
                     << z
                     << endl;
                y=3*x+z;
            }
      else
            y=x-1;
      cout << "\nRezultati y=";
      cout << y
           <<"\n\n";
return 0;
}
```

Në program, në degën e parë të komandës `if` është vendosur procedura e gjetjes së anëtarit më të madh të vektorit `R(n)`. Në fund të kësaj dege, pasi shtypet vlera e gjetur `z`, llogaritet dhe shtypet vlera përkatëse e funksionit `y`. Në *Fig.4.14* është dhënë pamja e ekranit pas ekzekutimit të programit, nëse përmes tastierës, si vlerë hyrëse për variablën `x`, kompjuterit i jepet vlera `7`.

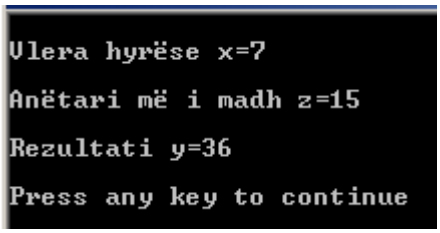
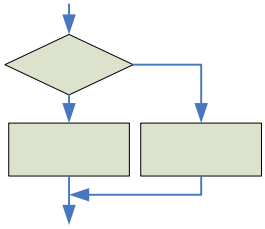


Fig.4.14
Rezultati i programit if19

Dega e dytë e përbërë

Pjesa e bllok-diagramit me dy degë (shih në vijim - djathtas), nëse dega e dytë përmban më shum komanda, në formë të përgjithshme programohet kështu:

```
if (k)
    a;
else
  {
    y;
  }
```



ku janë:

`k` - kushti për degëzim.
`a` - komandë e cila ekzekutohen nëse plotësohet kushti `k`.
`y` - komanda e përbërë e cila ekzekutohet nëse nuk plotësohet kushti `k`.

Shembull Programi përmes së cilit realizohet llogaritja e njëjtë si edhe në programin `if4` të dhënë më sipër, por tek i cili në urdhrin `if` është shkruar kushti i kundërt.

```
// Programi if5
#include <iostream>
using namespace std;
int main()
{
      int k=5,g=9,x,y,h,d;
      h=4*k-g;
      if (h<=g)
            d=h+2;
      else
         {
            x=2*k-1;
            y=3;
            d=3*x+y;
         }
  cout << "\nRezultati d=";
  cout << d
       <<"\n";
return 0;
}
```

Siç shihet në program, dega e dytë e komandës `if` është komandë e përbërë, sepse përmban `3` shprehje aritmetikore. Pas ekzekutimit të tij, në ekran do të fitohet:

```
Rezultati d=30
```

Shembull Versioni i modifikuar i programit `if10` për llogaritjen e vlerës numerike të funksionit `g`, përmes shprehjes:

$$g=\begin{cases}3 & \text{për } x=5\\ 2x+4\sum_{i=2}^{n}(x+i) & \text{për } x\neq 5\end{cases}$$

ashtu që te komanda `if`, për secilën shprehje të funksionit, të paraqitet dega përkatëse.

```
// Programi if12
```

```
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      float x,s,g;
      cout << "Vlera hyrëse n=";
      cin >> n;
      cout << "Vlera hyrëse x=";
      cin >> x;
      if (x==5)
            g=3;
      else
         {
            s=0;
            for (i=2;i<=n;i++)
                  s=s+(x+i);
            g=2*x+4*s;
         }
      cout << "\nRezultati g=";
      cout << g
           <<"\n";
return 0;
}
```

Këtu, në degën e parë të komandës `if` kalohet nëse `x=5`, për t'u llogaritur vlera e funksionit `g` me shprehjen e thjeshtë `g=3`. Kurse në degën e dytë, së pari gjendet vlera e shumës `s` dhe pastaj edhe vlera e funksionit `g` përmes shprehjes `g=2*x+4*s`.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepen vlerat `n=4` e `x=7`, rezultati në ekran do të duket ashtu siç është dhënë në *Fig.4.15*.

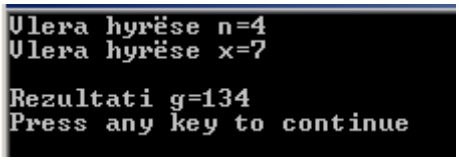
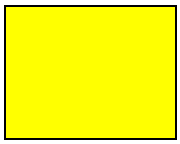


Fig.4.15
Rezultati i programit if12

Në degën e dytë, përveç llogaritjeve, paraqitet edhe komanda për shtypje.

Shembull Programi përmes së cilit gjendet shuma e anëtarëve pozitivë (d)



dhe shuma e katrorëve të anëtarëve negativë (e) të matricës së dhënë A(m,n). Njëkohësisht, gjatë llogaritjes shtypen katrorët e anëtarëve negativë të matricës si dhe indekset përkatëse.

```
// Programi if20
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const int m=3,n=4;
      float A[m][n]={{5,-4,7,2},
                     {-3,-9,6,8},
                     {4,1,-2,-1}};
      int i,j;
      float d,e,k;
      char G[]="-----------------------------------";
      cout << "\nKatrorët e anëtarëve  negativë\n\n"
           << "    i    j     Anëtari   Katrori"
           << endl
           << G
           << endl;
      d=0;
      e=0;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                if (A[i][j]>=0)
                        d=d+A[i][j];
                else
                  {
                        k=A[i][j]*A[i][j];
                        e=e+k;
                        cout << setw(5)
                             << i
                             << setw(5)
                             << j
                             << setw(10)
                             << A[i][j]
                             << setw(10)
                             << k
                             << endl;
                  }
      cout << G
           << endl;
      cout << "Shuma e anëtarëve pozitivë d="
           << d
           << endl;
      cout << "Shuma e katrorëve të anëtarëve negativë e="
```

```
            << e
            << "\n\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.4.16*.

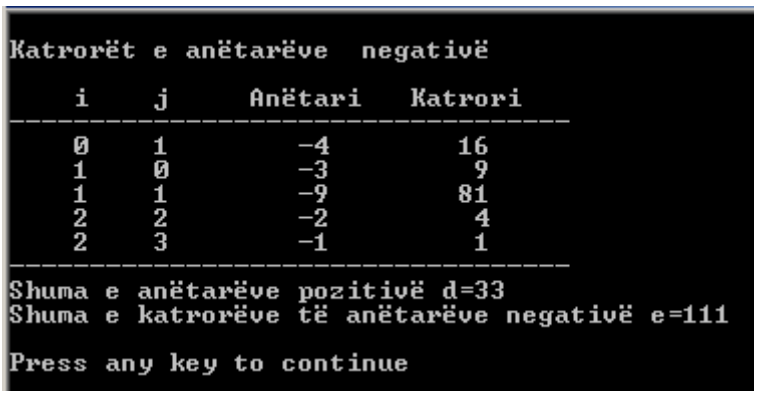
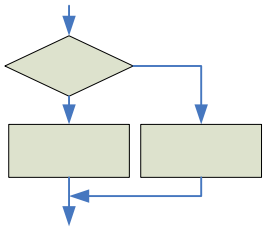


Fig.4.16 Rezultati i programit if20

Të dy degët të përbëra

Pjesa e bllok-diagramit me dy degë (shih në vijim - djathtas), nëse të dy degët përmbajnë më shum komanda, në formë të përgjithshme në program shkruhet kështu:

```
if (k)
  {
    x;
  }
else
  {
    y;
  }
```



ku janë:

k - kushti për degëzim.
x - komanda e përbërë e cila ekzekutohet nëse plotësohet kushti k.
y - komanda e përbërë e cila ekzekutohet nëse nuk plotësohet kushti k.

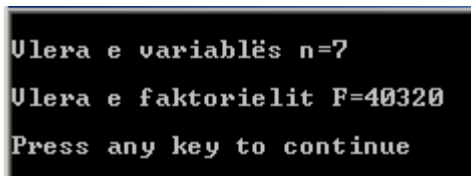
Shembull Programi përmes së cilit llogaritet shuma e numrave natyrorë

mes 2 dhe n, nëse vlera hyrëse n është më e vogël ose barazi me 5, përndryshe llogaritet vlera e faktorielit F=(n+1)!.

```
// Programi if6
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      double S,F;
      cout << "\nVlera e variablës n=";
      cin >> n;
      if (n<=5)
      {
            S=0;
            for (i=2;i<=n;i++)
                S=S+i;
            cout << "Shuma S="
                 << S;
      }
      else
      {
             F=1;
             for (i=1;i<=n+1;i++)
                 F=F*i;
             cout << "\nVlera e faktorielit F="
                  << F;
      }
      cout << "\n\n";
return 0;
}
```

Në program, llogaritjet e faktorielit dhe të shumës janë përcaktuar duke i vendosur komandat e degëve përkatëse brenda kllapave të mëdha. Nëse ekzekutohet programi dhe përmes tastierës, si vlerë hyrëse për n, kompjuterit i jepet numri 7, me kushtin e shkruar nën komandën if kapërcehet në llogaritjen e faktorielit dhe rezultati që shtypet në ekran do të duket si në *Fig.4.17*.

Fig.4.17
Rezultati i programit if6



Në degët e përbëra të komandës if mund të operohet edhe me vlera të anëtarëve të fushave.

Shembull

Programi përmes së cilit formohet matrica katrore `D(m,m)`. Anëtarët e matricës të llogariten përmes shprehjes:

$$d_{ij} = \begin{cases} i + 2\prod_{k=1}^{j+1}(i+k) & \text{për } i \le j \\ j - 3\sum_{k=1}^{i+2}(j-k) & \text{për } i > j \end{cases}$$

Dimensionet e matricës kompjuterit t'i jepen si konstante në fillim të programit.

```
// Programi if21
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=4,n=3;
      int i,j,k;
      float s,p,D[m][n];
      char T[]="-----------------------";
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                if (i<=j)
                  {
                        p=1;
                        for (k=1;k<=j+1;k++)
                              p=p*(i+k);
                        D[i][j]=i+2*p;
                  }
                else
                  {
                        s=0;
                        for (k=1;k<=i+2;k++)
                              s=s+(j-k);
                        D[i][j]=j-3*s;
                  }
      cout << "Matrica e formuar D"
           << endl;
      cout << T
           << endl;
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
```

```
                cout << setw(7)
                     << D[i][j];
           cout << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati në ekran do të duket si në *Fig.4.18*.

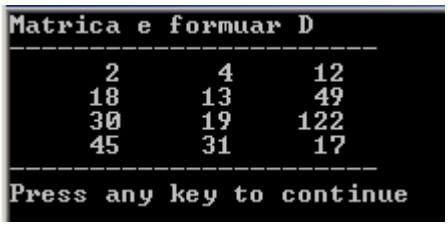


Fig.4.18
Rezultati i programit if21

Degëzime të ndërthurura

Me një komandë `if` realizohet degëzimi në një ose në dy degë. Por, tek algoritmet e ndryshme shpesh paraqiten degëzime të ndërthurura, kur prej një dege mund të ketë degëzime të reja. Për këtë qëllim, në degët e komandave `if` shfrytëzohen edhe komanda të tjera `if`.

Shembull

Programi përmes së cilit llogaritet shuma e anëtarëve të vektorit `A(m)`, të cilët janë më të mëdhenj se numri `3` dhe më të vegjël se numri `9`.

```
// Programi if7
#include <iostream>
using namespace std;
int main()
{
  const int m=8;
  int i,s;
  int A[m]={5,-2,7,4,1,8,10,6};
  s=0;
  for (i=0;i<m;i++)
    if (A[i]>3)
       if (A[i]<9)
           s=s+A[i];
```

```
  cout << "Shuma s="
       << s
       << "\n";
  return 0;
}
```

Nga programi i dhënë shihet se pas komandës së parë `if`, nëse plotësohet kushti `A[i]>3`, vjen komanda e dytë `if` për testim lidhur me kushtin e dytë `A[i]<9`. Pastaj, nëse plotësohen të dy kushtet, shumës i shtohet anëtari përkatës i vektorit. Në fund, rezultati që shtypet në ekran është:

```
Shuma s=30
```

Vlera e shumës është fituar me mbledhjen e anëtarëve të vektorit, të cilët i plotësojnë të dy kushtet, përkatësisht:

```
s=5+7+4+8+6=30
```

Pjesa e programit me komandat if mund të shkruhet edhe duke shfrytëzuar vetëm një komandë if, kështu:

```
s=0;
for (i=0;i<m;i++)
      if ((A[i]>3) && (A[i]<9))
         s=s+A[i];
cout << "Shuma s="
     << s
     << "\n";
```

Këtu, mes dy pjesëve të komandës `if` është shfrytëzuar operatori `&&`, me të cilin kompjuteri e nënkupton operacionin `DHE`. Kështu, kompjuteri do të kalojë te shprehja e shumës `s=s+A[i]`, vetëm nëse `A[i]>3` dhe `A[i]<9`.

Brenda degës njëkohësisht mund të paraqiten edhe më shumë komanda të tjera `if` për degëzim.

Shembull

Programi përmes së cilit numërohen anëtarët pozitivë (`p`), zero (`z`) dhe anëtarët negativë (`n`) të matricës së dhënë `F(k,m)`.

```
// Programi if22
#include <iostream>
```

```
#include <iomanip>
using namespace std;
int main()
{
      const k=4,m=3, F[k][m]={{7,-4,12},
                              {0,17,-6},
                              {4,-1,-5},
                              {15,8,19}};
      int i,j,n,z,p;
      n=0;
      z=0;
      p=0;
      for (i=0;i<k;i++)
            for (j=0;j<m;j++)
                if (F[i][j]<0)
                        n=n+1;
                  else
                        if (F[i][j]==0)
                              z=z+1;
                        else
                              p=p+1;
      cout << "\nNumri i anëtarëve negativë n="
           << n
           << "\n\n";
      cout << "Numri i anëtarëve me vlerë zero z="
           << z
           << "\n\n";
      cout << "Numri i anëtarëve pozitivë p="
           << p
           << "\n\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.4.19*.

```
Numri i anëtarëve negativë n=4

Numri i anëtarëve me vlerë zero z=1

Numri i anëtarëve pozitivë p=7

Press any key to continue
```

Fig.4.19
Rezultati i programit if22

Degëzimet mund të jenë edhe të shumëfishta, në formë kaskadike.

Shembull Programi përmes së cilit llogaritet vlera e funksionit:

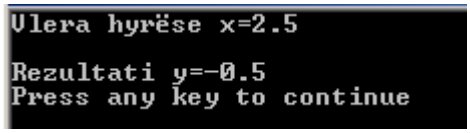
$$y=\begin{cases}5x+4 & \text{për } x<1.5\\ x-3 & \text{për } 1.5\le x<2.7\\ 5 & \text{për } 2.7\le x<4\\ 2x & \text{për } x\ge 4\end{cases}$$

për vlerën e variablës x, e cila kompjuterit i jepet përmes tastierës si vlerë hyrëse.

```
// Programi if23
#include <iostream>
using namespace std;
int main()
{
      float x,y;
      cout << "Vlera hyrëse x=";
      cin >> x;
      if (x<1.5)
            y=5*x+4;
      else
            if (x<2.7)
                  y=x-3;
            else
                  if (x<4)
                        y=5;
                  else
                        y=2*x;
      cout << "\nRezultati y=";
      cout << y
           << endl;
return 0;
}
```

Nëse pas ekzekutimit të programit të dhënë, përmes tastierës për variablën hyrëse x kompjuterit i jepet vlera 2.5, për funksionin y do të fitohet vlera -0.5, ashtu siç shihet në *Fig.4.20*.

Fig.4.20
Rezultati i programit if23



Kjo vlerë e funksionit është llogaritur me shprehjen e dytë `y=x-3`, sepse vlera hyrëse `x=2.5` shtrihet në diapazonën e dytë të kufijve që paraqiten në shprehjen përkatëse.

Degëzimet gjatë komunikimit interaktiv

Komunikimi interaktiv (i drejtpërdrejtë) me kompjuterin realizohet duke shfrytëzuar më shumë komanda `if`. Në vijim do të jepen disa versione të shembullit të një programi elementar, përmes së cilit tregohet shfrytëzimi i komandës `if` gjatë komunikimit interaktiv me kompjuterin.

Shembull Programi përmes së cilit kompjuteri e detekton raportin e dy vlerave hyrëse `a` dhe `b`.

```
// Programi Raporti1
#include <iostream>
using namespace std;
int main()
{
      float a,b;
      cout << "\nSa është vlera e numrit a: ";
      cin >> a;
      cout << "\nSa është vlera e numrit b: ";
      cin >> b;

      if (a<b)
         cout << "\nNumri a është më i vogël se numri b";
      else
         if (a==b)
          cout << "\nNumrat janë të barabartë";
         else
          cout << "\nNumri a është më i madh se numri b";

      cout << "\n\n";
return 0;
}
```

Pasi të ekzekutohet programi i dhënë, në ekran do të paraqitet mesazhi:

```
Vlera e numrit a:
```

i cili shtypet përmes komandës së parë `cout`. Si komandë vijuese në program ekzekutohet komanda `cin` për lexim të vlerës së variablës `a`. Gjatë ekzekutimit të kësaj komande kompjuteri e ndërpret punën duke pritur që në tastierë ta shënojmë vlerën përkatëse hyrëse.

Më tutje përsëritet procedurë e njëjtë me atë paraprake edhe për leximin e vlerës së variablës b, gjatë së cilës fillimisht në ekran paraqitet mesazhi:

```
Vlera e numrit b:
```

Nëse për variablat a dhe b kompjuterit i jepen vlerat hyrëse 5 dhe 8, pas përfundimit të ekzekutimit të programit, në ekran do ta kemi pamjen e dhënë në *Fig.4.21*.

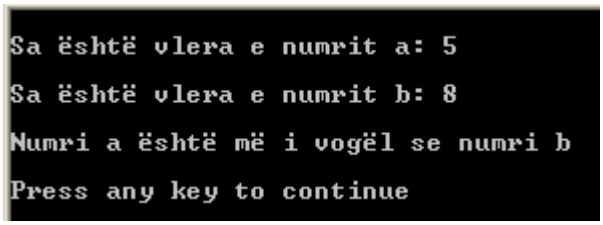


Fig.4.21
Rezultati i programit Raporti1, për a<b

Mesazhi i shtypur në ekran:

```
Numri a është më i vogël se numri b
```

është rezultat i krahasimit përmes komandës së parë `if (a<b)`.

Nëse vlerat hyrëse merren me raport të kundërt me atë që u dha më sipër, përkatësisht nëse, p.sh., përmes tastierës kompjuterit si vlera hyrëse për variablat a dhe b i jepen vlerat 9 dhe 4, në ekran do të shtypet mesazhi:

```
Numri a është më i madh se numri b
```

ashtu siç shihet në *Fig.4.22*.

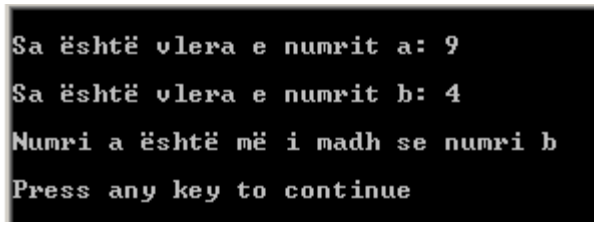


Fig.4.22
Rezultati i programit Raporti1, për a>b

Në këtë rast, fillimisht vlerat hyrëse krahasohen përmes komandës:

```
if (a<b)
```

Por, pasi që nuk është plotësuar kushti a<b, krahasimi vazhdon me komandën e dytë:

```
if (a==b)
```

me të cilën kontrollohet se a janë vlerat hyrëse të barabarta. Duke i pasur parasysh vlerat hyrëse, më tutje ekzekutimi i programit vazhdon me komandën:

```
cout << "\nNumri a është më i madh se numri b";
```

Në fakt, te kjo komandë arrihet pasi nuk plotësohen kushtet në dy komandat `if`, përkatësisht pasi nuk është:

```
a < b
```

dhe

```
a = b
```

Shembull

Programi `Raporti2` si version i përmirësuar i programit `Raporti1`, në të cilin gjatë shtypjes së mesazheve është paraparë që në vend të variablave të përgjithshme `a` dhe `b`, brenda tyre të shtypen vlerat numerike përkatëse.

```
// Programi Raporti2
#include <iostream>
using namespace std;
int main()
{
      float a,b;
      cout << "Vlera e numrit a: ";
      cin >> a;
      cout << "\nVlera e numrit b: ";
      cin >> b;
      if (a<b)
            cout << "\nNumri "
                 << a
                 << " është më i vogël se numri "
                 << b;
      else
            if (a==b)
                  cout << "\nNumrat janë të barabartë";
            else
                  cout << "\nNumri "
                       << a
                       << " është më i madh se numri "
```

```
                        << b;
      cout << endl;
return 0;
}
```

Pas ekzekutimit të programit të dhënë, për vlerat hyrëse 5 dhe 8, në ekran do ta kemi pamjen e dhënë në *Fig.4.23*.

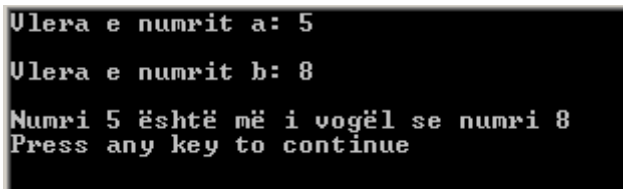


Fig.4.23
Rezultati i programit Raporti2, për a<b

Në program, përmes komandave për shtypje të rezultatit të krahasimit të dy vlerave hyrëse, përveç teksteve shtypen edhe vlerat përkatëse të variablave a dhe b. Gjatë shtypjes është paraparë që, para dhe pas vlerave numerike që shtypen, të ketë edhe nga një zbrazësirë. Për këtë qëllim, zbrazësirat përfshihen nën thonjëzat e teksteve që shtypen.

Kapërcimi pa kusht

Për kapërcim pa kusht, prej një pjese të programit në një pjesë tjetër, ose në fund të tij, shfrytëzohet komanda goto. Kjo komandë, në formën e saj të përgjithshme, shkruhet kështu:

```
goto a;
```

ku a është *labela* e zgjedhur lirisht si identifikator. Përmes kësaj komande kompjuterit i urdhërohet që pa kusht ta vazhdojë ekzekutimin e programit, duke kapërcyer te komanda para së cilës gjendet labela a.

Shembull

Programi Raporti3, i cili e paraqet një version të përmirësuar të programit Raporti2 i dhënë më sipër, ashtu që të përsëritet ekzekutimi i tij në bazë të vlerës hyrëse që kompjuterit i jepet përmes tastierës.

```
// Programi Raporti3
#include <iostream>
using namespace std;
int main()
{
      float a,b;
      char x;
```

```
Fillimi:
      cout << "Vlera e numrit a: ";
      cin >> a;
      cout << "\nVlera e numrit b: ";
      cin >> b;
      if (a<b)
            cout << "\nNumri "
                 << a
                 << " është më i vogël se numri "
                 << b;
      else
            if (a==b)
                  cout << "\nNumrat janë të barabartë";
            else
                  cout << "\nNumri "
                       << a
                       << " është më i madh se numri "
                       << b;

// ------------------- Përsëritja -------------------------

      cout << "\n\nA të përsëritet? P - Po, J - Jo: ";
      cin >> x;
      if (x == 'P')
         {
            cout << "\n\nPërsëritje e ekzekutimit"
                 << "\n-------------------------"
                 << endl;;
            goto Fillimi;
         }
      else
            cout << "\nEkzekutimi përfundoi"
                 << endl;
cout << endl;
return 0;
}
```

Nëse programi i dhënë krahasohet me programin `Raporti2` të dhënë më parë do të shihet se në fund të tij është shtuar pjesa përmes së cilës mundësohet përsëritja e programit. Për këtë qëllim fillimisht është shkruar komanda për leximin e vlerës së variablës `x`, e shoqëruar me komandën paraprake:

```
      cout << "\n\nA të përsëritet? P - Po, J - Jo: ";
```

përmes së cilës në ekran shtypet mesazhi:

`A të përsëritet? P - Po, J - Jo:`
p.sh., ashtu siç shihet në *Fig.4.24*.

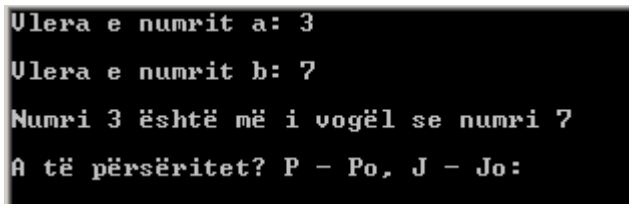


Fig.4.24
Rezultati i programit Raporti3, për a<b

Pastaj, pasi në tastierë të shënohet vlera hyrëse për variablën `x`, ekzekutimi i programit vazhdon me njërën nga dy degët e komandës:

```
if (x=='P')
```

Nëse, p.sh., si vlerë hyrëse në tastierë shtypet shkronja P, do të ekzekutohet dega e parë e kësaj komande dhe si rezultat në ekran fillimisht paraqitet mesazhi:

```
Përsëritje e llogaritjes
-------------------------
```

dhe pastaj përmes komandës `goto Fillimi`, kompjuterit i urdhërohet që ekzekutimin e programit ta vazhdojë duke kaluar në fillim të tij. Me këtë komandë në fakt urdhërohet kapërcimi pa kusht në pjesën e programit pas fjalës `Fillimi:`, e cila është vendosur në fillim të programit. Në këtë mënyrë do të përsëritet ekzekutimi i programit për vlera të tjera të variablave hyrëse `a` dhe `b`, kurse pamja vijuese e ekranit duket si në *Fig.4.25*.

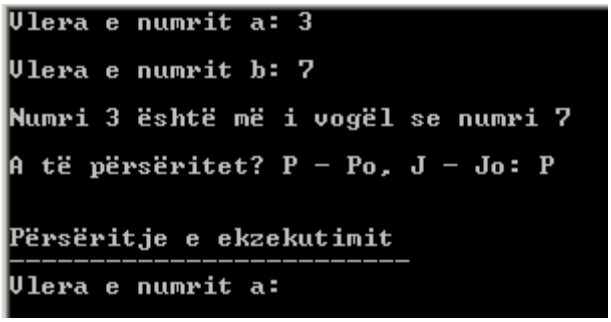


Fig.4.25
Rezultati i programit Raporti3

Më tutje vazhdohet duke ia dhënë kompjuterit përmes tastierës vlerat e variablave `a` dhe `b`. Pastaj, me qëllim të përfundimit të ekzekutimit të programit, nëse pas pyetjes për përsëritje të ekzekutimit, si vlerë hyrëse përmes tastierës shkruhet shkronja `J`, ekzekutimi i programit përfundon dhe në ekran do ta kemi pamjen e dhënë në *Fig.4.26*.

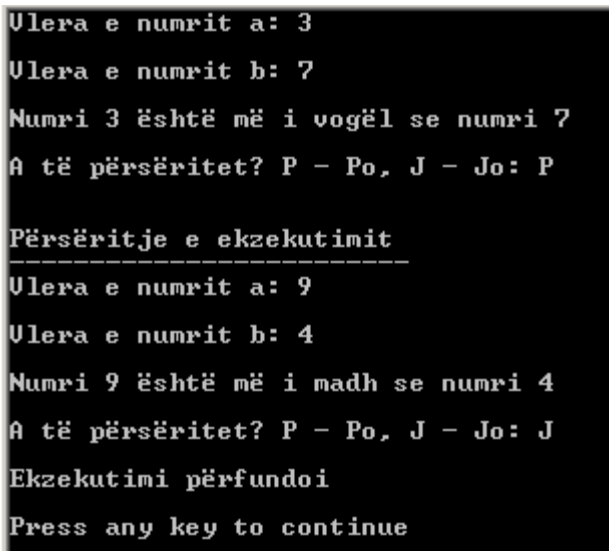


Fig.4.26
Rezultati i programit Raporti3

Ekzekutimi i programit përfundon jo vetëm për vlerën `J` të variablës hyrëse `x`, por edhe për çfarëdo vlerë tjetër e cila nuk është `P`, sepse te komanda:

```
if (x == 'P')
```

pyetet vetëm se a është `x` e barabartë me vlerën `'P'`. Rezultati i kësaj pyetje do të jetë `true`, nëse është `x='P'`, përndryshe në të gjitha rastet e tjera do të jetë `false` (nëse `x` nuk është barazi me `'P'`).

Krijimi i unazës

Komanda e kapërcimit pa kusht `goto` mund të shfrytëzohet me qëllim të krijmit të unazave të zakonshme për përsëritje të pjesëve të caktuara të programit.

Shembull Programi përmes së cilit llogariten dhe shtypen shumat parciale të numrave natyrorë prej `1` deri në `9`.

```
// Programi goto1
#include <iostream>
using namespace std;
int main()
{
      int i=1;
      double s=0;
g:if (i<10)
```

```
     {
          s=s+i;
          cout << "Për i="
               << i;
          cout << " Shuma s="
               << s
               << "\n";
          i++;
          goto g;
     }
return 0;
}
```

Në programin e dhënë, për llogaritjen e shumave parciale, është paraparë të përsëritet pjesa e programit mes komandës `if` me labelën `g` para tij dhe komandës `goto g`, derisa vlera e variablës `i` është më e vogël se `10`. Rezultati që shtypet në ekran pas ekzekutimit të programit në fjalë do të duket si në *Fig.4.27*.

```
Për i=1 Shuma është s=1
Për i=2 Shuma është s=3
Për i=3 Shuma është s=6
Për i=4 Shuma është s=10
Për i=5 Shuma është s=15
Për i=6 Shuma është s=21
Për i=7 Shuma është s=28
Për i=8 Shuma është s=36
Për i=9 Shuma është s=45
Press any key to continue
```

Fig.4.27
Rezultati i programit goto1

Përsëritja e leximit të vlerave hyrëse

Komanda `goto` mund të shfrytëzohet për përsëritjen e leximit të vlerave hyrëse, nëse këto vlera nuk i plotësojnë kushtet e përcaktuara.

Shembull

Programi përmes së cilit gjendet se sa anëtarë pozitivë të matricës `R(m,n)` janë numra më të mëdhenj (`x`) se numri pozitiv `g`, dhe sa janë më të vegjël ose barazi (`y`) me këtë numër. Vlera e numrit `g` kompjuterit i jepet si vlerë hyrëse përmes tastierës. Gjatë kësaj, përmes kontrollës përkatëse pengohet që vlera hyrëse për variablën `g` të jetë numër negativ.

```
// Programi goto2
#include <iostream>
```

```
#include <iomanip>
using namespace std;
int main()
{
      const m=3,n=4,R[m][n]={{-5,6,4,9},
                             {2,-8,5,7},
                             {3,1,8,-6}};
      int i,j,x,y,g;
Lexo:
      cout << "\nVlera e numrit g: ";
      cin >> g;
            if (g<0)
            {
                  cout << "Lejohet vetëm numër pozitiv"
                       << endl;
                  goto Lexo;
            }
      cout << endl;

// -------------- Numërimi i anëtarëve ------------
      x=0;
      y=0;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  if (R[i][j]>0)
                        if (R[i][j]>g)
                              x=x+1;
                        else
                              y=y+1;
      cout << "Numri i anëtarëve më të mëdhenj se "
           << g
           << " është "
           << x
           << endl;
      cout << "Numri i anëtarëve më të vegjël ose barazi me "
           << g
           << " është "
           << y
           << "\n\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse për variablën g përmes tastierës kompjuterit i jepet numri 4, në ekran do ta kemi pamjen e dhënë në *Fig.4.28*.

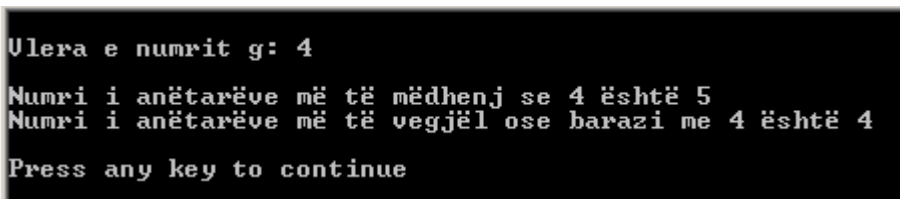


Fig.4.28 Rezultati i programit goto2

Përmes pjesës së programit:

```
Lexo:
      cout << "Vlera e numrit g: ";
      cin >> g;
            if (g<0)
            {
                  cout << "Lejohet vetëm numër pozitiv"
                       << endl;
                  goto Lexo;
     }
```

pengohet që për variablën `g` kompjuterit t'i jepet vlerë hyrëse negative. Kështu, nëse, p.sh., në tastierë shtypet numri -5, në ekran do të paraqitet mesazhi:

```
Lejohet vetëm numër pozitiv
```

dhe pastaj, përmes komandës `goto Lexo`, ekzekutimi i programit kthehet përsëri në pjesën paraprake të tij, për ta përsëritur leximin. Kështu, nëse pas vlerës së gabueshme `-5` kompjuterit i jepet vlera hyrëse `4`, në ekran do ta kemi pamjen përfundimtare të dhënë në *Fig.4.29*.

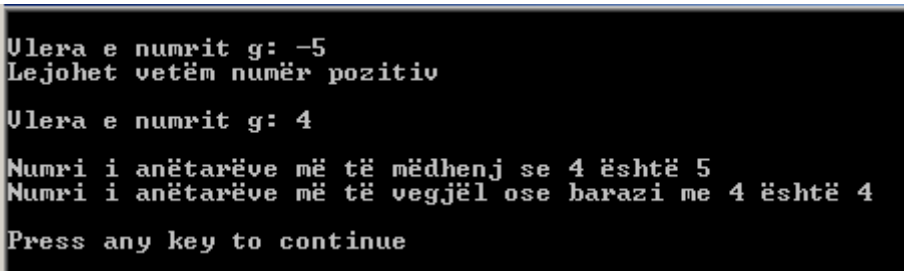
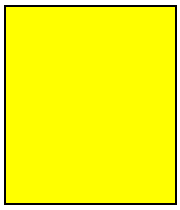


Fig.4.29 Rezultati i programit if18 me mesazhin kur fillimisht si vlerë hyrëse jepet numër negativ

Dalja nga programi

Për ta penguar ekzekutimin e programit për vlera të palogjikshme, pasi përmes tastierës kompjuterit t'i jepen vlerat hyrëse, ato mund të kontrollohen. Pastaj, nëse këto vlera janë të palogjikshme, mund të dilet nga programi duke e shfrytëzuar komandën e kapërcimit pa kusht goto.

Shembull Programi përmes së cilit llogaritet vlera e faktorielit:



F=(2m-3)!

Vlera e variablës `m` kompjuterit t'i jepet përmes tastierës si vlerë hyrëse.

```
// Programi goto3
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int k,m;
      unsigned long F;
      char g[]="-------------------------";
      cout << "\nVlera e variablës m: ";
      cin >> m;
      if (m<=1)
      {
            cout << "\nVlera hyrëse është e palogjikshme"
                 << "\nEkzekutimi i programit ndërpritet"
                 << endl;
            goto Fundi;
      }
      cout << "\n   Numri     Faktorieli\n"
           << g
           << endl;
      F=1;
      for (k=1;k<=(2*m-3);k++)
      {
            F=F*k;
            cout << setw(6)
                 << k
                 << setw(15)
                 << F
                 << endl;
      }
      cout << g
           << endl;
Fundi:
return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse për variablën `m` kompjuterit i jepet vlera `6`, rezultati që shtypet në ekran do të duket si në *Fig.4.30*.

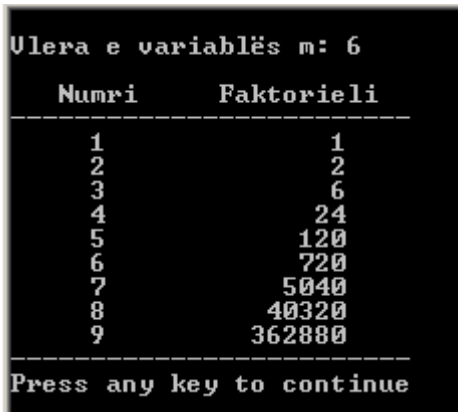


Fig.4.30
Rezultati i programit goto3

Në program, përmes kontrollës:

```
if (m<=1)
```

është paraparë ndërprerja e ekzekutimit të tij, nëse plotësohet kushti i shënuar brenda kllapave. Për ndërprerje të programit shfrytëzohet komanda e kapërcimit pa kusht:

```
goto Fundi;
```

e cila është vendosur në fund të degës së komandës if. Kështu, nëse gjatë ekzekutimit të programit, kompjuterit si vlerë hyrëse për variablën `m` i jepet numri `1`, ekzekutimi i programit do të ndërpritet dhe në ekran do ta kemi pamjen e dhënë në *Fig.4.31*.

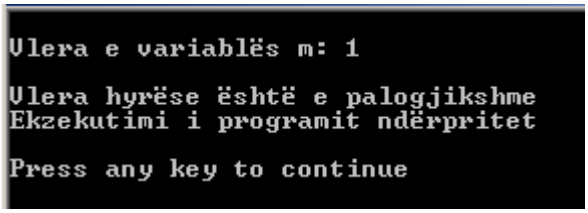


Fig.4.31
Pamja e rezultatit në ekran gjatë ndërprerjes së ekzekutimit të programit goto3

Për ta ndërprerë ekzekutimin e programit, në vend të komandës për kapërcim pa kusht në fund të tij, mund të shfrytëzohet edhe komanda `exit(0)`. Kështu, p.sh., nëse kjo komandë shfrytëzohet te programi `goto3`, i cili u dha më sipër, dega e komandës `if` do të shkruhet:

```
if (m<=1)
{
      cout << "\nVlera hyrëse është e palogjikshme"
```

```
                << "\nEkzekutimi i programit ndërpritet\n"
                << endl;
        exit(0);
}
```

ku, siç shihet, në fund të kësaj dege, në vend të komandës:

```
goto Fundi;
```

është shfrytëzuar komanda:

```
exit(0);
```

Gjatë kësaj, para komandës:

```
return 0;
```

nuk nevojitet më labela `Fundi:`.

Vlera `0` brenda kllapave të komandës `exit` praktikohet të shënohet për të treguar se ekzekutimi i programit është ndërprerë pa gabime.

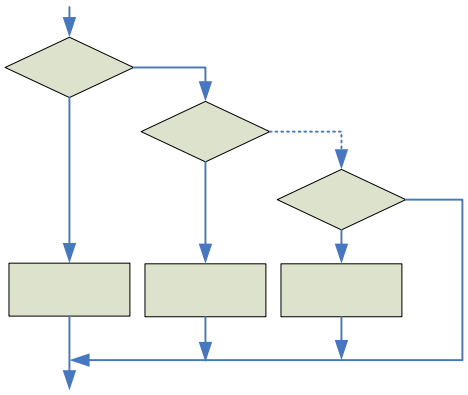
Degëzime me komandën switch

Programet te të cilat njëkohësisht paraqiten më shumë degëzime, thjeshtohen, nëse për degëzim shfrytëzohet komanda `switch`. Ekzistojnë disa versione të kësaj komande, ashtu siç është shpjeguar në vijim.

Versioni bazik i komandës switch

Pjesa e bllok-diagramit i cili shihet në vijim - djathtas, në program mund të paraqitet duke e shfrytëzuar versionin bazik të komandës `switch` kështu:

```
switch (k)
{
case 0: a0;
        break;
case 1: a1;
        break;
.................
```



```
    case n: an;
    }
```

Gjatë shfrytëzimit të kësaj komande, për `k=0` do të ekzekutohet komanda `a0`, për `k=1` - komanda `a1` dhe kështu me radhë, për `k=n` do të ekzekutohet komanda `an`. Nëse `k>n`, vazhdon ekzekutimi i pjesës së programit jashtë kllapave të komandës `switch`. Për ndarje të degëve të veçanta shfrytëzohet komanda `break`, pas së cilës ekzekutimi i programit vazhdon jashtë kllapave të komandës `switch`.

Shembull Programi përmes së cilit llogariten vlerat e funksionit `y` duke shfrytëzuar `4` shprehje të ndryshme për llogaritje varësisht nga `4` vlerat e ndryshme të variablës `i`.

```
// Programi case1
#include <iostream>
using namespace std;
int main()
{
  int i;
  double y;
  for (i=0; i<=3; i++)
  {

    switch(i)
    {
     case 0:    y=3*i+2;
                break;
     case 1:    y=i+6;
                break;
     case 2:    y=-5;
                break;
     case 3:    y=2*i-9;
    }
    cout << "i="
         << i
         <<" y="
         << y
         << "\n";
  }
return 0;
```

```
}
```

Këtu, për `i=0`, përmes komandës `switch(i)` kompjuterit i urdhërohet që ekzekutimin e programit ta vazhdojë te dega:

```
case 0:     y=3*i+2;
            break;
```

Kurse, me komandën `break` në fund të kësaj dege, kompjuterit i urdhërohet që ekzekutimin ta vazhdojë me komandën për shtypje `cout` e cila gjendet menjëherë pas kllapës së mbyllur të trupit të komandës `switch`. Ngjashëm veprohet edhe për vlerat e tjera të variablës `i`.

Rezultati, i cili shtypet në ekran pas ekzekutimit të programit, do të duket si në *Fig.4.32*.

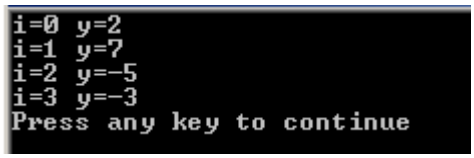


Fig.4.32
Rezultati i programit case1

Nëse nuk vendosen komandat `break`, ekzekutimi i programit vazhdon me të gjitha komandat që gjenden në vijim, pavarësisht nga ajo se ato u takojnë degëve të ndryshme.

Nuk është e domosdoshme që vlerat e variablës, përmes së cilës bëhen degëzimet, të fillojnë me vlerën `0`.

Shembull

Programi përmes së cilit llogariten vlerat e funksionit `y`, duke shfrytëzuar `4` shprehje të ndryshme, nëse kapërcimi bëhet përmes variablës `i`, e cila fillon me vlerën `5`.

```
// Programi case2
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
      int i;
      double x,y;
      for (i=5; i<=8; i++)
      {
            x=2*i;
            switch(i)
            {
                  case 5:    y=sin(x);
```

```
                              break;
                    case 6:   y=sqrt(x);
                              break;
                    case 7:   y=tan(x);
                              break;
                    case 8:   y=exp(x);
               }
               cout << "i="
                    << i
                    <<" y="
                    << y
                    << "\n";
     }
return 0;
}
```

Këtu, për vlerën `i=5`, përmes komandës `switch(i)`, kompjuterit i urdhërohet që ekzekutimin e programit ta vazhdojë me degën:

```
case 5:     y=sin(x);
            break;
```

Pasi të llogaritet vlera e funksionit `y` me shprehjen përkatëse, me komandën `break` pas funksionit, siç u shpjegua edhe më sipër, ekzekutimi i programit vazhdohet menjëherë pas kllapës së mbyllur të trupit të komandës `switch`.

Meqë në programin e dhënë, gjatë llogaritjes janë shfrytëzuar disa funksione matematikore, në fillim të tij, përmes komandës paraprocesorike:

```
#include <math.h>
```

kompjuterit i është urdhëruar që për llogaritje të vlerave të tyre ta shfrytëzojë pakon me funksione të ndryshme matematikore.

Rezultati që fitohet në ekran pas ekzekutimit të programit `case2` duket si në *Fig.4.33*.

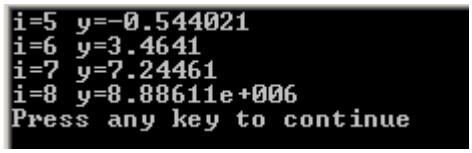


Fig.4.33
Rezultati i programit case2

Kapërcimi për disa vlera në një degë

Kapërcimi në një degë të komandës `case` mund të bëhet edhe për dy ose më shumë vlera të ndryshme të variablës për kapërcim të saj.

Shembull Programi përmes së cilit tregohet kapërcimi në një degë për më shumë vlera të variablës për kapërcim.

```
// Programi case3
#include <iostream>
using namespace std;
int main()
{
  int i;
  double y;
  for (i=0; i<=6; i++)
  {
    switch(i)
    {
     case 0:
     case 5:    y=3*i+2;
                break;
     case 1:    y=i+6;
                break;
     case 2:
     case 6:
     case 4:    y=-5;
                break;
     case 3:    y=2*i-9;
    }
    cout << "i="
         << i
         <<" y="
         << y
         << "\n";
  }
return 0;
}
```

Në program, p.sh., kapërcehet në degën në të cilën gjendet shprehja `y=-5`, për vlerat e variablës `i=2`, `6` dhe `4`. Pas ekzekutimit të programit `case3`, rezultati në ekran do të duket si në *Fig.4.34*.

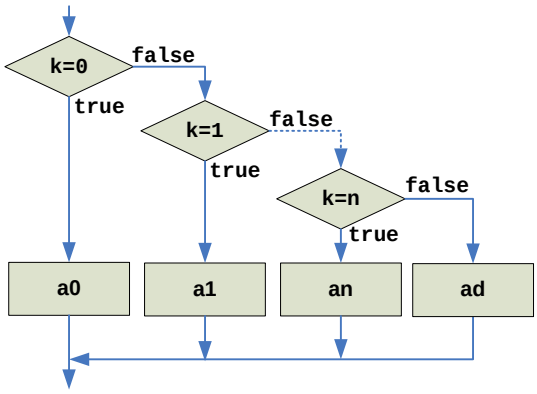
Fig.4.34
Rezultati i programit case3

```
i=0 y=2
i=1 y=7
i=2 y=-5
i=3 y=-3
i=4 y=-5
i=5 y=17
i=6 y=-5
Press any key to continue
```

Kapërcimi për vlerat e çfarëdoshme

Pjesa e bllok-diagramit i cili shihet në vijim - djathtas, në program mund të paraqitet duke e shfrytëzuar versionin e komandën `switch` të kësaj forme:

```
switch (k)
{
  case 0: a0;
          break;
  case 1: a1;
          break;
.................
  case n: an;
          break;
 default: ad;
}
```



Këtu, nëse variabla për degëzim `k` nuk e ka asnjërën prej vlerave: `0, 1, 2, ...`, `n`, ekzekutohet komanda `ad` në degën `default` të komandës `switch`.

Shembull Programi përmes së cilit tregohet kapërcimi në degën `default` të komandës për degëzim `switch`.

```
// Programi case4
#include <iostream>
using namespace std;
int main()
{
  int i;
  double y;
  for (i=0; i<=6; i++)
  {
```

```
    switch(i)
    {
     case 0: y=3*i+2;
             break;
     case 1: y=i+6;
             break;
     case 2: y=-5;
             break;
     case 3: y=2*i-9;
             break;
    default: y=88;
    }
    cout << "i="
         << i
         <<" y="
         << y
         << "\n";
  }
  return 0;
}
```

Nëse ekzekutohet programi i dhënë, për vlera të variablës `i=4`, `5` dhe `6` do të shtypet vlera `y=88`, ashtu siç shihet në *Fig.4.35*, sepse për këto vlera të variablës `i` kapërcehet në degën `default` të komandës `switch`.

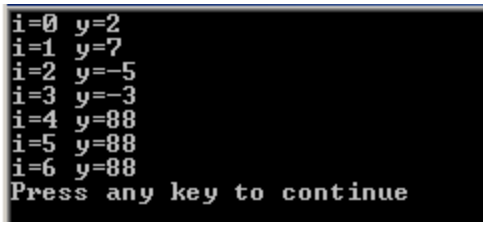


Fig.4.35
Rezultati i programit case4

Kapërcimi përmes vlerave të tipit karakter

Vlerat e variablës për degëzim mund të jenë edhe të tipit karakter.

Shembull

Programi përmes së cilit tregohet degëzimi përmes komandës `switch`, duke e shfrytëzuar për degëzim një variabël të tipit karakter.

```
// Programi case5
#include <iostream>
using namespace std;
int main()
```

```
{
  char h;
  int z;
  cout << "Shtyp P për Po, "
       << "Shtyp J për Jo: ";
  cin >> h;
    z=9;
    switch(h)
    {
     case 'P': z=1;
               break;
     case 'J': z=0;
    }
    cout << "z="
         << z
         << "\n";
return 0;
}
```

Në programin e dhënë është paraparë të kapërcehet në dy degë, për dy vlerat `P` dhe `J` të variablës hyrëse `h`. Nëse nuk zgjedhet asnjëra nga këto dy vlera të variablës `h`, komanda `switch` do të kapërcehet krejtësisht dhe si rezultat do të shtypet `z=9`, e cila është marrë si vlerë fillestare e variablës.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepet vlera hyrëse `P`, si rezultat në ekran do të shtypet vlera `z=1`, ashtu siç shihet në *Fig.4.36*.

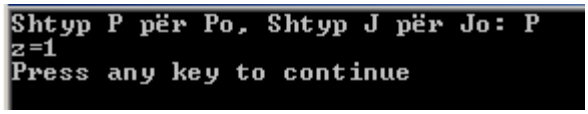


Fig.4.36 Rezultati i programit case5

5

Unazat

Strukturat kontrolluese përmes së cilave përsëriten pjesë të caktuara të programeve quhen *unaza* (ang. loop). Për realizimin e unazave në gjuhën C++ shfrytëzohen komandat `for`, `while` dhe `do-while`. Kurse unazat përkatëse zakonisht emërohen në bazë të këtyre komandave.

Unazat for

Unazat të cilat realizohen duke e shfrytëzuar komandën `for`, shkurt njihen si unaza `for`. Gjatë realizimit të unazave `for` mund të paraqiten disa raste karakteristike, të cilat njihen si: *unaza të zakonshme*, *unaza të përbëra* dhe *unaza të ndërthurura*.

Unazat e zakonshme

Unazat brenda të cilave përfshihet vetëm një komandë mund të quhen unaza të zakonshme. Nëse për realizimin e unazave të zakonshme shfrytëzohet komanda `for`, në formë të përgjithshme ajo shkruhet kështu:

```
for (i=f;i<=p;i=i+h)
    a;
```

ku janë:

`i` - variabla e unazës.
f - vlera fillestare e variablës i.
p - vlera përfundimtare e variablës i.
`h` - hapi me të cilin ndryshohen vlerat e variablës `i`.
a - komanda e cila ekzekutohet brenda unazës.

Përmes kësaj unaze përsëritet ekzekutimi i komandës `a`, për vlera të ndryshme të variablës `i`, prej vlerës fillestare `f`, deri në vlerën përfundimtare p, duke e ndryshuar atë me hapin `h`. Nëse unaza for paraqitet përmes bllok-diagramit, forma e përgjithshme e saj do të duket si në *Fig.5.1*.

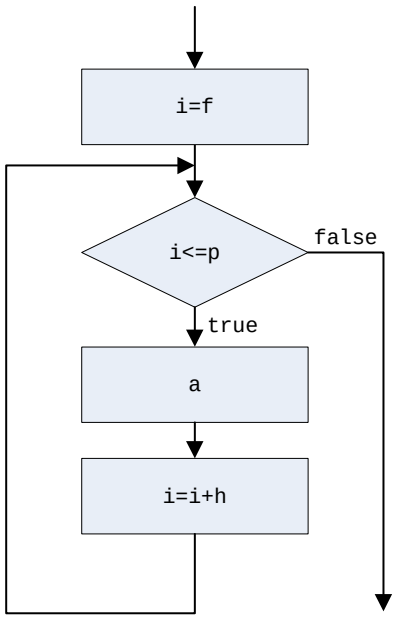


Fig.5.1
Forma e përgjithshme e unazës for e paraqitur përmes bllok-diagramit

Shembull Programi përmes së cilit llogaritet shuma e katrorëve të numrave natyrorë çift mes vlerës 2 dhe n.

```
// Programi for1
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      double s=0;
      cout << "Vlera e variablës n=";
      cin >> n;
      for (i=2;i<=n;i=i+2)
            s=s+i*i;
      cout << "Shuma e katrorëve të numrave çift s="
           << s
           << "\n";
return 0;
}
```

Këtu, brenda komandës `for`, si vlerë fillestare e variablës së unazës `i` është marrë vlera 2, kurse vlera e variablës `i` rritet me hapin 2 përmes shprehjes `i=i+2` në fund të komandës. Nëse ekzekutohet programi i dhënë dhe për

variablën n, përmes tastierës kompjuterit i jepet vlera 7, rezultati që shtypet në ekran është:

```
Shuma e katrorëve të numrave çift s=56
```

gjë që fitohet përmes llogaritjes:

$$s=2^2+4^2+6^2=56$$

Komanda for në programin for1 mund të shkruhet edhe kështu:

```
for (i=2;i<=n;i+=2)
      s=s+i*i;
```

ku, siç shihet, është shfrytëzuar forma e shkurtuar i+=2 e rritjes së variablës së unazës. Në vijim, në rastet kur hapi nuk është 1, do të shfrytëzohet forma e zakonshme e rritjes së vlerës së variablave të unazave, gjë që nuk kërkon një kujdes të veçantë për eliminimin e gabimeve eventuale.

Meqë zakonisht hapi me të cilin ndryshohet vlera e variablës së unazës i është 1, komanda for shkruhet edhe kështu:

```
for (i=f;k;i++)
    a;
```

Shembull Programi përmes së cilit llogaritet shuma e numrave natyrorë mes vlerës 1 dhe n.

```
// Programi for2
#include <iostream>
using namespace std;
int main()
{
      int i,n;
      double s;
      cout << "Vlera e variablës n=";
      cin >> n;
      s=0;
      for (i=1;i<=n;i++)
            s=s+i;
      cout << "Shuma e kërkuar s="
           << s
           << "\n";
```

```
return 0;
}
```

Në programin e dhënë, brenda unazës `for` është përfshirë vetëm komanda:

```
s=s+i;
```

kurse variabla e unazës `i` ndyshohet mes vlerës fillestare `i=1` dhe vlerës përfundimtare `i=n`, e përcaktuar përmes kushtit:

```
i<=n
```

me hapin `1`, meqë në fund të komandës është shënuar shprehja:

```
i++
```

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse për variablën `n`, përmes tastierës kompjuterit i jepet numri `5`, rezultati që shtypet në ekran është:

```
Shuma e kërkuar s=15
```

sepse për `n=5`, kemi:

```
s=1+2+3+4+5=15
```

Që të shihet shpejtësia me të cilën kompjuteri e llogarit shumën në fjalë, nëse ekzekutohet edhe një herë programi dhe si vlerë hyrëse kompjuterit i jepet numri `n=500`, rezultati, i cili shtypet pothuajse momentalisht në ekran, do të jetë `s=125250`.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$z = (m+1)!-2x+1$$

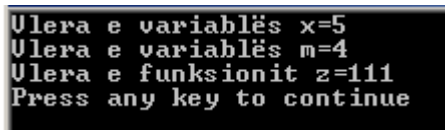
nëse vlerat hyrëse të variablave `m` dhe `x` kompjuterit i jepen përmes tastierës.

```
// Programi for9
#include <iostream>
using namespace std;
int main()
{
```

```
      int i,m;
      double F,x,z;
      cout << "Vlera e variablës x=";
      cin >> x;
      cout << "Vlera e variablës m=";
      cin >> m;
      F=1;
      for (i=1;i<=(m+1);i++)
            F=F*i;
      z=F-2*x+1;
      cout << "Vlera e funksionit z="
           << z
           << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse 5 dhe 4, rezultati në ekran do të duket si në *Fig.5.2*.

Fig.5.2
Rezultati i programit for9



Këtu, siç shihet nga unaza:

```
for (i=1;i<=(m+1);i++)
      F=F*i;
```

variabla e unazës `i` fillon me vlerën `i=1`, dhe rritet për `1` me shprehjen `i++`, duke shkuar deri te vlera kufitare `m+1`, gjë që përcaktohet me kushtin:

```
i<=(m+1)
```

Brenda unazës përsëritet vetëm llogaritja e faktorielit, pasi që brenda saj gjendet vetëm komanda:

```
F=F*i;
```

Variabla e unazës si dhe hapi i saj mund të jenë edhe vlera jo të plota.

Shembull

Programi përmes së cilit shtypet tabela e sipërfaqeve dhe e perimetrave të rrethit me rreze `r`, e cila ndryshohet mes vlerës `1` dhe 5, me hap `0.5`.

```
// Programi for4
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const a=1,b=5;
      double r,pi=3.14159;
      char T[]="--------------------------";
      cout << "   r       s       p"
           << endl
           << T
           << endl;
      cout << fixed
           << setprecision(2);
      for (r=a;r<=b;r=r+0.5)
            cout << setw(6)
                 << r
                 << setw(8)
                 << pi*r*r
                 << setw(8)
                 << 2*pi*r
                 << endl;
      cout << T
           << endl;
return 0;
}
```

Në program, si variabël e unazës `for` është marrë variabla `r` e tipit `double`, gjë që e nënkupton një numër dhjetor. Kjo variabël ndryshohet me hapin 0.5. Rezultatet që shtypen në ekran pas ekzekutimit të programit të dhënë do të duken ashtu siç është dhënë në *Fig.5.3*.

```
   r       s       p
--------------------------
  1.00    3.14    6.28
  1.50    7.07    9.42
  2.00   12.57   12.57
  2.50   19.63   15.71
  3.00   28.27   18.85
  3.50   38.48   21.99
  4.00   50.27   25.13
  4.50   63.62   28.27
  5.00   78.54   31.42
--------------------------
Press any key to continue
```

Fig.5.3
Rezultati i programit for4

Në program, gjatë shtypjes së rezultateve, meqë kemi të bëjmë me numra dhjetorë, shfrytëzohen manipulatorët `fixed`, `setw` dhe `setprecision`.

Përmes manipulatorit `fixed` kompjuteri njoftohet që gjatë shtypjes ta shfrytëzojë formatin fiks, me të cilin nënkuptohet shtypja e numrave në formë dhjetore, përfshirë edhe zerot pas pikës dhjetore.

Siç është shpjeguar edhe më parë, përmes manipulatorit `setw` përcaktohet hapësira (numri i vendeve) në të cilën shtypen vlerat numerike. Kurse, manipulatori `setprecision` shfrytëzohet për përcaktimin e numrit të vendeve pas pikës dhjetore të cilat shtypen. Gjatë kësaj, njëkohësisht rrumbullakësohen vlerat që shtypen, duke i shfrytëzuar rregullat për rrumbullakësim të shpjeguara më parë. Rrumbullakësimi nuk i ndryshon vlerat numerike përkatëse në memorien e kompjuterit.

Komandë e përbërë

Brenda unazës, e cila realizohet përmes komandës `for`, mund të përfshihen edhe më shumë komanda. Në këtë rast, komanda `for` shkruhet kështu:

```
for (i=f; k; i=i+h)
{
     a;
}
```

ku me `a` duhet nënkuptuar grumbullin e komandave brenda trupit të unazës të cilat ekzekutohen.

Shembull Programi përmes së cilit llogariten dhe shtypen faktorielët e numrave natyrorë mes `1` dhe `n`.

```
// Programi for5
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const n=8;
      int i;
      double F;
      char T[]="------------------";
      cout << "    i         F"
           << endl
           << T
           << endl;
      F=1;
      for (i=1;i<=n;i++)
```

```
      {
            F=F*i;
            cout << setw(5)
                 << i
                 << setw(10)
                 << F
                 << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Në programin e dhënë, nën unazën e realizuar me komandën `for`, përfshihet grumbulli i urdhërave që është shkruar brenda kllapave të mëdha. Nëse ekzekutohet ky program, meqë komanda për shtypje është vendosur brenda unazës, në ekran do të shtypen vlerat e faktorielit për çdo vlerë të variablës `i`, ashtu siç shihet në *Fig.5.4*.

```
    i         F
-----------------
    1         1
    2         2
    3         6
    4        24
    5       120
    6       720
    7      5040
    8     40320
-----------------
Press any key to continue
```

Fig.5.4
Rezultati i programit for5

Shembull

Programi përmes së cilit numri i plotë `x` ngritet në fuqinë e numrit të plotë `k`, përkatësisht llogaritet vlera e shprehjes:

$$y = x^k$$

Vlerat e variablave x dhe k kompjuterit i jepen përmes tastierës si vlera hyrëse.

```
// Programi Fuqi1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,k,x;
      double y;
```

```
      char T[]="--------------------";
      cout << "Numri që fuqizohet: ";
      cin >> x;
      cout << "\nVlera e fuqisë k: ";
      cin >> k;
      cout << "\n  Fuqia     Vlera"
           << endl
           << T
           << endl;
      y=1;
      for (i=1;i<=k;i++)
      {
            y=y*x;
            cout << setw(5)
                 << i
                 << setw(10)
                 << y
                 << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse 2 e 10 të variablave x dhe k, rezultati në ekran do të duket si në *Fig.5.5*.

```
Numri që fuqizohet: 2

Vlera e fuqisë k: 10

  Fuqia     Vlera
--------------------
    1         2
    2         4
    3         8
    4        16
    5        32
    6        64
    7       128
    8       256
    9       512
   10      1024
--------------------
Press any key to continue
```

Fig.5.5
Rezultati i programit Fuqi1

Në program, për gjetjen e fuqisë së k-të të numrit x, është shfrytëzuar një unazë me komandën for, përmes së cilës trupi i unazës përsëritet k-herë. Në këtë mënyrë, vlera e fuqisë gjendet përmes shumëzimit të përsëritur. Në praktikë, për llogaritje të fuqizimit shfrytëzohet një funksion i zakonshëm, gjë që do të shpjegohet më vonë.

Brenda unazës mund të përfshihen edhe degëzime të ndryshme, duke e shfrytëzuar komandën `if`.

Shembull Programi përmes së cilit gjendet shuma e katrorëve të numrave natyrorë mes numrave `a` dhe `b`, duke mos i përfshirë në shumë katrorët e numrave `5`, `6` dhe `9`.

```
// Programi for13
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,a,b;
      bool z;
      double s,x;
      char T[]="-----------------------";
      cout << "\nVlera e variablës a: ";
      cin >> a;
      cout << "\nVlera e variablës b: ";
      cin >> b;
      cout << "  Numri      Katrori"
           << endl
           << T
           << endl;
      s=0;
      for (i=a;i<=b;i++)
      {
            z=(i==5) || (i==6) || (i==9);
            if (z==false)
            {
                  x=i*i;
                  s=s+x;
                  cout << setw(5)
                       << i
                       << setw(12)
                       << x
                       << endl;
            }
      }
      cout << T
           << endl;
      cout << "Vlera e shumës s="
           << s
           << endl;
return 0;
}
```

Në program, për ta eliminuar përfshirjen e numrave 5, 6 dhe 9, në llogaritjen e shumave është shfrytëzuar shprehja logjike:

```
z=(i==5) || (i==6) || (i==9);
```

përkatësisht komanda përkatëse për degëzim:

```
if (z==false)
```

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse 1 dhe 15 të variablave a dhe b, rezultati në ekran do të duket si në *Fig.5.6*.

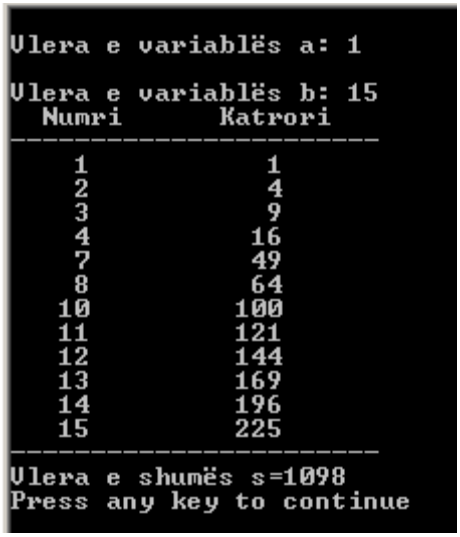


Fig.5.6
Rezultati i programit for13

Nëse përmes tastierës, për variablat a dhe b kompjuterit i jepen, p.sh., vlerat 9 dhe 3, përkatësisht nëse a>b, tërësia e komandave të përfshira brenda kllapave të unazës nuk do të ekzekutohet asnjëherë dhe në ekran do ta kemi pamjen e dhënë në *Fig.5.7*.

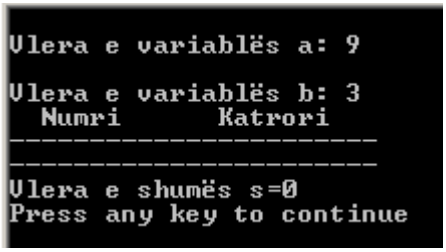


Fig.5.7
Rezultati i programit for13, kur a>b

Përveç degëzimeve, brenda unazës mund të përfshihen edhe llogaritje të ndryshme.

Shembull Programi përmes së cilit gjenden numrat natyrorë mes vlerave `m` dhe `n`, të cilët janë të plotpjesëtueshëm me numrin `r`.

```
// Programi for11
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,m,n,r,x,y;
      char T[]="----------------------";

      cout << "\nVlera e variablës m: ";
      cin >> m;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nVlera e variablës r: ";
      cin >> r;
      cout << "\nTë plotpjesëtueshëm me "
           << r
           << endl
           << "  Numri    Sa herë"
           << endl
           << T
           << endl;

      for (i=m;i<=n;i++)
      {
            x=i/r;
            y=x*r;
            if (y==i)
                  cout << setw(5)
                       << i
                       << setw(10)
                       << x
                       << endl;
      }

      cout << T
           << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse 3, 50 dhe 7 të variablave përkatëse m, n dhe r, rezultati në ekran do të duket si në *Fig.5.8*.

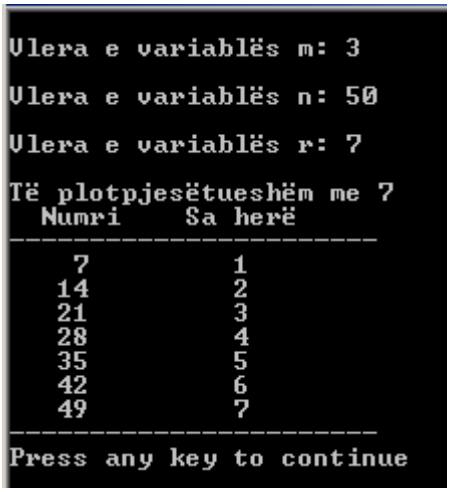


Fig.5.8
Rezultati i programit for11

Nga rezultati i fituar, p.sh., shihet se numri 35 është i plotpjesëtueshëm me 7, sepse pjesëtimi 35:7=5 është pa mbetje.

Në program, përmes shprehjes x=i/r gjendet pjesa e plotë e pjesëtimit të numrit i me r. Kjo arrihet kështu:

- pjesëtimi i/r e jep si rezultat një numër dhjetor.
- vlera e këtij pjesëtimi i shoqërohet variablës x.
- meqë variabla x është e tipit int, në të ruhet vetëm pjesa e plotë e pjesëtimit.

Për ta penguar ekzekutimin e komandave brenda unazës for për vlera të palogjikshme, pasi përmes tastierës kompjuterit t'i jepen vlerat hyrëse, ato mund të kontrollohen.

Shembull Versioni i programit for11 në të cilin është shtuar pjesa përmes së cilës kontrollohen vlerat hyrëse.

```
// Programi for11a
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
```

```
      int i,m,n,r,x,y;
      char T[]="----------------------";
      cout << "\nVlera e variablës m: ";
      cin >> m;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nVlera e variablës r: ";
      cin >> r;

      if ((r==0) || (m>n))
         goto Fundi;

      cout << "\nTë plotpjesëtueshëm me "
           << r
           << endl
           << "  Numri    Sa herë"
           << endl
           << T
           << endl;

      for (i=m;i<=n;i++)
      {
            x=i/r;
            y=x*r;
            if (y==i)
                  cout << setw(5)
                       << i
                       << setw(10)
                       << x
                       << endl;
      }

      cout << T
           << endl;
Fundi:
return 0;
}
```

Këtu, në fillim të programit, menjëherë pas leximit të vlerave për variablat hyrëse, përmes komandës:

```
      if ((r==0) || (m>n))
         goto Fundi;
```

pengohet vazhdimi i ekzekutimit të komandave brenda unazës `for`, sepse hyrja në unazë nuk ka kuptim, nëse kërkohet gjetja e plotpjesëtimeve me zero (`r=0`). Gjithashtu, nuk ka kuptim as rasti i përpilimit të tabelës, nëse `m>n`.

Unaza të ndërthurura

Brenda një unaze mund të përfshihen edhe unaza të tjera, por ashtu që ato të mos priten mes vete.

Shembull Programi përmes së cilit shtypen vlerat e variablës së unazës së brendshme, e cila si vlerë kufitare e ka vlerën e variablës së unazës së jashtme.

```
// Programi p6
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,j;
      for (i=0;i<=9;i++)
      {
            for (j=0;j<=i;j++)
                  cout << setw(2)
                        << j;
            cout << endl;
      }
return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati në ekran do të duket si në *Fig.5.9*.

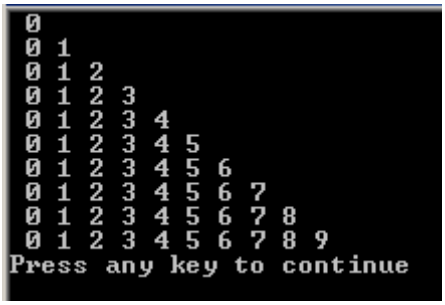


Fig.5.9
Rezultati i programit p6

Në program janë shfrytëzuar dy unaza, me variablat përkatëse `i` dhe `j`. Vlerat e variablës i në unazën e jashtme, e cila është realizuar me komandën:

```
for (i=0;i<=9;i++)
```

ndryshojnë mes vlerave 0 dhe 9, duke u rritur me hapin 1. Në trupin e kësaj unaze janë vendosur komandat që gjenden mes dy kllapave, përfshirë edhe unazën e brendshme:

```
for (j=0;j<=i;j++)
```

Kurse, kjo unazë e përfshin vetëm komandën:

```
cout << setw(2)
     << j;
```

Komanda:

```
cout << endl;
```

nuk përfshihet në unazën e brendshme, por është pjesë e trupit të unazës së jashtme. Përmes saj, kompjuterit i urdhërohet që shtypjen ta vazhdojë në rreshtin vijues, pasi të përfundojë shtypja e vlerave të variablës `j` të unazës së brendshme, për një vlerë të variablës `i` të unazës së jashtme.

Brenda unazave mund të kryhen llogaritje të ndryshme.

Shembull

Programi përmes së cilit përpilohet tabela e vlerave të funksionit:

$$y = \frac{x}{3} - 4\sum_{i=2}^{m+1}(2k + i - 1)$$

për vlera të ndryshme të variablës `k` mes vlerave `0` dhe `n`. Vlerat numerike të variablave `x`, `m` dhe `n` kompjuterit i jepen si vlera hyrëse përmes tastierës.

```
// Programi for15
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
```

```
      int i,k,m,n;
      double s,x,y;
      char T[]="----------------------";
      cout << "Vlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës m: ";
      cin >>m;
      cout << "\nVlera e variablës n: ";
      cin >>n;
      cout << "\n    k          y"
           << endl
           << T
           << endl;

      for (k=0;k<=n;k++)
      {
           s=0;
           for (i=2;i<=(m+1);i++)
                s=s+(2*k+i-1);
           y=x/3-4*s;
           cout << setw(5)
                << k;
           cout.precision(4);
           cout << setw(15)
                << fixed
                << y
                << endl;
      }

      cout << T
           << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket si në *Fig.5.10*.

```
Vlera e variablës x: 7

Vlera e variablës m: 3

Vlera e variablës n: 6

    k          y
----------------------
    0       -21.6667
    1       -45.6667
    2       -69.6667
    3       -93.6667
    4      -117.6667
    5      -141.6667
    6      -165.6667
----------------------
Press any key to continue
```

Fig.5.10

Rezultati i programit for15

Siç shihet në program, unaza e jashtme e realizuar përmes komandës:

```
for (k=0;k<=n;k++)
```

paraqet një unazë të përbërë, sepse brenda trupit të saj janë përfshirë të gjitha komandat që gjenden mes dy kllapave të mëdha. Kurse unaza e brendshme, e realizuar përmes komandës:

```
for (i=2;i<=(m+1);i++)
```

në trupin e saj e përfshin vetëm komandën:

```
s=s+(2*k+i-1);
```

Gjatë ekzekutimit të programit të dhënë, për një vlerë të variablës `k` në unazën e jashtme, gjenerohen të gjitha vlerat e mundshme të variablës `i` në unazën e brendshme. Kështu, p.sh., për vlerat hyrëse të marra më sipër, për çdo vlerë të variablës `k` mes `0` dhe `6`, variabla `i` do t'i marrë vlerat `2`, `3` dhe `4`.

Në një program mund të paraqiten njëkohësisht edhe më shumë unaza të ndërthurura mes vete.

Shembull

Programi përmes së cilit gjenden *numrat e Pitagorës* `x`, `y` dhe `z`, për të cilët vlen barazimi:

$$x^2 + y^2 = z^2$$

```
// Programi for12
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int x,y,z,n,a,b;
      char T[]="--------------------";
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nNumrat e Pitagorës"
           << endl
           << "    x    y    z"
           << endl
           << T
```

```
            << endl;

      for (x=1;x<=n-2;x++)
            for (y=x+1;y<=n-1;y++)
                  for (z=y+1;z<=n;z++)
                  {
                        a=x*x+y*y;
                        b=z*z;
                        if (a==b)
                              cout << setw(5)
                                   << x
                                   << setw(5)
                                   << y
                                   << setw(5)
                                   << z
                                   << endl;
                  }

      cout << T
           << endl;
return 0;
}
```

Në program janë shfrytëzuar 3 unaza `for`, përmes të cilave merren vlera të ndryshme të variablave `x`, `y` dhe `z`, nga grumbulli i numrave natyrorë mes `1` dhe `n`.

Nëse ekzekutohet programi i dhënë, për vlerën hyrëse `n=30`, rezultati që fitohet në ekran do të duket si në *Fig.5.11*.

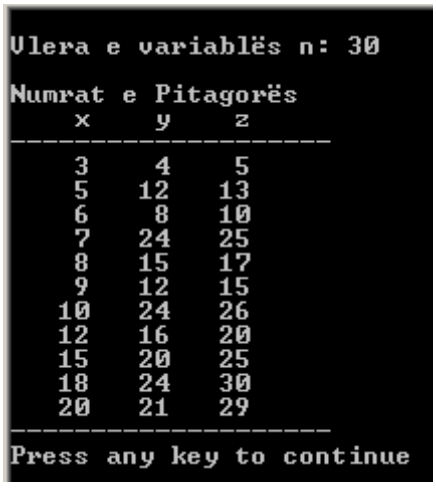


Fig.5.11
Rezultati i programit for12

Nga vlerat e fituara, përmes llogaritjes me dorë mund të vërtetohet se vlen ekuacioni i dhënë më sipër. Kështu, p.sh., për vlerat në rreshtin e parë të tabelës vlen:

$$3^2 + 4^2 = 5^2$$

Variabla e unazës e tipit karakter

Variabla e unazës, e cila realizohet me komandën `for`, mund të jetë edhe variabël e tipit karakter.

Shembull Programi përmes së cilit në ekran shtypen shkronjat e mëdha të alfabetit ndërkombëtar dhe kodet përkatëse të shkronjave të këtij alfabeti.

```
// Programi for14
#include <iostream>
using namespace std;
int main()
{
      char k;
      int x;
      cout << "Alfabeti ndërkombëtar dhe kodet e shkronjave"
           << endl
           << endl;

      for (k='A';k<='Z';k++)
      {
            x=k;
            cout << k
                 << "  "
                 << x
                 << "   ";
      }

      cout << endl
           << endl;
return 0;
}
```

Këtu, vlerat e variablës së unazës shkojnë mes vlerës fillestare `'A'` dhe asaj përfundimtare `'Z'`. Kjo do të thotë se gjatë ekzekutimit të programit, si vlera të variablës `k` merren të gjitha shkronjat e alfabetit ndërkombëtar, gjë që shihet edhe në pamjen e vlerave të shtypura në ekran, ashtu siç është dhënë në *Fig.5.12*.

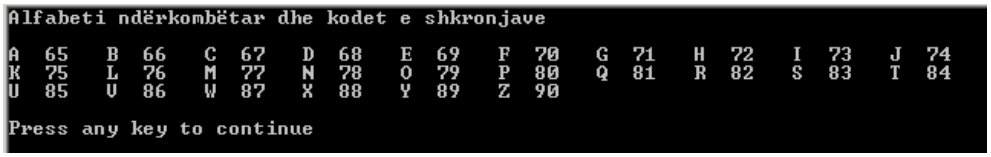


Fig.5.12 Rezultati i programit for14

Me program është paraparë që pranë çdo shkronje të shtypet edhe kodi përkatës. Kështu, p.sh., shkronjës A si kod i është shoqëruar numri 65 (p.sh., ashtu siç është kodi 137 për shkronjën e gjuhës shqipe ë).

Gjatë shfrytëzimit të shkronjave si vlera të tipit karakter duhet pasur parasysh se raporti mes tyre është:

```
'A' < 'B' < 'C' < … <'X' < 'Y' < 'Z'
```

Unazat e ndërthurura mund të formohen edhe duke shfrytëzuar variabla të tipit karakter.

Shembull

Programi përmes së cilit në ekran shtypen shkronjat e mëdha të alfabetit ndërkombëtar, duke shfytëzuar dy unaza të ndërthurura të formuara me variabla të tipit karakter.

```
// Programi p7
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      char x,y;
      for (x='Z';x>='A';x--)
      {
            for (y='A';y<=x;y++)
                  cout << setw(2)
                        << y;
            cout << endl;
      }
return 0;
}
```

Rezultati që fitohet në ekran pas ekzekutimit të programit të dhënë do të duket si në Fig.5.13.

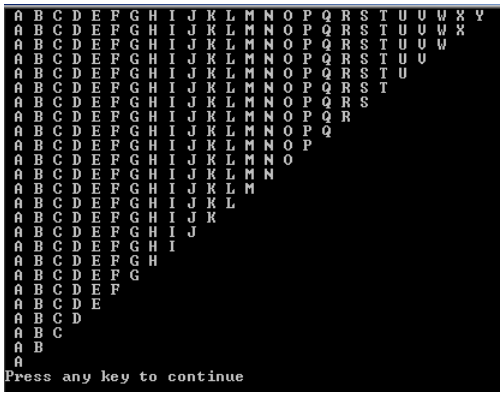


Fig.5.13
Rezultati i programit p7

Këtu, kur variabla `x` e unazës së jashtme e merr vlerën e parë `'Z'`, variabla `y` e unazës së brendshme i merr të gjitha vlerat prej `'A'` deri në vlerën e variablës `x`, e cila fillimisht e ka vlerën `'Z'`. Pastaj, njëlloj përsëritet procedura, por për vlerën e dytë të variablës `x`, e cila është `'Y'`. Kurse variabla e unazës së dytë i merr të gjitha vlerat prej `'A'` deri në `'Y'`. Dhe kështu procedura vazhdon deri në fund, kur `x='A'` dhe `y='A'`.

Unazat gjatë operimit me fusha

Për operim me anëtarët e vektorëve, të matricave ose edhe të fushave shumëdimensionale, indekset përkatëse mund të përcaktohen përmes vlerave të variablave të unazave, të cilat mund të realizohen me komandën `for`.

Operimi me vektorë

Meqë vektorët janë fusha njëdimensionale, për t'i shfrytëzuar anëtarët e tyre gjatë llogaritjeve të ndryshme indekset përkatëse mund të përcaktohen përmes variablës së një unaze.

Shembull Programi përmes së cilit llogaritet prodhimi i antëtarëve të vektorit `A(m)`.

```
// Programi for3
#include <iostream>
using namespace std;
int main()
{
      const m=6;
      int i,A[m]={3,2,6,5,8,4};
      double p;
      p=1;
      for (i=0;i<m;i++)
            p=p*A[i];
      cout << "Prodhimi p="
           << p
           << "\n";
return 0;
}
```

Pas ekzekutimit të programit të dhënë, si rezultat fitohet:

```
Prodhimi p=5760
```

ku vlera e prodhimit është fituar përmes llogaritjes:

```
p=3*2*6*5*8*4=5760
```

Edhe gjatë operimit me anëtarët e vektorëve, brenda trupit të unazës së realizuar me komandën for mund të paraqiten më shumë komanda.

Shembull

Programi përmes së cilit nga anëtarët e vektorit `Z(m)` me indeks çift formohet vektori i ri `G(m)`, duke i rritur anëtarët e tij për vlerën e indekseve përkatëse. Njëkohësisht, në program gjendet shuma e diferencave të vlerave të anëtarëve përkatës të vektorit të dhënë dhe të vektorit të formuar.

```
// Programi p8
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=10;
      char T[]="--------------------------";
      int i,k,G[5],Z[m]={5,-2,9,7,3,-8,4,12,10,6};
      cout << "\n    i   Z[i]    k    G[k]\n"
           << T
           << endl;
      k=-1;

      for (i=1;i<m;i=i+2)
      {
            k=k+1;
            G[k]=i+Z[i];
            cout << setw(5)
                 << i
                 << setw(6)
                 << Z[i]
                 << setw(6)
                 << k
                 << setw(7)
                 << G[k]
```

```
                << endl;
      }

      cout << T
           << endl;
return 0;
}
```

Në program, me qëllim të shfrytëzimit të anëtarëve me indeks tekë të matricës së dhënë `Z(m)` për formimin e vektorit të ri `G`, është shfrytëzuar komanda for:

```
for (i=1;i<m;i=i+2)
```

Përmes kësaj komande përcaktohet që të merren vetëm vlerat tekë të variablës `i` (vlera fillestare `i=1` dhe përmes shprehjes `i=i+2` rritet për `2`), e cila njëkohësisht shfrytëzohet si indeks për anëtarët e vektorit `Z(m)` që përdoren në llogaritjen e anëtarëve të vektorit `G`. Indeksi `k` i vektorit që formohet fillimisht është marrë `k=-1`, meqë vektori i ri `G` ende nuk ka asnjë anëtar. Por, para se te ky vektor të vendoset një anëtar i ri, vlera e indeksit rritet për `1`, përmes shprehjes:

```
k=k+1;
```

Në këtë mënyrë, indeksi i anëtarit të parë do të jetë:

```
k=-1+1=0
```

ashtu që anëtari në fjalë të vendoset në pozitën e parë të vektorit.

Nëse ekzekutohet programi i dhënë, rezultati që paraqitet në ekran do të duket si në *Fig.5.14*.

```
   i   Z[i]    k    G[k]
-------------------------
   1    -2     0     -1
   3     7     1     10
   5    -8     2     -3
   7    12     3     19
   9     6     4     15
-------------------------
Press any key to continue
```

Fig.5.14
Rezultati i programit p8

Në fund, vektori `G` do të përmbajë gjithsej `k+1` anëtarë, meqë indekset përkatëse shkojnë mes vlerave `0` dhe `k`.

Brenda unazës mund të ketë edhe degëzime të ndryshme, duke e shfrytëzuar komandën `if`.

Shembull Programi përmes së cilit llogaritet shuma `s` e katrorëve të anëtarëve negativë dhe e kubeve të anëtarëve pozitivë të vektorit `A(m)`.

```
// Programi for17
#include <iostream>
using namespace std;
int main()
{
      const m=6;
      int i,A[m]={5,-3,4,9,-5,8};
      double s,p;
      s=0;

      for (i=0;i<m;i++)
      {
            p=A[i]*A[i];
            if (A[i]<0)
                  s=s+p;
            else
                  s=s+p*A[i];
      }

      cout << "Shuma e kërkuar është s="
           << s
           << "\n";
return 0;
}
```

Në program, përmes komandës:

```
      if (A[i]<0)
            s=s+p;
      else
            s=s+p*A[i];
```

përcaktohet mbledhja e katrorit të anëtarit negativ, ose e kubit të anëtarit pozitiv të vektorit. Para kësaj komande, përmes shprehjes:

```
      p=A[i]*A[i];
```

gjendet katrori i anëtarit aktual të vektorit. Pastaj, kjo vlerë e llogaritur i shtohet shumës:

```
s=s+p;
```

nëse konstatohet se anëtari është negativ. Por, kur anëtari aktual i vektorit është pozitiv, për t'ia shtuar shumës kubin e anëtarit aktual:

```
s=s+p*A[i];
```

prodhimi p është shumëzuar edhe një herë me anëtarin aktual të vektorit.

Nëse ekzekutohet programi i dhënë, si rezultat në ekran do të fitohet:

```
Shuma e kërkuar është s=1464
```

Gjatë llogaritjes së shumës në fjalë është shfrytëzuar shprehja:

$s=5^3 + (-3)^2 + 4^3 + 9^3 + (-5)^2 + 8^3$

Komanda `if`, që paraqitet brenda trupit të unazës, mund të jetë edhe komandë e përbërë.

Shembull Programi përmes së cilit gjendet vlera absolute më e vogël `x` në vektorin e dhënë `R(m)`, si dhe indeksi përkatës `k`.

```
// Programi for6
#include <iostream>
using namespace std;
int main()
{
      const n=8;
      int i,k,x,R[n]={5,-8,6,4,-2,6,9,3};
      x=abs(R[1]);
      k=1;

      for (i=0;i<n;i++)
      {
            if (abs(R[i])<x)
            {
                  x=abs(R[i]);
                  k=i;
            }
      }

      cout << "Vlera absolute më e vogël x="
           << x
           << "\nIndeksi i anëtarit me këtë vlerë k="
           << k
```

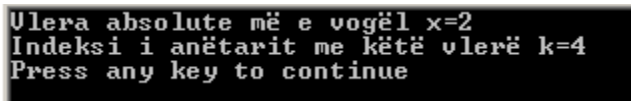
```
          << endl;
return 0;
}
```

Këtu, brenda unazës paraqitet komanda e përbërë `if`, në të cilën përfshihet dega me komandat për ruajtjen e anëtarit me vlerë absolute më të vogël si dhe e indeksit përkatës në vektor:

```
x=abs(R[i]);
k=i;
```

Nëse ekzekutohet programi i dhënë, në ekran rezultati do të duket si në *Fig.5.15*.

Fig.5.15
Rezultati i programit p8



Gjatë operimit me vektorë mund të paraqiten edhe unaza të ndërthurura.

Shembull

Programi përmes së cilit rradhiten sipas madhësisë anëtarët e vektorit `A(m)`, prej më të madhit kah më i vogli, duke e shfrytëzuar një version të metodës për sortim e cila njihet si `bubble sort`.

```
// Programi bubble
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=12;
      int i,j,x,Z[m]={4,7,9,-3,5,12,6,17,-2,13,3,8};

      cout << "\nVektori i dhënë\n"
           << "\nZ={";
      for (i=0;i<m;i++)
            cout << setw(3)
                 << Z[i];
      cout << "  }\n";

      for (i=0;i<(m-1);i++)
            for (j=0;j<(m-1-i);j++)
                  if (Z[j]>Z[j+1])
```

```
                    {
                          x=Z[j];
                          Z[j]=Z[j+1];
                          Z[j+1]=x;
                    }
      cout << "\nVektori i sortuar\n"
           << "\nZ={";
      for (i=0;i<m;i++)
            cout << setw(3)
                 << Z[i];
      cout << "  }\n\n";
return 0;
}
```

Në program është paraparë që anëtarët e vektorit `Z(m)` të rradhiten duke u nisur prej anëtarit më të vogël. Prandaj, gjatë procesit të sortimit në unazën e brendshme është vendosur komanda:

```
if (Z[j]>Z[j+1])
```

Nëse plotësohet kushti i shkruar brenda kësaj komande, përmes grupit të komandave:

```
x=Z[j];
Z[j]=Z[j+1];
Z[j+1]=x;
```

kompjuterit i urdhërohet që t'i ndërroi vlerat e anëtarëve `Z[j]` dhe `Z[j+1]`, duke e shfrytëzuar variablën ndërmjetësuese `x`.

Nëse ekzekutohet programi i dhënë, rezultati do të duket si në *Fig.5.15a*.

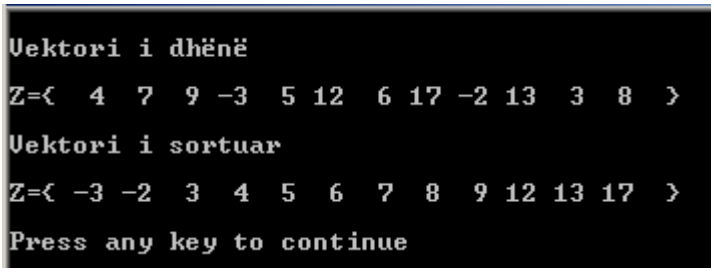


Fig.5.15a
Rezultati i programit bubble

Operimi me matrica

Matricat janë fusha dydimensionale dhe për operim me anëtarët e tyre gjatë llogaritjeve të ndryshme, idekset përkatëse mund të përcaktohen përmes variablave të dy unazave.

Shembull

Programi përmes së cilit formohet matrica `Z(m,n)`, duke i përcaktuar anëtarët e saj me shprehjen:

$$z_{ij} = i + j$$

```
// Programi p19
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=6,n=5;
      int i,j,Z[m][n];
      char d[]="-----------------------";

      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  Z[i][j]=i+j;

// Shtypja e matricës

      cout << "\n  Matrica e formuar Z\n"
           << d
           << endl;

      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(4)
                       << Z[i][j] ;
            cout << endl;
      }

      cout << d
           << endl;
return 0;
}
```

Në programin e dhënë janë shfrytëzuar dy unaza, përmes së cilave përcaktohen indekset e anëtarëve të matricës, të cilët llogariten duke i mbledhur vlerat e indekseve përkatëse. Për realizimin e unazës së jashtme shfrytëzohet komanda:

```
      for (i=0;i<m;i++)
```

kurse unaza e brendshme është realizuar me komandën:

```
for (j=0;j<n;j++)
```

Gjatë ekzekutimit të programit, për një vlerë të variablës `i` në unazën e jashtme, merren të gjitha vlerat e mundshme të variablës `j` në unazën e brendshme. Kjo do të thotë se, meqë me indeksin e parë përcaktohen rreshtat e matricës, pas zgjedhjes së rreshtit me vlerën e variablës i, merren të gjitha vlerat e mundshme të indeksit të dytë j, përkatësisht kalohet nëpër të gjitha kolonat e rreshtit të zgjedhur.

Në fund, pasi formohet komplet matrica, vazhdohet me ekzekutimin e dy unazave të tjera, përmes së cilave shtypet matrica e formuar. Këtu, unaza e brendshme formohet nga komandat:

```
for (j=0;j<m;j++)
      cout << setw(4)
           << Z[i][j] ;
```

kurse në unazën e jashtme përfshihen komandat brenda kllapave.

Përmes komandës së fundit në unazën e jashtme:

```
cout << endl;
```

kompjuterit i urdhërohet të kalojë në një rresht të ri, meqë ka përfunduar shtypja e rreshtit paraprak të matricës. Në këtë mënyrë, në ekran shihet forma fizike e matricës, duke i dalluar rreshtat e veçantë të saj.

Matrica e formuar pas ekzekutimit të programit të dhënë do të duket si në *Fig.5.16*.

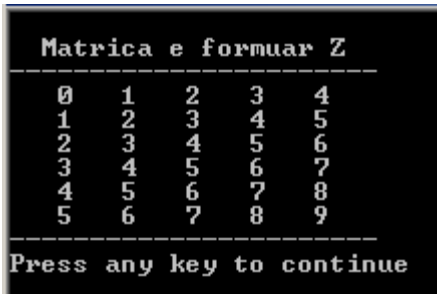


Fig.5.16
Rezultati i programit p19

Për shtypje të anëtarëve të matricës, njëkohësisht, mund të shfrytëzohen unazat brenda të cilave përcaktohen vlerat e tyre. Kështu, pas ndryshimeve të nevojshme, programi do të duket si në vijim.

```
// Programi p20
#include <iostream>
```

```
#include <iomanip>
using namespace std;
int main()
{
      const m=6,n=5;
      int i,j,Z[m][n];
      char d[]="----------------------";
      cout << "\n  Matrica e formuar Z\n"
           << d
           << endl;

      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
            {
                  Z[i][j]=i+j;
                  cout << setw(4)
                       << Z[i][j];
            }
            cout << endl;
      }

      cout << d
           << endl;
return 0;
}
```

Këtu, në unazën e brendshme, pasi përcaktohet një anëtar i matricës, ai edhe shtypet, duke e shfrytëzuar komandën:

```
cout << setw(4)
     << Z[i][j];
```

Rezultati që fitohet, nëse ekzekutohet ky version i programit, do të jetë i njëjtë me atë që u dha në *Fig.5.16*.

Unazat mund të përmbajnë edhe degëzime përmes komandës `if`.

Shembull Programi përmes së cilit numërohen anëtarët negativë (`x`) dhe anëtarët pozitivë (`y`) të matricës së dhënë `A(m,n)`.

```
// Programi for7
#include <iostream>
using namespace std;
int main()
{
```

```
      const m=4,n=5;
      int A[m][n]={{4,3,-4,-6,2},
                   {1,-5,2,7,-8},
                   {-3,9,5,-2,4},
                   {4,-1,-3,6,8}};
      int i,j,x,y;

      x=0;
      y=0;

      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  if (A[i][j]<0)
                        x=x+1;
                  else
                        y=y+1;

      cout << "Numri i anëtarëve negativë x="
           << x
           << "\nNumri i anëtarëve pozitivë y="
           << y
           << "\n";
return 0;
}
```

Këtu, komanda `if`, që është vendosur në unazën e brendshme:

```
      if (A[i][j]<0)
            x=x+1;
      else
            y=y+1;
```

përmban dy degë. Në njërën degë rritet numëratori i numrave negativë (`x`), kurse në degën tjetër - numëratori i numrave pozitivë (`y`).

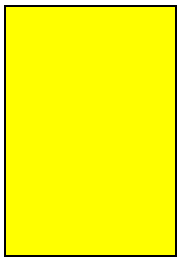
Nëse ekzekutohet programi i dhënë, rezultati që paraqitet në ekran do të duket si në *Fig.5.17*.

Fig.5.17
Rezultati i programit for7

```
Numri i anëtarëve negativë x=8
Numri i anëtarëve pozitivë y=12
Press any key to continue
```

Brenda unazave mund të përfshihen edhe komanda për degëzime të shumëfishta.

Shembull Programi përmes së cilit formohet matrica katrore `R(m,m)`,



duke i përcaktuar anëtarët e saj me shprehjen:

$$r_{ij} = \begin{cases} 0 & \text{për } i < j \\ i+1 & \text{për } i = j \\ -3 & \text{për } i > j \end{cases}$$

```
// Programi p18
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=6;
      int i,j,R[m][m];
      char z[]="--------------------";

      for (i=0;i<m;i++)
            for (j=0;j<m;j++)
                  if (i<j)
                        R[i][j]=0;
                  else
                        if (i==j)
                              R[i][j]=i+1;
                        else
                              R[i][j]=-3;

// Shtypja e matricës

      cout << "\n\n Matrica e formuar R\n"
           << z
           << endl;

      for (i=0;i<m;i++)
      {
            for (j=0;j<m;j++)
                  cout << setw(3)
                       << R[i][j];
            cout << endl;
      }

      cout << z
           << endl;
return 0;
```

```
}
```

Këtu, brenda unazave `for`, gjatë përcaktimit të anëtarëve të matricës, janë shfrytëzuar dy komanda `if`. Nëse plotësohet kushti `i<j` në komandën e parë:

```
if (i<j)
      R[i][j]=0;
```

përmes degës së dhënë përcaktohen anëtarët mbi diagonalen kryesore, duke i marrë me vlerë `0`. Kur nuk plotësohet kushti i përmendur më sipër, ekzekutohet komanda e dytë për degëzim:

```
if (i==j)
      R[i][j]=i+1;
```

me të cilën testohet se a është `i=j`. Nëse ky kusht plotësohet, anëtarët në diagonalen kryesore të matricës llogariten me shprehjen e dhënë në degën përkatëse, përkatësisht vlerat e tyre gjenden duke e rritur indeksin `i` për `1`. Përndryshe, nëse nuk plotësohet asnjëri nga kushtet në dy komandat `if`, përcaktohen anëtarët nën diagonalen kryesore, duke ua ndarë vlerat `-3`.

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.5.18*.

Fig.5.18
Rezultati i programit p18

Gjatë operimit me anëtarët e matricave, meqë kemi të bëjmë me fusha dydimensionale, anëtarët mund të përcaktohen me dy indekse. Kjo e imponon nevojën që, për zgjedhje të indekseve, të shfrytëzohen variablat e dy unazave të ndërthurura.

Shembull

Programi përmes së cilit formohet vektori `F`, duke i shfrytëzuar anëtarët e matricës `A(m,n)`, të cilët janë më të mëdhenj se numri `x`, i cili kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi p9
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=4,n=5;
      int A[m][n]={{6,2,4,17,-2},
                   {9,-1,-2,8,3},
                   {-6,4,11,3,5},
                   {12,7,15,2,6}};
      int i,j,k,x;
      int F[m*n];
      char T[]="-----------------------";
      cout << "\nVlera e numrit x: ";
      cin >> x;
      cout << "\n Vektori i formuar\n\n"
           << "     k     F[k]\n"
           << T
             << endl;
      k=-1;

      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
            {
                  if (A[i][j]>x)
                  {
                        k=k+1;
                        F[k]=A[i][j];
                        cout << setw(6)
                             << k
                             << setw(8)
                             << F[k]
                             << endl;
                  }
            }

      cout << T
           << endl;
return 0;
}
```

Në fillim të programit, gjatë deklarimit të tipit të vektorit F, përmes shprehjes:

```
      int F[m*n];
```

janë rezervuar vende sa ka edhe matrica, kjo sepse në rast të veçantë mund të ndodhë që të gjithë anëtarët e matricës të jenë numra më të mëdhenj se numri x, i cili kompjuterit i jepet përmes tastierës.

Në trupin e unazës së brendshme, e cila përcaktohet përmes shprehjes:

```
for (j=0;j<n;j++)
```

këtu, fillimisht është përfshirë komanda `if` me një degë të përbërë, në të cilën, nëse plotësohet kushti:

```
if (A[i][j]>x)
```

përmes shprehjes:

```
F[k]=A[i][j];
```

vektorit `F` i shtohet anëtari `A[i][j]`, duke e rritur fillimisht vlerën e indeksit për `1`, përmes shprehjes:

```
k=k+1;
```

Pastaj, brenda kësaj unaze është vendosur edhe komanda për shtypje të indekseve dhe e vlerave të anëtarëve të vektorit që formohet `F`.

Variabla `i` e unazës së jashtme, e cila realizohet përmes komandës:

```
for (i=0;i<m;i++)
```

shfrytëzohet vetëm për zgjedhjen e indekseve të rreshtave të matricës `A`.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës, si vlerë hyrëse kompjuterit i jepet vlera `5`, rezultati në ekran do të duket si në *Fig.5.19*.

Fig.5.19
Rezultati i programit p9

Përmes dy unazave njëkohësisht mund të përcaktohen indekset për operim me anëtarët e dy ose edhe të më shumë matricave.

Shembull

Programi përmes së cilit nga anëtarët e matricës së dhënë `Z(m,n)` formohet matrica `D(m,n)`, duke i shumëzuar anëtarët përkatës të matricës `Z(m,n)` me konstanten `k`.

```
// Programi p9
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=4,n=5;
      int Z[m][n]={{3,7,-2,4,5},
                   {1,5,12,9,6},
                   {-4,11,3,6,8},
                   {14,9,15,0,7}};
      int i,j,k,D[m][n];
      char T[]="-------------------------------------";
      cout << "\nVlera e konstantes k: ";
      cin >> k;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  D[i][j]=k*Z[i][j];

// Shtypja e matricës

      cout << "\n\n        Matrica e formuar D\n"
           << T
           << endl;
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(7)
                       << D[i][j] ;
            cout << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Në pjesën e parë të programit të dhënë, përmes dy unazave `for`, formohet matrica e re `D`. Anëtarët e saj llogariten duke e vendosur brenda dy unazave shprehjen:

```
D[i][j]=k*Z[i][j];
```

ku, siç shihet, anëtarët përkatës të matricës `Z` shumëzohen me konstanten `k`.

Nëse ekzekutohet programi i dhënë, rezultati që fitohet në ekran do të duket si në *Fig.5.20*.

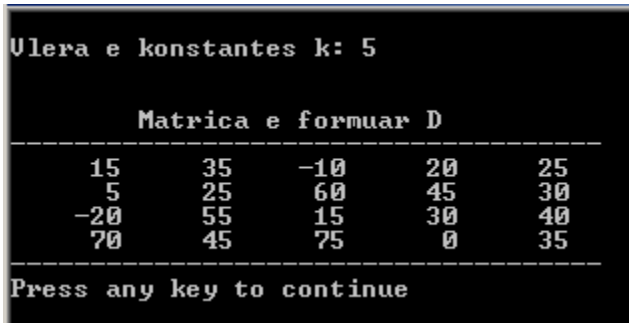


Fig.5.20
Rezultati i programit p9

Për shtypjen e saj janë shfrytëzuar dy unazat e vendosura në pjesën e dytë të programit, gjë që është shpjeguar edhe më parë.

Në programin e dhënë, komandat për shtypjen e anëtarëve të matricës që formohet mund të vendosen edhe brenda unazave që shfrytëzohen për formimin e saj. Pas ndryshimeve të bëra për këtë qëlllim, programi përkatës do të duket si në vijim.

```
// Programi p9a
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=4,n=5;
      int Z[m][n]={{3,7,-2,4,5},
                   {1,5,12,9,6},
                   {-4,11,3,6,8},
                   {14,9,15,0,7}};
      int i,j,k,D[m][n];
      char T[]="-------------------------------------";
      cout << "\nVlera e konstantes k: ";
      cin >> k;
```

```
      cout << "\n\n        Matrica e formuar D\n"
           << T
           << endl;
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
            {
                  D[i][j]=k*Z[i][j];
                  cout << setw(7)
                       << D[i][j] ;
            }
            cout << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Këtu, meqë unazat përmbajnë më shumë komanda, për vendosjen e komandave që përfshihen brenda tyre janë shfrytëzuar kllapat. Unaza e brendshme me komandat që përfshihen brenda saj duket kështu:

```
      for (j=0;j<n;j++)
      {
            D[i][j]=k*Z[i][j];
            cout << setw(7)
                 << D[i][j] ;
      }
```

kurse nën kllapat e unazës së jashtme, e cila formohet me komandën:

```
      for (i=0;i<m;i++)
```

përveç komandave të unazës së brendshme, përfshihet edhe komanda për kalim në rresht të ri:

```
      cout << endl;
```

pasi të ketë përfunduar shtypja e rreshtit paraprak të matricës.

Nëse ekzekutohet ky version i programit, për vlerën hyrëse `k=5`, rezultati që fitohet në ekran do të jetë plotësisht i njëjtë me atë që u dha në *Fig.5.20*.

Shembull Programi përmes së cilit gjenden shumat e rreshtave të veçantë të matricës së dhënë `F(m,n)`.

```
// Programi for16
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=5,n=4;
      int F[m][n]={{5,2,14,8},
                   {7,9,1,-6},
                   {8,3,15,4},
                   {12,6,5,-1},
                   {6,4,-5,2}};
      int i,j,s;
      char T[]="------------------";
      cout << "Shumat e rreshtave\n"
           << "\n Rreshti   Shuma\n"
           << T
           << endl;

      for (i=0;i<m;i++)
      {
            s=0;
            for (j=0;j<n;j++)
                  s=s+F[i][j];
      cout << setw(5)
           << i+1
           << setw(10)
           << s
           << endl;
      }

      cout << T
           << endl;
return 0;
}
```

Në program, përmes unzës së brendshme:

```
      for (j=0;j<n;j++)
            s=s+F[i][j];
```

gjendet shuma e rreshtit të `i`-të të matricës `F(m,n)`. Kurse, për zgjedhje të rreshtave të matricës dhe për shtypje të shumave të rreshtave të veçantë shfrytëzohet unaza e jashtme, e formuar me komandën:

```
     for (i=0;i<m;i++)
```

Në të përfshihen komandat brenda kllapave, bashkë me komandat që kanë të bëjnë me unazën e brendshme.

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.5.21*.

```
Shumat e rreshtave

 Rreshti   Shuma
------------------
    1        29
    2        11
    3        30
    4        22
    5         7
------------------
Press any key to continue
```

Fig.5.21
Rezultati i programit for16

Shembull Programi nga detyra paraprake i modifikuar ashtu që shumat shtypen pasi paraprakisht janë shtypur anëtarët e rreshtit përkatës të matricës.

```
// Programi for16a
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=5,n=4;
      int F[m][n]={{5,2,14,8},
                   {7,9,1,-6},
                   {8,3,15,4},
                   {12,6,5,-1},
                   {6,4,-5,2}};
      int i,j,s;
      char T[]="-----------------------------";
      cout << "Shumat e rreshtave\n"
           << "\n    M a t r i c a       Shuma\n"
           << T
           << endl;
      s=0;
      for (i=0;i<m;i++)
```

```
      {
            s=0;
            for (j=0;j<n;j++)
            {
                  s=s+F[i][j];
                  cout << setw(5)
                       << F[i][j];
            }
      cout << setw(8)
           << s
           << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Në këtë version të programit, përmes unazës së brendshme:

```
      for (j=0;j<n;j++)
      {
            s=s+F[i][j];
            cout << setw(5)
                 << F[i][j];
      }
```

përveç që llogaritet shuma e anëtarëve të rreshtit të `i`-të, njëkohësisht shtypen edhe anëtarët e këtij rreshti. Kurse, për shtypjen e vlerave të shumave të rreshtave të veçantë shfrytëzohen komandat:

```
      cout << setw(8)
           << s
           << endl;
```

të cilat bashkë me komandat e unazës së brendshme përfshihen në unazën e jashtme, e cila përcaktohet me komandën:

```
      for (i=0;i<m;i++)
```

Brenda kësaj unaze, për çdo vlerë të variablës i (indeksit të rreshtit), njëkohësisht merret edhe vlera fillestare `s=0` e shumës së anëtarëve të rreshtit përkatës.

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.5.22*.

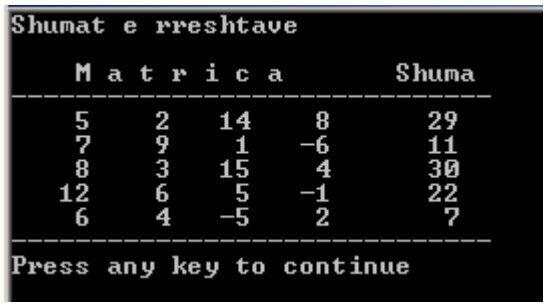


Fig.5.22
Rezultati i programit for16a

Këtu, në kolonën mbi të cilën figuron teksti Shuma, janë shënuar vlerat e shumave të rreshtave të veçantë të matricës.

Operimi me më shumë fusha

Në një program mund të operohet njëkohësisht me më shumë fusha. Gjatë kësaj, për zgjedhjen e indekseve të anëtarëve të fushave, të cilat marrin pjesë në llogaritje të ndryshme, shfrytëzohen edhe disa unaza njëkohësisht.

Shembull Programi përmes së cilit numërohen zanoret e alfabetit ndërkombëtar në fjalinë e cila lexohet.

```
// Programi p21
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=50,n=6;
      char Z[n]={'a','e','o','i','y','u'};
      int i,j;
      int F[n]={0,0,0,0,0,0};
      char A[m],x[]="--------------------";
      cout << "\nFjalia që lexohet:\n\n";

      cin.getline(A,m);

      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  if (A[i]==Z[j])
                        F[j]=F[j]+1;

// ----- Shtypja e rezultatit ------

      cout << "\n  Zanorja    Numri\n"
           << x
           << endl;
      for (j=0;j<n;j++)
            cout << setw(6)
```

```
                << Z[j]
                << setw(10)
                << F[j]
                << endl;
      cout << x
           << endl;
return 0;
}
```

Në programin e dhënë, fjalia e cila kompjuterit i jepet përmes tastierës ruhet te vektori `A(m)`. Për leximin e kësaj fjalie është shfrytëzuar komanda:

```
cin.getline(A,m);
```

Gjatë numërimit të zanoreve në fjali, për zgjedhje të shkronjave të përfshira në te shfrytëzohet unaza e jashtme:

```
for (i=0;i<m;i++)
```

Secila nga këto shkronja krahasohet me anëtarët e vektorit `Z(n)` ku ruhen zanoret, duke e shfrytëzuar unazën e brendshme:

```
for (j=0;j<n;j++)
```

Në momentin kur plotësohet kushti i cili është shënuar te komanda:

```
if (A[i]==Z[j])
```

rritet për `1` vlera e anëtarit përkatës `F(j)` të vektorit në të cilin ruhen shumat e zanoreve të numëruara, duke e shfrytëzuar shprehjen:

```
F[j]=F[j]+1;
```

Meqë kemi të bëjmë me numërim, vlerat fillestare të anëtarëve të vektorit `F(n)` janë përcaktuar përmes deklarimit:

```
int F[n]={0,0,0,0,0,0};
```

Ky deklarim mund të bëhet edhe duke e shënuar vlerën zero të së paku vetëm njërit anëtar të vektorit, kështu:

```
int F[n]={0};
```

Nëse ekzekutohet programi i dhënë, për fjalinë:

```
Mësimi i gjuhës programuese
```

të cilën kompjuterit ia japim përmes tastierës, rezultati do të duket si në *Fig.5.23*.

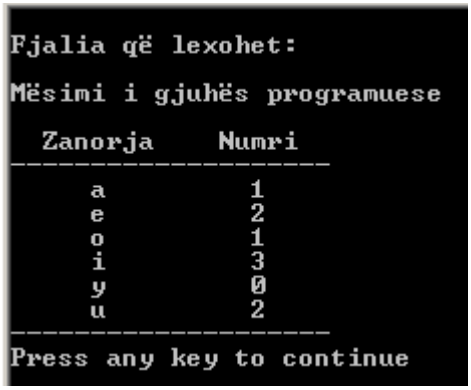


Fig.5.23
Rezultati i programit p21

Në fund të programit, gjatë shtypjes së anëtarëve të vektorëve `Z(n)` dhe `F(n)`, për përcaktimin e indekseve përkatëse është shfrytëzuar vetëm një unazë.

Brenda programit njëkohësisht mund të operohet edhe me më shumë matrica.

Shembull

Programi përmes së cilit formohet matrica `C(m,n)`, duke i shfrytëzuar anëtarët e matricave `A(m,n)` dhe `B(m,n)`, në bazë të parimeve vijuese.
Nëse çiftet e anëtarëve që kanë pozita të njëjta në matricat `A` dhe `B` janë me parashenja:
- të njëjta, anëtari i matricës `C` merret i barabartë me anëtarin korrespondues të matricës `A`;
- të ndryshme, anëtari i matricës `C` merret i barabartë me numrin `99`.

```
// Programi p22
#include <iostream>
#include <iomanip>
using namespace std;
```

```
int main()
{
      const m=5,n=4;
      int A[m][n]={{4,-5,9,3},
                   {-7,13,6,-2},
                   {8,-3,12,-4},
                   {13,9,-2,6},
                   {-4,-1,7,5}};
      int B[m][n]={{4,-3,5,8},
                   {2,-8,12,-5},
                   {-2,9,17,14},
                   {9,-3,5,15},
                   {19,-7,-2,4}};
      int i,j,C[m][n];
      char T[]="--------------------";
      cout << "\nMatrica e formuar C\n"
           << T
           << endl;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
            {
                  if ((A[i][j]<0 && B[i][j]<0)
                                 ||
                     (A[i][j]>=0 && B[i][j]>=0))
                        C[i][j]=A[i][j];
                  else
                        C[i][j]=99;
            }
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(4)
                       << C[i][j] ;
            cout << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Në program, gjatë mbushjes së matricës `C`, si indekse shfrytëzohen variablat e dy unazave, të cilat njëkohësisht i paraqesin edhe indekset e anëtarëve të dy matricave të tjera. Me qëllim të përcaktimit të parimit për mbushje të anëtarëve të matricës është shfrytëzuar komanda `if`:

```
      if ((A[i][j]<0 && B[i][j]<0)
                     ||
         (A[i][j]>=0 && B[i][j]>=0))
```

Me pjesën e parë të shprehjes logjike brenda kësaj komande:

```
(A[i][j]<0 && B[i][j]<0)
```

kontrollohet se a janë anëtarët në pozitat e njëjta të dy matricave numra negativë (meqë mes dy pjesëve të shprehjes figuron operatori `&&`, i cili i përgjigjet operacionit logjik `DHE`). Kurse me pjesën e dytë të shprehjes logjike në fjalë:

```
(A[i][j]>=0 && B[i][j]>=0))
```

kontrollohet se a janë të dy anëtarët numra pozitivë. Nëse është e vërtetë njëra nga këto dy pjesë të shprehjes logjike brenda komandës `if` (meqë mes tyre është vendosur operatori `||`, që i përgjigjet operacionit logjik `OSE`), anëtari i matricës që formohet merret i barabartë me anëtarin korrespondues të matricës `A`:

```
C[i][j]=A[i][j];
```

Përndryshe, nëse nuk plotësohet qoftë edhe njëra nga dy pjesët e shprehjes logjike brenda komandës `if`, përkatësisht nëse shprehja brenda kllapave të kësaj komande është e pavërtetë, anëtarit në fjalë të matricës `C` i jepet vlera `99`. Pas ekzekutimit të programit, rezultati në ekran do të duket si në *Fig.5.24*.

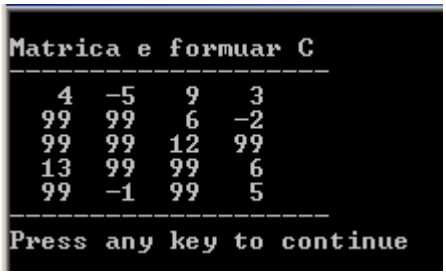


Fig.5.24
Rezultati i programit p22

Kapërcimi i komandave brenda unazës

Sipas nevojës, ekzekutimi i një pjese të caktuar brenda unazës mund të kapërcehet, pa dalë nga unaza, duke e shfrytëzuar komandën `continue`, ose komandën e kapërcimit pa kusht `goto`.

Nëse brenda unazës vendoset komanda `continue`, gjatë ekzekutimit të saj kompjuteri do ta kapërcejë pjesën vijuese brenda trupit të unazës.

Shembull

Programi përmes së cilit llogaritet shuma (`a`) e numrave natyrorë mes `1` dhe `n`, shuma e katrorëve (`b`) dhe e kubeve (c) të këtyre numrave. Gjatë gjetjes së shumave, në llogaritje nuk përfshihen

numrat natyrorë 3 dhe 7. Vlera e variablës n kompjuterit i jepet përmes tastierës si vlerë hyrëse.

```
// Programi p5
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,n;
      float a,b,c;
      char T[]="----------------------------";
      cout << "\nVlera e variablës n: ";
      cin >> n;
      a=0;
      b=0;
      c=0;
      cout << "\n  Numri    a      b      c"
           << endl
           << T
             << endl;
      for (i=1; i<=n; i++)
      {
            if ((i==3) || (i==7))
               continue;
            a=a+i;
            b=b+i*i;
            c=c+i*i*i;
            cout << setw(5)
                 << i
                 << setw(7)
                 << a
                 << setw(7)
                 << b
                 << setw(7)
                 << c
                 << endl;
      }
      cout << T
           << endl;
return 0;
}
```

Në program, me qëllim të eliminimit të pjesëmarrjes në llogaritje të shumave për vlerat 3 dhe 7 të variablës i, në degën e komandës if është vendosur komanda continue, kështu:

```
if ((i==3) || (i==7))
   continue;
```

Përmes kësaj komande kapërcehet pjesa e shprehjeve për llogaritje të shumave dhe komanda për shtypje të rreshtit përkatës në tabelë, përkatësisht për `i=3` dhe `i=7` kalohet në vlerat vijuese të variablës `i`, pa dalë nga unaza.

Nëse ekzekutohet programi i dhënë, për vlerën hyrëse `n=10`, rezultati në ekran do ta ketë pamjen e dhënë në *Fig.5.25*.

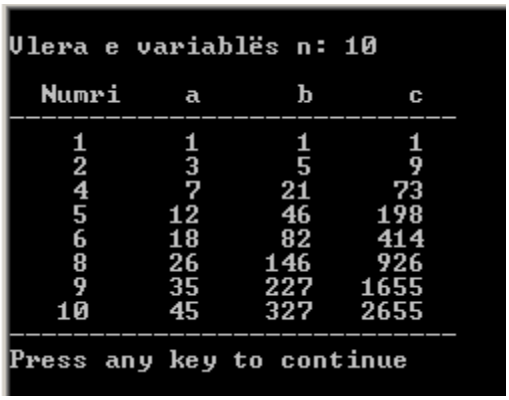


Fig.5.25
Rezultati i programit p5

Programi `p5` mund të shkruhet edhe duke e shfrytëzuar komandën e kapërcimit pa kusht `goto`, në vend të komandës `continue`. Pjesa e unazës së këtij programi është dhënë në vijim.

```
for (i=1; i<=n; i++)
{
      if ((i==3) || (i==7))
         goto Jeta;
      a=a+i;
      b=b+i*i;
      c=c+i*i*i;
      cout << setw(5)
           << i
           << setw(7)
           << a
           << setw(7)
           << b
           << setw(7)
           << c
           << endl;
Jeta:;
}
```

Këtu, kapërcimi i nevojshëm është realizuar me komandën `goto Jeta`, në vend të kapërcimit me komandën `continue`.

Shembull

Programi për llogaritjen e vlerës së funksionit:

$$g = x + 2 \sum_{\substack{i=1 \\ (i \neq 2,5)}}^{n+1} (2i + k)$$

për vlera të ndryshme të numrit natyrorë `k`, mes vlerave `1` dhe `m`. Njëkohësisht, gjatë llogaritjes së shumës kapërcehen anëtarët për vlerat `2` dhe `5` të variablës `i`. Vlerat e variablave `m`, `n` dhe `x` kompjuterit i jepen përmes tastierës si vlera hyrëse.

```
// Programi p23
#include <iostream>
using namespace std;
int main()
{
      int i,k,m,n;
      double x,g,s;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës m: ";
      cin >> m;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << endl;

      for (k=1;k<=m;k++)
      {
            cout << "k="
                 << k
                 << "  i: ";
            s=0;
            for (i=1;i<=n+1;i++)
            {
                  if ((i==2) || (i==5))
```

```
                        continue;
                    else
                    {
                        cout << i
                             << "  ";
                        s=s+(2*i+k);
                    }
                }
                g=x+2*s;
                cout << " g="
                     << g
                     << endl;
        }
return 0;
}
```

Në unazën e brendshme të programit të dhënë, është vendosur komanda `if`:

```
if ((i==2) || (i==5))
```

Nëse plotësohet kushti i shënuar brenda kllapave të kësaj komande, përkatësisht nëse vlera e shprehjes:

```
(i==2) || (i==5)
```

është `true`, do të ekzekutohet komanda:

```
continue;
```

e cila është vendosur në degën e parë të komandës `if`. Përmes kësaj komande kalohet në fund të unazës, përkatësisht llogaritja vazhdohet me vlerën vijuese të variablës `i`, ashtu siç shihet edhe në tabelën e rezultateve të shtypura, që është dhënë në *Fig.5.26*.

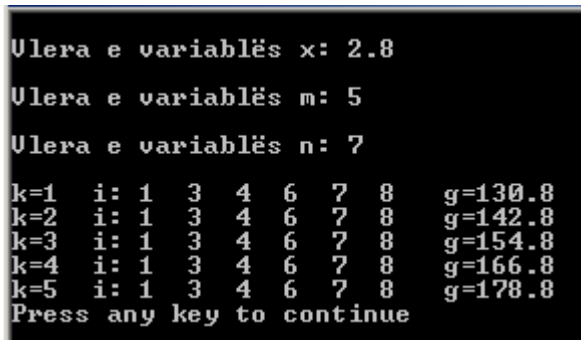


Fig.5.26
Rezultati i programit p23

Shembull Programi i dhënë në shembullin paraprak, por tek i cili parashihet kapërcimi i ekzekutimit të komandave brenda unazës së jashtme, për vlerat `3` dhe `4` të variablës `k`.

```
// Programi p24
#include <iostream>
using namespace std;
int main()
{
.................................
      for (k=1;k<=m;k++)
      {
            if ((k==3) || (k==4))
                  continue;
            cout << "k="
                 << k
                 << "  i: ";
            s=0;
.................................
      }
return 0;
}
```

Për dallim nga versioni paraprak i programit, këtu është shtuar komanda:

```
if ((k==3) || (k==4))
      continue;
```

përmes së cilës, për vlerat `3` dhe `4` të variablës `k`, kapërcehet pjesa e komandave vijuese brenda unazës së jashtme. Këtu përfshihet edhe kapërcimi i unazës së brendshme përmes së cilës zgjedhen vlerat e variablës `i`. Nëse pas shtesës në fjalë ekzekutohet versioni i ri i programit, rezultati që do të fitohet në ekran për vlerat e njëjta hyrëse si edhe te shembulli paraprak do të duket si në *Fig.5.27*.

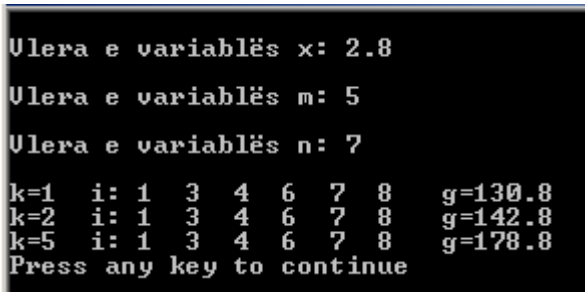


Fig.5.27
Rezultati i programit p23

Prej këtu qartë shihet se janë kapërcyer llogaritjet për vlerat 3 dhe 4 të variablës k.

Dalja nga unaza

Përsëritja e ekzekutimit të komandave brenda trupit të unazës mund të ndërpritet, duke dalë prej saj përmes komandës goto, ose komandës break.

Dalja përmes komandës goto

Gjatë ekzekutimit të unazës, vlerat e rezultateve të caktuara brenda trupit të saj mund të kontrollohen, për t'i shfrytëzuar ato me qëllim të ndërprerjes së ekzekutimit të unazës. Gjatë kësaj ndërprerjeje, për dalje nga unaza, shfrytëzohet komanda goto për kapërcim pa kusht jashtë saj, pa i marr variabla e unazës të gjitha vlerat e mundshme.

Shembull

Programi përmes së cilit gjendet vlera e funksionit:

$$y = 2x + 3\sum_{k=2}^{n+1}\left\{\sin(2x) - \frac{4}{k-x}\right\}$$

nëse vlerat x dhe n kompjuterit i jepen përmes tastierës.

```
// Programi for10
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
      int k,n;
      double x,y,s;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      s=0;
      for (k=2;k<=n+1;k++)
      {
```

```
            if ((k-x)==0)
            {
              cout << "\nRasti kur pjesëtohet me zero"
                   << "\nLlogaritja e shumës nuk ka kuptim"
                   << endl;
              goto Fundi;
            }
            else
              s=s+(sin(2*x)-4/(k-x));
      }
      y=2*x+3*s;
      cout << "\nVlera e funksionit y="
           << y
           << endl;
Fundi:
return 0;
}
```

Në fillim të programit të dhënë më sipër është vendosur komanda paraprocesorike:

```
#include <math.h>
```

sepse brenda fajlit `math.h` është përcaktuar funksioni trigonometrik `sin`, i cili shfrytëzohet brenda programit.

Nëse ekzekutohet programi i dhënë, për vlera hyrëse të caktuara, p.sh., për `x=7.5` dhe `n=3`, rezultati në ekran do të duket si në *Fig.5.28*.

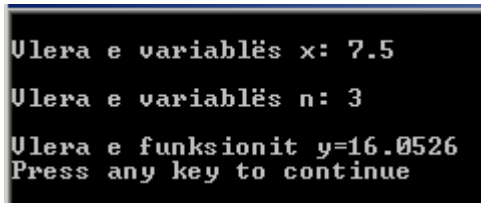


Fig.5.28
Rezultati i programit for10

Nëse përmes tastierës, për variablën `x`, kompjuterit i jepet ndonjë vlerë e plotë, shprehja `k-x` nën thyesë mund të ketë vlerë zero. Pasi pjesëtimi me zero nuk ka kuptim, në ekran do të gjenerohet mesazhi përkatës, ashtu siç shihet në *Fig.5.29*.

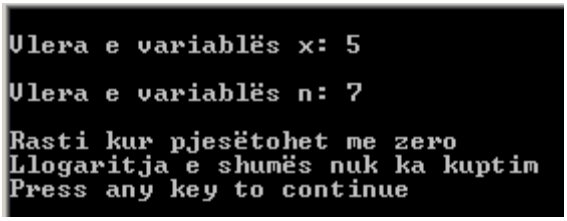


Fig.5.29
Rezultati i programit for10 për x=0

Kapërcimi në mesazhin në fjalë mundësohet duke e kontrolluar përmes komandës:

```
if ((k-x)==0)
```

se a është zero vlera e emëruesit `k-x`. Pas shtypjes së mesazhit, ekzekutimi i programit përcillet në pjesën pas labelës `Fundi:`, përmes komandës `goto Fundi`.

Shembull

Programi përmes së cilit gjendet anëtari i parë në vektorin `F(m)`, i cili është me vlerë absolute më të madhe se numri pozitiv x, si dhe pozita e tij në vektor (indeksi përkatës).

```
// Programi p3
#include <iostream>
using namespace std;
int main()
{
      const m=6;
      int i,x,F[m]={5,-3,4,-9,5,8};
      cout << "Vlera e variablës x: ";
      cin >> x;
      for (i=0;i<m;i++)
      {
            if (abs(F[i])>x)
            {
                  cout << "\nAnëtari i kërkuar: "
                       << F[i];
                  cout << "\nIndeksi përkatës: "
                       << i
                       << endl;
                  goto Jasht;
            }
      }
      cout << "\nNë vektor nuk ka anëtar të tillë"
           << endl;
Jasht:
return 0;
}
```

Në program, pasi të plotësohet kushti:

```
if (abs(F[i])>x)
```

përmes komandave:

```
cout << "\nAnëtari i kërkuar: "
     << F[i];
cout << "\nIndeksi përkatës: "
     << i
     << endl;
```

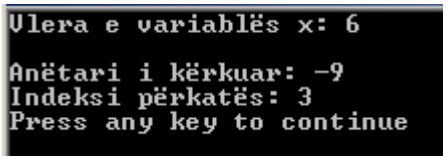
shtypet vlera e anëtarit `F[i]`, i cili është më i madh se numri `x` për nga vlera absolute, si dhe indeksi përkatës. Pastaj, përmes komandës:

```
goto Jasht;
```

dilet nga unaza dhe shkohet në fund të programit.

Nëse programi i dhënë ekzekutohet për vlerën `6` të variablës hyrëse `x`, rezultati në ekran do të duket si në *Fig.5.30*.

Fig.5.30
Rezultati i programit p3



Duke i pasur parasysh anëtarët e vektorit `F(m)`, nga rezultati i fituar shihet se anëtari i parë që ka vlerë absolute më të madhe se numri x=6 është anëtari i katërt, me indeks `3` (meqë indekset fillojnë me vlerën `0`) dhe vlera e tij është `-9`.

Nëse në vektor nuk gjendet asnjë anëtar i cili e plotëson kushtin e dhënë, unaza do të ekzekutohet deri në fund dhe pas daljes nga unaza do të shtypet mesazhi:

```
Në vektor nuk ka anëtar të tillë
```

Kjo do të ndodhë, p.sh., nëse si vlerë hyrëse për variablën `x`, përmes tastierës kompjuterit i jepet numri `15`. Në këtë rast në ekran do ta kemi pamjen e dhënë në *Fig.5.31*.

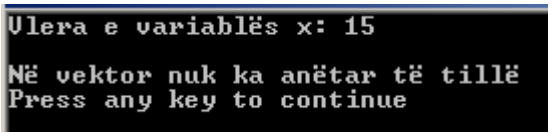


Fig.5.31 Rezultati i programit p3 kur nuk gjendet asnjë anëtar i vektorit që e plotëson kushtin e dhënë

Dalja përmes komandës break

Nëse brenda unazës vendoset komanda `break`, ekzekutimi i programit vazhdon me komandën e cila gjendet menjëherë pas kllapës së fundit të trupit të saj, duke dalë pa kusht nga unaza.

Shembull

Programi përmes së cilit mblidhen anëtarët e vektorit `B(n)` gjersa nuk arrihet vlera kufitare e shumës `z`, e cila vlerë kompjuterit i jepet përmes tastierës.

```
// Programi p4
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const n=7;
      int i,B[n]={2,5,-3,4,1,-6,9};
      double s,z;
      bool k;
      char h[]="-------------------------";
      cout << "\nVlera kufitare z: ";
      cin >> z;
      k=false;
      cout << "\n    Nr   Anëtari    Shuma\n"
           << h
           << endl;
      s=0;
      for (i=0;i<n;i++)
      {
            s=s+B[i];
            cout << setw(6)
                 << i
                 << setw(7)
                 << B[i]
                 << setw(10)
                 << s
                 << endl;
            if (s>z)
            {
                  cout << h
                       << endl;
                  cout << "\nShuma e llogaritur: "
```

```
                    << s;
               cout << "\nNumri i anëtarëve në shumë: "
                    << i+1
                    << endl;
               k=true;
               break;
          }
     }
     if (k==false)
         cout << "\nVlera e kërkuar e shumës nuk u gjet"
              << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, për vlerën kufitare të shumës `z=7`, të cilën kompjuterit ia japim përmes tastierës, rezultati në ekran do të duket si në *Fig.5.32*.

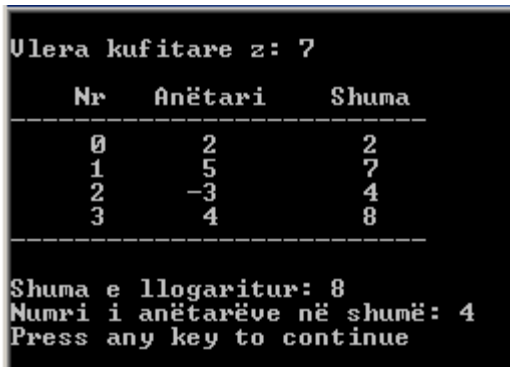


Fig.5.32
Rezultati i programit p4, për rastin kur arrihet të jetë s>z

Siç shihet në program, për dalje nga unaza kontrollohet se a e ka tejkaluar shuma vlerën kufitare `z`, duke e shfrytëzuar komandën:

```
if (s>z)
```

Nëse plotësohet kushti i dhënë, kalohet në ekzekutimin e pjesës së programit:

```
{
      cout << h
           << endl;
      cout << "\nShuma e llogaritur: "
           << s;
      cout << "\nNumri i anëtarëve në shumë: "
           << i+1
           << endl;
      k=true;
      break;
```

```
}
```

përmes së cilës fillimisht shtypet një rresht me viza (`h`), pastaj vlera e shumës (`s`), dhe në fund edhe numri i anëtarëve të cilët janë përfshirë në shumë `(i+1)`. Me qëllim të ruajtjes së informatës se është kaluar nëpër këtë degë të komandës `if`, variablës `k` i është shoqëruar vlera `true`, përmes shprehjes:

```
k=true;
```

të cilës para se të fillohet me ekzekutimin e unazës `for`, i është ndarë vlera fillestare:

```
k=false;
```

Në fund të degës në fjalë të komandës `if` është shkruar edhe komanda:

```
break;
```

përmes së cilës kompjuterit i urdhërohet të dalë nga unaza, meqë u gjet shuma që e kërkuam. Pas kësaj, ekzekutimi i programit vazhdon me komandën e parë, e cila gjendet pas kllapës së fundit të unazës që është realizuar me komandën `for`.

Nëse kushti i shtruar te komanda `if` brenda unazës nuk plotësohet edhe pas mbledhjes së të gjithë anëtarëve të vektorit, dalja prej unazës do të jetë e rregullt (pa ndërprerje me komandën `break`). Edhe në këtë rast, përsëri ekzekutimi i programit do të vazhdojë me komandën e parë pas kllapës së fundit të unazës `for`. Për këtë qëllim është paraparë që në këtë rast të shtypet mesazhi:

```
Vlera e kërkuar e shumës nuk u gjet
```

Por, meqë në të dy rastet vazhdohet me ekzekutimin e pjesës së programit pas kllapës së fundit të unazës, përmes komandës:

```
if (k==false)
```

mesazhi në fjalë do të shtypet vetëm nëse `k=false`, gjë që plotësohet vetëm kur nuk kalohet nëpër degën e komandës `if` brenda unazës, në fund të së cilës bëhet `k=true`. Rezultati që shtypet në ekran, nëse paraqitet rasti në fjalë, p.sh., për `z=15`, do të duket si në *Fig.5.33*.

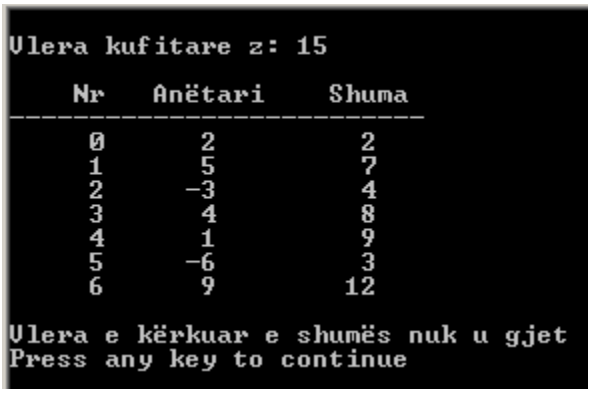


Fig.5.33
Rezultati i programit p4, për rastin kur nuk arrrihet që të jetë s>z

Siç shihet nga figura e dhënë, në fund të tabelës nuk janë vendosur vizat, sepse shtypja e vizave është paraparë vetëm nëse kalohet nëpër degën e komandës `if` që gjendet brenda unazës. Gjithashtu, në vend të dy mesazheve për vlerën e shumës dhe për numrin e anëtarëve që janë përfshirë në shumë, është shtypur një mesazh tjetër.

Nëse duam që edhe në rastin e dytë, në fund të tabelës të paraqitet rreshti me viza, komanda për shtypje, e cila është vendosur në degën e komandës `if` pas kllapës së fundit të unazës, duhet të shkruhet kështu:

```
if (k==false)
    cout << h
         << "\nVlera e kërkuar e shumës nuk u gjet"
         << endl;
```

Unazat while

Unazat të cilat realizohen duke e shfrytëzuar komandën `while`, ngjashëm si edhe ato që realizohen me komandën for, shkurt njihen si unaza `while`.

Në formë të përgjithshme unaza `while` duket kështu:

```
i=f;
while(i<=p)
{
      a;
      i=i+h;
}
```

ku janë:

`i` - variabla e unazës.
`f` - vlera fillestare e variablës së unazës.
`p` - vlera përfundimtare e variablës së unazës.
`h` - hapi me të cilin ndryshohen vlerat e variablës `i`.
`a` - komandat e përfshira brenda unazës.

Ekzekutimi i komandave `a` të përfshira brenda kllapave të unazës do të përsëritet derisa të plotësohet kushti `i<=p`.

Nëse unaza `while` paraqitet përmes bllok-diagramit, në formën e përgjithshme do ta kemi pamjen e dhënë në *Fig.5.34*.

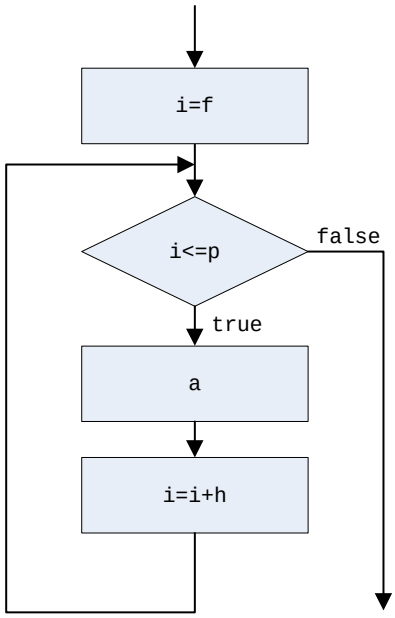


Fig.5.34
Forma e përgjithshme e unazës while e paraqitur përmes bllok-diagramit

Nga bllok-diagrami i dhënë shihet se unaza do të ekzekutohet për vlera të ndryshme të variablës `i`, gjersa vlera e kushtit `i<=p` është `true`. Në momentin kur kjo vlerë është `false`, ekzekutimi i unazës do të ndërpritet, përkatësisht do të dilet prej saj. Komandat `a`, të përfshira brenda unazës, nuk do të ekzekutohen asnjëherë, nëse menjëherë në fillim nuk plotësohet kushti `i<=p`.

Shembull Programi për llogaritjen e vlerës së faktorielit `F=(n+1)!` duke e realizuar unazën përmes komandës `while`.

```
// Programi while1
#include <iostream>
using namespace std;
int main()
{
      double F=1;
      int i,n;
      cout << "\nVlera e variablës n: ";
```

```
      cin >> n;
      i=1;
      while (i<=n+1)
      {
            F=F*i;
            i=i+1;
      }
      cout << "\nVlera e faktorielit F="
           << F
           << "\n\n";
      return 0;
}
```

Nga programi i dhënë shihet se kushti i ekzekutimit të unazës:

```
(i<=n+1)
```

është vendosur në fillim të saj, menjëherë pas komandës `while`. Në momentin kur ky kusht nuk plotësohet, llogaritja vazhdon me komandën e cila gjendet pas kllapës së fundit të trupit të komandës `while`.

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran duket si në *Fig.5.35*.

Fig.5.35
Rezultati i programit while1

```
Vlera e variablës n: 7
Vlera e faktorielit F=40320
Press any key to continue
```

Në program, në vend të shprehjes:

```
i=i+1;
```

për rritjen e vlerës së variablës `i` brenda unazës mund të shfrytëzohet edhe shprehja:

```
i++;
```

Nëse kushti i shënuar në fillim të unazës nuk plotësohet menjëherë në fillim, trupi i unazës `while` nuk do të ekzekutohet asnjëherë.

Programi i dhënë më sipër mund të realizohet edhe duke e shfrytëzuar unazën `for`, ashtu siç është dhënë në vijim.

```
// Programi while1a
```

```
#include <iostream>
using namespace std;
int main()
{
      double F=1;
      int i,n;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      for (i=1;i<=n+1;i++)
            F=F*i;

      cout << "\nVlera e faktorielit F="
           << F
           << "\n\n";
      return 0;
}
```

Prej këtu shihet qartë thjeshtimi i dukshëm i programit, sepse komanda `for` i përfshin në vete të gjithë parametrat e nevojshëm për unazën. Brenda kllapave të saj shënohet vlera fillestare e variablës së unazës (`i=1`), hapi me të cilin rritet (`i++`) si dhe kufiri (`i<=n+1`).

Vlera fillestare e variablës së unazës kompjuterit mund t'i jepet si vlerë hyrëse përmes tastierës.

Shembull

Programi për gjetjen e mbetjes nga pjesëtimi i dy numrave, duke e ndjekur procedurën e zbritjes sukcesive të pjesëtuesit nga i pjesëtuari.

```
// Programi Mbetja1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int k,x;
      char v[]="--------------------";
      cout << "\nNumri që pjesëtohet k: ";
      cin >> k;
      cout << "\nNumri me të cilin pjesëtohet x: ";
      cin >> x;
      cout << "\nMbetjet pas zbritjeve\n"
           << v
           << endl;
      while (k>=x)
```

```
      {
            k=k-x;
            cout << setw(10)
                 << k
                 << endl;
      }
      cout << v
           << "\nMbetja e pjesëtimit: "
           << k
           << endl;
return 0;
}
```

Në programin e dhënë, për ta gjetur mbetjen, e cila është rezultat i pjesëtimit jo të plotë të numrit `k` me numrin `x`, zbatohet procedura e zbritjes sukcesive të numrit `x` nga numri `k`. Gjatë kësaj, si vlerë fillestare e variablës së unazës është marrë numri `k`, e cila brenda trupit të unazës zvogëlohet me hapin `x`. Në këtë mënyrë, vlera të cilën e ka variabla `k` në momentin e daljes nga unaza, përkatësisht vlera e saj, kur nuk plotësohet kushti `k>=x`, është vlera e kërkuar e mbetjes nga pjesëtimi.

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse `k=27` dhe `x=5`, rezultati në ekran do të duket si në *Fig.5.36*.

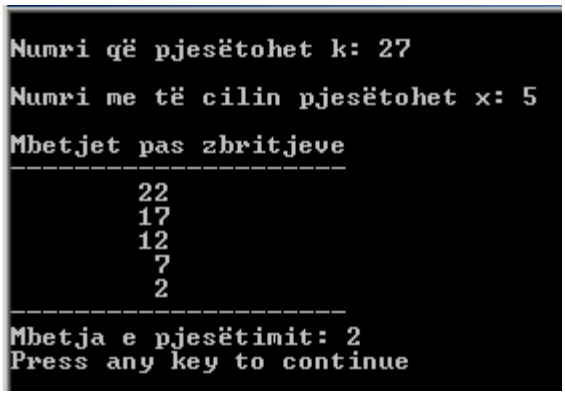


Fig.5.36
Rezultati i programit Mbetja1

Në programin `Mbetja1` trupi i unazës nuk do të ekzekutohet asnjëherë, nëse raporti i vlerave hyrëse është `k<x`.

Shembull

Programi për gjetjen e mbetjes nga pjesëtimi i dy numrave, i fituar me modifikimin e programit `Mbetja1`, ashtu që për gjetjen e mbetjes shfrytëzohet shprehja përkatëse matematikore.

```
// Programi Mbetja2
#include <iostream>
using namespace std;
int main()
{
      int m,k,x,a;
      cout << "\nNumri që pjesëtohet k: ";
      cin >> k;
      cout << "\nNumri me të cilin pjesëtohet x: ";
      cin >> x;
      a=k/x;
      cout << "\nVlera e pjesëtimit: "
           << a;
      m=k-a*x;
      cout << "\nMbetja e pjesëtimit: "
           << m
           << endl;
return 0;
}
```

Mbetja e pjesëtimit të numrit `k` me numrin `x` mund të gjendet edhe direkt, duke e shfrytëzuar në program vetëm shprehjen:

```
m=k-k/x*x;
```

Për ta gjetur rezultatin e kësaj shprehje, duhet të merret parasysh radha e ekzekutimit të operacioneve elementare aritmetikore. Me qëllim të shpjegimit më të thjeshtë, gjatë llogaritjes në fjalë, te programi `Mbetja2` shfrytëzohen dy shprehjet vijuese:

```
a=k/x;
m=k-a*x;
```

duke i llogaritur variablat përkatëse si variabla të tipit `int`. Kështu, për shembullin e vlerave hyrëse `k=27` dhe `x=5` të marra më sipër, do të kemi:

```
1.   a=k/x=27/5=5
2.   m=k-a*x=27-5*5=2
```

Gjatë kësaj, te variabla `a` është ruajtur vetëm vlera e plotë e pjesëtimit, meqë, siç thamë, kemi të bëjmë me variabël të tipit `int`.

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse të përmendura më sipër, rezultati do të duket si në *Fig.5.37*.

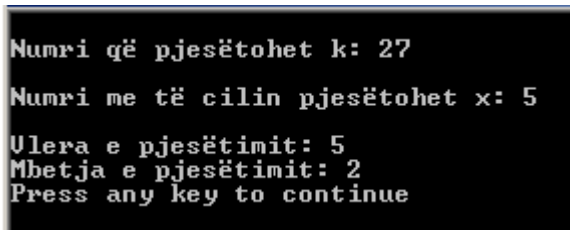


Fig.5.37
Rezultati i programit Mbetja2

Trupi i unazës që realizohet me komandën `while` mund të përmbajë komanda të çfarëdoshme.

Shembull Programi përmes së cilit shtypet tabela e katrorëve të numrave natyrorë çift mes `2` dhe `m`, nëse vlera e variablës `m` kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi p10
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,m;
      char T[]="----------------------";
      cout << "\nVlera kufitare m: ";
      cin >> m;
      cout << "\n    Numri   Katrori\n"
           << T
           << endl;
      i=2;
      while (i<=m)
      {
            cout << setw(7)
                 << i
                 << setw(10)
                 << i*i
                 << endl;
            i=i+2;
      }
      cout << T
           << endl;
return 0;
}
```

Këtu, vlera fillestare e variablës së unazës është `i=2`. Kjo vlerë rritet për 2 përmes komandës:

```
      i=i+2;
```

Trupi i unazës do të ekzekutohet vetëm nëse plotësohet kushti `i<=m`, i cili është shënuar në fillim të saj:

```
while (i<=m)
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do ta ketë pamjen e dhënë në *Fig.5.38*.

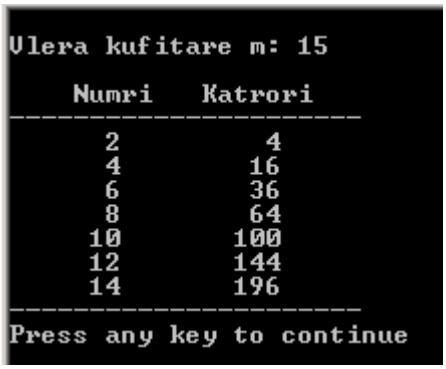


Fig.5.38
Rezultati i programit p10

Brenda trupit të unazës `while` mund të paraqiten edhe degëzime.

Shembull Programi përmes së cilit gjendet vlera më e madhe absolute në vektorin e dhënë `R(m)`.

```
// Programi while2
#include <iostream>
using namespace std;
int main()
{
      const m=7;
      int i,x,A[m]={4,7,-2,-9,1,8,3};
      x=abs(A[0]);
      i=1;
      while (i<m)
      {
            if (abs(A[i])>x)
               x=abs(A[i]);
            i++;
      }

      cout << "Vlera më e madhe absolute x="
```

```
            << x
            << "\n";

      return 0;
}
```

Në program, përmes unazës `while` krahasohen me radhë të gjithë anëtarët e vektorit dhe te variabla `x` ruhet vlera absolute më e madhe. Në fund, duke i pasur parasysh vlerat e anëtarëve të vektorit, i cili është deklaruar në fillim të programit, rezultati që shtypet në ekran është:

```
Vlera më e madhe absolute x=9
```

Variabla e unazës e tipit karakter

Variabla e unazës e cila realizohet përmes komandës `while` mund të jetë edhe variabël e tipit karakter.

Shembull Programi përmes së cilit në ekran shtypen shkronjat e alfabetit ndërkombëtar.

```
// Programi p11
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      char x;
      x='A';
      while (x<='Z')
      {
            cout << setw(2)
                 << x;
            x++;
      }
      cout << endl;
return 0;
}
```

Rezultati që shtypet në ekran pas ekzekutimit të programit të dhënë do të duket si në *Fig.5.39*.

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Press any key to continue
```

Fig.5.39 Rezultati i programit p11

Me qëllim të shtypjes së shkronjave në ekran me një zhvendosje të caktuar, p.sh., programit mund t'i shtohet edhe numëratori `i`, për ta shfrytëzuar atë brenda kllapave të manipulatorit `setw`, ashtu siç shihet në vijim.

```
// Programi p12
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      char x;
      int i;
      i=1;
      x='A';
      while (x<='Z')
      {
            cout << setw(i)
                 << x
                 << endl;
            x++;
            i++;
      }
      cout << endl;
return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati do të duket si në *Fig.5.40*.

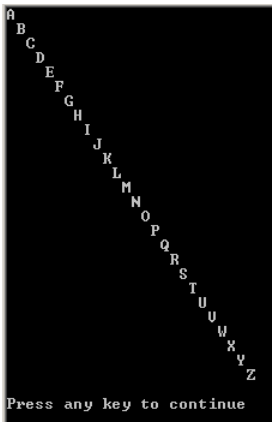


Fig.5.40
Rezultati i programit p12

Efekti i njëjtë i zhvendosjes së shkronjave gjatë shtypjes së tyre në ekran mund të arrihet edhe nëse variabla `i` rritet gjatë shfrytëzimit te manipulatori `setw`, ashtu siç shihet në versionin e programit që është dhënë në vijim.

```
// Programi p13
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      char x;
      int i;
      i=1;
      x='A';
      while (x<='Z')
      {
            cout << setw(i++)
                 << x
                   << endl;
            x++;
      }
      cout << endl;
return 0;
}
```

Nëse në versionin e fundit të programit ndryshohet hapi i rritjes së variablës së unazës për `2`, përkatësisht nëse shfrytëzohet shprehja:

```
      x=x+2;
```

në ekran shkronjat do të shtypen duke e kapërcyer çdo të dytën, ashtu siç shihet në *Fig.5.41*.

Fig.5.41
Rezultati i programit p13, nëse variabla e unazës rritet për 2

Unaza të ndërthurura

Brenda unazës `while` mund të përfshihen edhe unaza të tjera, duke formuar unaza të ndërthurura. Gjatë kësaj, si edhe tek unazat që formohen me

shfrytëzimin e komandës `for`, duhet pasur kujdes që unazat të mos priten mes vete.

Shembull Programi përmes së cilit gjenden numrat e përsosur në grumbullin e numrave natyrorë mes `1` dhe `n`.

```
// Programi persosur
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,j,k,n,s;
      char T[]="------------------";
      cout << "Vlera e numrit n: ";
      cin >> n;
      cout << "\nNumrat e përsosur"
           << endl
           << T
           << endl;
      i=2;
      while (i<=n)
      {
            s=1;
            j=2;
            k=i/2;
            while (j<=k)
            {
                  if ((i%j)==0)
                        s=s+j;
                  j=j+1;
            }
            if (s==i)
                  cout << "    "
                       << i
                       << endl;
            i=i+1;
      }
      cout << T
           << endl;
return 0;
}
```

Për një numër thuhet se është i përsosur, nëse shuma e numrave me të cilët ai plotpjesëtohet është e barabartë me vetë numrin. Kështu, p.sh., për numrin `28` thuhet se është i përsosur, sepse:

```
28=1+2+4+7+14
```

ku numrat që paraqiten në anën e djathtë të barazimit janë numrat me të cilët plotpjesëtohet numri `28`.

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse për variablën `n` kompjuterit i jepet numri `10000`, rezultati që shtypet në ekran do të duket si në *Fig.5.42*.

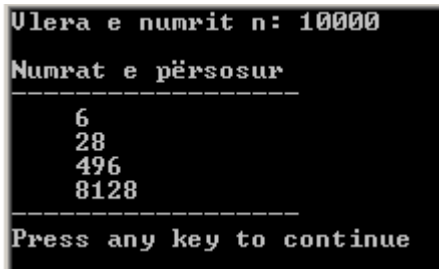


Fig.5.42
Rezultati i programit persosur

Në program, përmes unazës së jashtme e cila realizohet me komandën:

```
while (i<=n)
```

zgjedhen numrat e veçantë për t'u verifikuar se a janë numra të përsosur. Kurse, përmes unazës së brendshme:

```
while (j<=k)
{
      if ((i%j)==0)
            s=s+j;
      j=j+1;
}
```

llogaritet shuma e numrave me të cilët numri `i` plotpjesëtohet. Gjatë kësaj, numri `i` plotpjesëtohet me numrin `j`, nëse moduli i pjesëtimit, përkatësisht mbetja, nga pjesëtimi i këtyre dy numrave është zero. Për ta kontrolluar plotpjesëtimin e tyre, shfrytëzohet operatori `%`, i cili e jep si rezultat mbetjen nga pjesëtimi i numrit `i` me numrin `j`.

Pasi të përfundojë llogaritja e shumës, menjëherë pas daljes nga unaza, përmes kontrollës:

```
if (s==i)
```

shtypen numrat e përsosur në ekran.

Brenda një programi mund të paraqiten njëkohësisht edhe më shumë unaza.

Shembull Programi përmes së cilit shtypen numrat treshifrorë të Amstrongut.

```
// Programi Amstrong1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,x,y,z;
      char T[]="-------------------";
      cout << "\nNumrat e Amstrongut"
           << endl
           << T
           << endl;
      x=1;
      while (x<=9)
      {
            y=0;
            while (y<=9)
            {
                  z=0;
                  while (z<=9)
                  {
                        i=100*x+10*y+z;
                        if (i==(x*x*x+y*y*y+z*z*z))
                              cout << setw(10)
                                   << i
                                   << endl;
                        z=z+1;
                  }
                  y=y+1;
            }
            x=x+1;
      }
      cout << T
           << endl;
return 0;
```

```
}
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.5.43*.

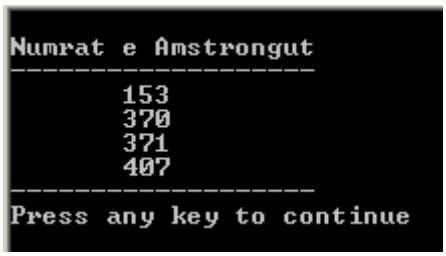


Fig.5.43
Rezultati i programit Amstrong1

Për një numër thuhet se paraqet numër të Amstrongut, nëse shuma e kubeve të shifrave të tij është e barabartë me vetë numrin. Kështu, p.sh., numri `153` bën pjesë në grupin e numrave në fjalë, meqë vlen:

$$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$$

Në program, për gjenerimin e të gjitha kombinimeve të mundshme të shifrave të numrave treshifrorë, janë shfrytëzuar `3` unaza të realizuara me komandën `while`.

Përmes shprehjes:

```
i=100*x+10*y+z;
```

e cila përfshihet në unazën e tretë (unazën e brendshme), gjendet numri i cili i përket kombinimit të `3` shifrave `x`, `y` dhe `z`, kurse shuma e kubeve të tyre llogaritet me shprehjen:

```
x*x*x+y*y*y+z*z*z
```

Në fund, përmes kontrollës:

```
if (i==(x*x*x+y*y*y+z*z*z))
```

gjenden dhe në degën përkatëse edhe shtypen numrat e kërkuar.

Për gjetje të numrave treshifrorë të Amstrongut mund të shfrytëzohet një version më i thjeshtë i programit, i cili është dhënë në vijim.

```
// Programi Amstrong2
#include <iostream>
#include <iomanip>
using namespace std;
```

```
int main()
{
      int i,x,y,z;
      char T[]="-------------------";
      cout << "\nNumrat e Amstrongut"
           << endl
           << T
           << endl;
      i=100;
      while (i<=999)
      {
            x=i/100;
            y=(i%100)/10;
            z=i%10;
            if (i==(x*x*x+y*y*y+z*z*z))
                  cout << setw(10)
                       << i
                       << endl;
            i=i+1;
      }
      cout << T
           << endl;
return 0;
}
```

Nëse ekzekutohet programi i dhënë, në ekran do të fitohet rezultati i njëjtë me atë që shihet në *Fig.5.43*. Në program, shifrat x, y dhe z nuk merren si variabla të tri unazave të veçanta, ashtu siç është vepruar te programi paraprak `Amstrong1`. Por, ato këtu përcaktohen nga vlerat e variablës `i` të unazës së vetme, përmes shprehjeve:

```
x=i/100;
y=(i%100)/10;
z=i%10;
```

dhe si rezultat japin vetëm numra të plotë.

Unazat gjatë operimit me fusha

Indekset e anëtarëve të vektorëve, të matricave ose të fushave shumëdimensionale, të cilat shfrytëzohen gjatë llogaritjeve të ndryshme, mund të përcaktohen përmes vlerave të variablave të unazave të formuara duke i shfrytëzuar komandat `while`.

Operimi me vektor

Gjatë operimit me anëtarët e vektorëve, meqë vektorët paraqesin fusha njëdimensionale, për zgjedhjen e indekseve të tyre mund të shfrytëzohet variabla e unazës e cila realizohet përmes komandës `while`.

Shembull

Programi përmes së cilit gjendet shuma e katrorëve të anëtarëve të vektorit `R(m)`, të cilët kanë indeks tekë. Njëkohësisht, anëtarët që shfrytëzohen gjatë llogaritjes së shumës shtypen në ekran.

```
// Programi p13
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=8;
      int i,R[m]={4,6,-7,9,-2,5,3,-1};
      double s;
      char g[]="----------------------------";
      cout << "\nAnëtarët e vektorit\n"
           << "\n Indeksi  Anëtari  Katrori\n"
           << g
           << endl;
      s=0;
      i=1;
      while (i<m)
      {
            s=s+R[i]*R[i];
            cout << setw(5)
                 << i
                 << setw(9)
                 << R[i]
                 << setw(10)
                 << R[i]*R[i]
                 << endl;
            i=i+2;
      }
      cout << g
           << "\nShuma e llogaritur s="
           << s
```

```
            << endl;
return 0;
}
```

Këtu, variabla `i` e unazës shfrytëzohet si indeks i anëtarëve të vektorit. Vlera fillestare e kësaj variable është marrë `i=1`, kurse brenda unazës rritet me hapin `2`, ashtu që shumës i shtohen vetëm katrorët e anëtarëve me indeks tekë.

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.5.44*.

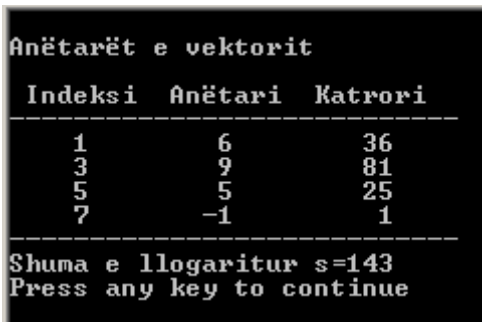


Fig.5.44
Rezultati i programit p13

Në kolonën e parë të tabelës janë dhënë indekset e anëtarëve të vektorit, në kolonën e dytë - vlerat e anëtarëve, kurse në kolonën e tretë - katrorët që përfshihen në shumë.

Brenda unazës `while` mund të paraqiten më shumë komanda të ndryshme.

Shembull

Programi përmes së cilit nga anëtarët e vektorit `F(n)` formohet vektori i ri `R`, duke i shfrytëzuar vetëm anëtarët e vektorit të cilët janë më të mëdhenj se `4` dhe më të vegjël se `11`.

```
// Programi p14
#include <iostream>
#include <iomanip>
using namespace std;
```

```
int main()
{
      const n=10;
      int i,k,R[n],F[n]={6,-4,8,7,12,9,15,-3,5,10};
      char g[]="-----------------";
      cout << "\nAnëtarët që përfshihen në vektor\n"
           << "\n    i      F[i]\n"
           << g
           << endl;
      k=-1;
      i=0;
      while (i<n)
      {
            if ((F[i]>4) && (F[i]<11))
            {
                  k=k+1;
                  R[k]=F[i];
            cout << setw(5)
                 << i
                 << setw(9)
                 << F[i]
                 << endl;
            }
            i++;
      }
      cout << g
           << "\n\nVektori i formuar \n"
           << "\nR={";
      for (i=0;i<=k;i++)
            cout << setw(3)
                 << R[i];
      cout << "  }\n\n";
return 0;
}
```

Këtu, brenda unazës `while`, paraqitet komanda për degëzim:

```
if ((F[i]>4) && (F[i]<11))
```

me një degë të përbërë, përmes së cilës zgjedhen anëtarët e vektorit të cilët janë më të mëdhenj se 4 dhe më të vegjël se 11. Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket si në *Fig.5.45*.

Fig.5.45
Rezultati i programit p14

Në pjesën e parë të rezultatit janë dhënë anëtarët që përfshihen në vektorin që formohet si dhe indekset përkatëse në vektorin `F(m)`. Kurse në fund është shtypur edhe vektori i formuar `R`, duke e shfrytëzuar unazën e realizuar me komandën `for`. Indekset e anëtarëve të vektorit të formuar `R` shkojnë mes vlerave `0` dhe `k`, përkatësisht ky vektor ka gjithsej `k+1` anëtarë.

Shembull Programi përmes së cilit radhiten sipas madhësisë anëtarët e vektorit `Z(m)`, duke filluar prej anëtarit me vlerë më të madhe.

```
// Programi sort1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=12;
      int i,j,k,Z[m]={4,7,9,-3,5,12,6,17,-2,13,3,8};
      i=0;
      while (i<m)
      {
            for (j=i+1;j<m;j++)
                  if (Z[i]<=Z[j])
                  {
                        k=Z[i];
                        Z[i]=Z[j];
                        Z[j]=k;
                  }
            i++;
      }
      cout << "\nVektori i sortuar \n"
           << "\nZ={";
```

```
      for (i=0;i<m;i++)
            cout << setw(3)
                 << Z[i];
      cout << "  }\n\n";
return 0;
}
```

Procedura e sortimit, e cila është shfrytëzuar në program, mbështetet në metodën e sortimit të zakonshëm, sipas së cilës në çdo hap `i` gjendet anëtari më i madh dhe kjo vlerë fiksohet tek anëtari `Z[i]` i vektorit. Brenda trupit të unazës `while` këtu është shfrytëzuar unaza e realizuar me komandën `for`:

```
      for (j=i+1;j<m;j++)
            if (Z[i]<=Z[j])
            {
                  k=Z[i];
                  Z[i]=Z[j];
                  Z[j]=k;
            }
```

përmes së cilës, nëse plotësohet kushti i shkruar në vazhdim të komandës `if`, u ndërrohen vendet anëtarëve `Z[i]` dhe `Z[j]` të vektorit, me ndërmjetësimin e variablës ndihmëse `k`.

Në fund të programit, për shtypjen e vektorit të sortuar `Z(m)` është shfrytëzuar një unazë e veçantë `for`. Rezultati që shtypet në ekran përmes kësaj unaze do të duket si në *Fig.5.46*, ku, siç shihet, anëtarët e vektorit janë sortuar duke filluar me anëtarin me vlerë më të madhe.

Fig.5.46 Rezultati i programit sort1

Operimi me matrica

Meqë matricat paraqesin fusha dydimensionale, gjatë shfrytëzimit të anëtarëve të tyre idekset përkatëse mund të përcaktohen përmes variablave të dy unazave të realizuara përmes komandës `while`.

Shembull

Programi përmes së cilit gjendet shuma e anëtarëve të matricës `A(m,m)`, që gjenden mbi diagonalen kryesore.

```
// Programi while3
#include <iostream>
using namespace std;
int main()
{
      const m=5;
      int A[m][m]={{4,2,7,5,3},
                   {1,9,7,2,6},
                   {3,8,5,9,2},
                   {5,1,6,8,3},
                   {7,3,8,1,5}};
      double s;
      int i,j;
      s=0;
      i=0;
      while (i<m)
      {
            j=0;
            while (j<m)
            {
              if (i<j)
                  s=s+A[i][j];
              j++;
            }
      i++;
      }
      cout << "Shuma e kërkuar është: "
           << s
           << "\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati i cili shtypet në ekran do të duket kështu:

```
Shuma e kërkuar është: 46
```

ku vlera e shumës llogaritet me mbledhjen e anëtarëve mbi diagonalen kryesore të matricës, kështu:

```
s = 2 + 7 + 5 + 3
      + 7 + 2 + 6
          + 9 + 2
              + 3 = 46
```

Unaza e parë (e jashtme) e realizuar përmes komandës:

```
while (i<m)
```

këtu është shfrytëzuar për zgjedhjen e rreshtave të matricës (vlerat e indeksit `i`), kurse përmes unazës së dytë (të brendshme):

```
while (j<m)
```

zgjedhen kolonat e matricës (vlerat e indeksit të dytë `j`). Në unazën e jashtme `while` përfshihen komandat të cilat gjenden brenda kllapave të jashtme, kurse kllapat e brendshme e përcaktojnë trupin e komandave të unazës së brendshme `while`.

Edhe këtu, sikurse edhe tek unazat e ndërthurura të realizuara përmes komandave `for`, për një vlerë të variablës së unazës së jashtme (indeksit të parë `i`), merren të gjitha vlerat e variablës së unazës së brendshme (indeksit të dytë `j`).

Në programin e mësipërm, njëra ose të dy unazat mund të realizohen edhe duke e shfrytëzuar komandën `for`.

Shembull Programi `while3` tek i cili shfrytëzohet komanda `for` për realizimin e unazës së brendshme.

```
// Programi while4
#include <iostream>
using namespace std;
int main()
{
      const m=5;
      int A[m][m]={{4,2,7,5,3},
                   {1,9,7,2,6},
                   {3,8,5,9,2},
                   {5,1,6,8,3},
                   {7,3,8,1,5}};
      double s;
      int i,j;
      s=0;
      i=0;
      while (i<m)
      {
            for (j=0;j<m;j++)
               if (i<j)
                   s=s+A[i][j];
      i++;
      }
```

```
      cout << "Shuma e kërkuar është: "
           << s
           << "\n";
return 0;
}
```

Këtu, në unazën e brendshme përfshihen komandat:

```
     for (j=0;j<m;j++)
         if (i<j)
             s=s+A[i][j];
```

kurse në trupin e unazës së jashtme që realizohet përmes komandës `while`, përfshihet edhe komanda për rritje për `1` të vlerës së variablës përkatëse të kësaj unaze:

```
      i++;
```

Programi `while4` mund të thjeshtohet, duke i marrë vetëm vlerat e variablave `i` dhe `j`, të cilat u përkasin indekseve të anëtarëve mbi diagonalen kryesore të matricës. Si rezultat, pjesa e programit përmes së cilës përcaktohen unazat e nevojshme do të duket kështu:

```
      i=0;
      while (i<m)
      {
            for (j=i+1;j<m;j++)
                 s=s+A[i][j];
            i++;
      }
```

Prej këtu shihet se për çdo vlerë të indeksit `i`, të rreshtave të matricës, fillohet me vlerën `j=i+1` të indeksit të kolonave të matricës. Kjo, sepse indeksi i dytë i anëtarit mbi dijagonalën kryesore të matricës është për `1` më i madh se indeksi i rreshtit përkatës.

Shembull Programi përmes së cilit formohet vektori `R` nga anëtarët negativë të matricës së dhënë `F(m,n)`.

```
// Programi p15
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      const m=5,n=4;
      int F[m][n]={{7,-3,14,6},
```

```
                  {9,-2,-4,8},
                  {15,4,6,-7},
                  {-1,17,3,5},
                  {5,-8,-5,2}};
      int i,j,k,R[m*n];
      char v[]="--------------------";
      k=-1;
      i=0;
      while (i<m)
      {
            j=0;
            while (j<n)
            {
                  if (F[i][j]<0)
                  {
                        k=k+1;
                        R[k]=F[i][j];
                  }
                  j++;
            }
            i++;
      }
      cout << "\nVektori i formuar R\n"
           << "\n Indeksi   Anëtari\n"
           << v
           << endl;
      for (i=0;i<=k;i++)
            cout << setw(5)
                 << i
                 << setw(10)
                 << R[i]
                 << endl;
      cout << v
           << endl;
return 0;
}
```

Në program, për përcaktimin e indekseve `i` dhe `j` të anëtarëve të matricës janë shfrytëzuar dy unaza `while`. Në trupin e unazën së brendshme është përfshirë komanda `if`:

```
      if (F[i][j]<0)
      {
            k=k+1;
            R[k]=F[i][j];
      }
```

në degën e së cilës përcaktohet indeksi i anëtarit që i shtohet vektorit:

```
k=k+1;
```

si dhe anëtari përkatës vendoset në vektor:

```
R[k]=F[i][j];
```

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket si në *Fig.5.47*.

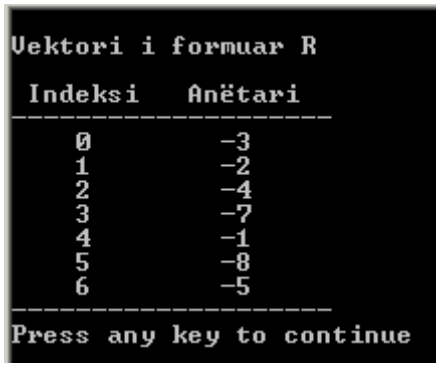


Fig.5.47
Rezultati i programit p15

Brenda unazave mund të paraqiten degëzime të ndryshme përmes komandës `if`.

Shembull

Programi përmes së cilit gjendet shuma e anëtarëve të matricës `A(m,n)`, të cilët për nga vlera absolute janë më të mëdhenj se `4` dhe më të vegjël se numri `10`. Njëkohësisht, numërohen anëtarët e matricës të cilët janë përfshirë gjatë llogaritjes së shumës.

```
// Programi p16
#include <iostream>
using namespace std;
int main()
{
```

```
      const m=4,n=5;
      int F[m][n]={{-7,3,2,15,4},
                   {6,-1,-9,5,-2},
                   {18,7,11,9,6},
                   {-5,8,-4,6,1}};
      int i,j,k;
      float s;
      s=0;
      k=0;
      i=0;
      while (i<m)
      {
            j=0;
            while (j<n)
            {
                  if (abs(F[i][j])>4)
                        if (abs(F[i][j])<10)
                  {
                              s=s+F[i][j];
                              k=k+1;
                  }
                  j++;
            }
            i++;
      }
      cout << "\nShuma e kërkuar e anëtarëve të matricës s="
           << s
           << endl
           << "\nNumri i anëtarëve të përfshirë në shumë k="
           << k
           << "\n\n";
return 0;
}
```

Në unazën e brendshme të programit të dhënë janë përfshirë dy komanda `if` për degëzim:

```
      if (abs(F[i][j])>4)
            if (abs(F[i][j])<10)
      {
                  s=s+F[i][j];
                  k=k+1;
      }
```

Nëse plotësohen dy kushtet e shtruara, në bazë të të cilëve vlerat e anëtarëve të matricës kontrollohen se a janë më të mëdhenj se 4 dhe më të vegjël

se 10, shumës i shtohet anëtari në fjalë. Njëkohësisht, numëratori k, përmes së cilit numërohen anëtarët e përfshirë në shumë, rritet për 1.

Pas ekzekutimit të programit të dhënë, rezultati do të duket si në *Fig.5.48*.

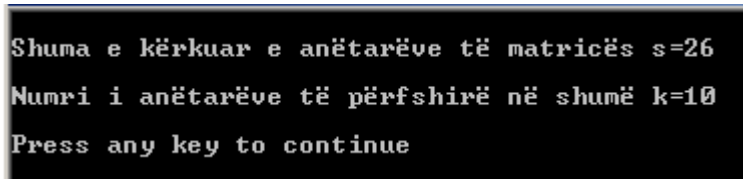


Fig.5.48 Rezultati i programit p16

Gjatë zgjedhjes së indekseve të anëtarëve të matricës, unazat mund të realizohen edhe duke e shfrytëzuar komandën for. Kështu, p.sh., te shembulli i programit p16, unaza e jashtme realizohet përmes komandës for, pjesa e trupit të programit ku janë vendosur unazat do të duket si në vijim.

```
s=0;
k=0;
for (i=0;i<m;i++)
{
      j=0;
      while (j<n)
      {
            if (abs(F[i][j])>4)
                  if (abs(F[i][j])<10)
            {
                        s=s+F[i][j];
                        k=k+1;
            }
            j++;
      }
}
```

Këtu, me ngjyrë të zezë janë theksuar unazat e realizuara përmes komandës while. Kurse, nëse shfrytëzohet komanda for për realizimin e unazës së brendshme, pjesa e unazave në trupin e programit do ta ketë pamjen vijuese, ku me ngjyrë të zezë janë theksuar komandat e përfshira në unazën for.

```
s=0;
k=0;
i=0;
while (i<m)
{
```

```
            for (j=0;j<n;j++)
            {
                  if (abs(F[i][j])>4)
                        if (abs(F[i][j])<10)
                  {
                              s=s+F[i][j];
                              k=k+1;
                  }
            }
            i++;
      }
```

Kapërcimi i komandave brenda unazës

Pjesë të caktuara brenda unazave `while` mund të kapërcehen duke e shfrytëzuar komandën `goto`.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y = \frac{x}{3} + 4 \sum_{\substack{i=1 \\ (i \neq 3,6)}}^{n+1} (2i+1)$$

Vlerat e variablave `x` dhe `n` kompjuterit i jepen si vlera hyrëse përmes tastierës.

```
// Programi p24
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int i,n;
      double s,x,y;
      char v[]="---------------";
      cout << "Vlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >>n;
      cout << "\n    i      s\n"
           << v
           << endl;
      s=0;
      i=1;
      while (i<=(n+1))
```

```
      {
            if ((i==3) || (i==6))
                  goto Vijues;
            s=s+(2*i+1);
            cout << setw(5)
                 << i
                 << setw(7)
                 << s
                 << endl;
Vijues:
            i++;
      }
      y=x/3+4*s;
      cout << v
           << "\nVlera e funksionit y="
           << y
           << endl;
return 0;
}
```

Këtu, brenda unazës `while`, kapërcimi i shtimit të anëtarëve të shumës për vlerat `3` dhe `6` të variablës `i` të unazës është realizuar përmes komandës:

```
goto Vijues;
```

Pas ekzekutimit të programit të dhënë, për vlera të caktuara hyrëse të variablave `x` dhe `n`, rezultati do të duket si në *Fig.5.49*.

```
Vlera e variablës x: 6.58

Vlera e variablës n: 7

    i      s
---------------
    1      3
    2      8
    4     17
    5     28
    7     43
    8     60
---------------
Vlera e funksionit y=242.193
Press any key to continue
```

Fig.5.49
Rezultati i programit p24

Tek unaza `while`, me qëllim të kapërcimit të një pjese të caktuar brenda saj, nuk mund të shfrytëzohet komanda `continue`. Kjo, sepse përmes saj kapërcehet edhe komanda për rritje të vlerës së variablës së unazës. Si rrjedhojë, hyhet në një unazë të pafund dhe programi mund të ndërpritet vetëm me forcë, p.sh., përmes kombinimit të tasteve `Ctrl+Break` në tastierë.

Dalja nga unaza

Për dalje nga unazat e realizuara me komandën `while`, mund të shfrytëzohen komandat `goto` dhe `break`.

Dalja përmes komandës goto

Shfrytëzimi i komandës për kapërcim pa kusht goto, për dalje nga unaza e cila është realizuar me komandën `while`, nuk dallon aspak nga ai që është shpjeguar tek unazat të realizuara me komandën `for`.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y = \sum_{i=1}^{\infty} \left\{ \frac{1}{i^3} \right\}$$

duke e ndërprerë llogaritjen kur anëtari që i shtohet shumës është më i vogël se numri i dhënë `e`.

```
// Programi p1
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main()
{
      const double e=0.001;
      int i;
      double a,y;
      char V[]="------------------------------";
      cout << "\n      i      a          y\n"
           << V
           << endl;
      cout << fixed
           << setprecision(5);
      y=0;
      i=1;
```

```
      while (i<=9999)
      {
            a=1/pow(i,3);
            if (a<e)
                  goto Dalja;
            y=y+a;
            cout << setw(7)
                 << i
                 << setw(10)
                 << a
                 << setw(11)
                 << y
                 << endl;
            i++;
      }
Dalja:
      cout << V
           << endl;
      cout << "Vlera e funksionit y="
           << y
           << "\n";
return 0;
}
```

Në program është paraparë që trupi i unazës së realizuar përmes komandës `while` të përsëritet `9999` herë (ky numër është marrë lirisht shumë i madh). Por, përsëritja e ekzekutimit të komandave të përfshira në unazë ndërpritet nëse vlera e anëtarit `a` të serisë është shumë e vogël, përkatësisht nëse kjo vlerë është më e vogël se numri `e`. Për këtë qëllim është paraparë dalja nga unaza duke i shfrytëzuar komandat:

```
            if (a<e)
                  goto Dalja;
```

me të cilat kapërcehet në pjesën e programit ku është vendosur labela `Dalja:`.

Rezultati që fitohet në ekran pas ekzekutimit të programit të dhënë më sipër do të duket si në *Fig.5.50*.

```
      i          a           y
------------------------------
      1    1.00000     1.00000
      2    0.12500     1.12500
      3    0.03704     1.16204
      4    0.01563     1.17766
      5    0.00800     1.18566
      6    0.00463     1.19029
      7    0.00292     1.19321
      8    0.00195     1.19516
      9    0.00137     1.19653
     10    0.00100     1.19753
------------------------------
Vlera e funksionit y=1.19753
Press any key to continue
```

Fig.5.50
Rezultati i programit p1

Siç shihet nga tabela e rezultatit, në të janë përfshirë edhe vlerat e anëtarëve `a` të serisë, për të parë se si ato zvogëlohen me rritjen e vlerës së variablës `i`. Vlera e fundit e këtij anëtari është e barabartë me vlerën e variablës `e`, e cila shfrytëzohet në përcaktimin e saktësisë së llogaritjes së vlerës së funksionit `y`.

Në program, përmes komandës:

```
cout << fixed
     << setprecision(5);
```

kompjuterit i urdhërohet që, gjatë shtypjes, vlerat e variablave `a` dhe `y` të shtypen në formë dhjetore (manipulatori `fixed`) si dhe me precizitet prej `5` vendesh pas pikës dhjetore (manipulatori `setprecision`).

Programi i dhënë më sipër mund të realizohet edhe pa e shfrytëzuar komandën `goto` për dalje nga unaza.

```
// Programi p2
#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;
int main()
{
      const double e=0.001;
      int i;
      double a,y;
      char V[]="------------------------------";
      cout << "\n      i      a          y\n"
           << V
           << endl;
      cout << fixed
           << setprecision(5);
      y=0;
      a=1;
      i=1;
      while (a>e)
      {
            a=1/pow(i,3);
            y=y+a;
            cout << setw(7)
```

```
                    << i
                    << setw(10)
                    << a
                    << setw(11)
                    << y
                    << endl;
               i++;
        }

        cout << V
             << endl;
        cout << "Vlera e funksionit y="
             << y
             << "\n";
return 0;
}
```

Në programin e dhënë, dalja nga unaza përcaktohet përmes komandës:

```
while (a>e)
```

e cila është vendosur menjëherë në fillim të saj. Por, që ky testim të mund të shfrytëzohet edhe në hapin e parë të llogaritjes, për variablën `a` është marrë vlera fillestare `1` (sepse atë vlerë e ka për i=1), menjëherë para testimit në fjalë. Rezultati që fitohet pas ekzekutimit të programit `p2` do të jetë plotësisht i njëjtë me atë të programit `p1`, i cili u dha më sipër në *Fig.5.50*.

Dalja përmes komandës break

Ngjashëm me unazat e realizuara përmes komandës `for`, nëse brenda unazës `while` vendoset komanda `break`, pa kusht dilet prej saj dhe ekzekutimi i programit vazhdon me komandën e cila gjendet menjëherë pas kllapës së fundit të trupit të unazës.

Shembull

Programi përmes së cilit llogariten vlerat e faktorielëve të numrave natyrorë mes 1 dhe n, duke ia dhënë kompjuterit vlerën e variablës `n` përmes tastierës. Në program është paraparë që llogaritja të ndërpritet, nëse faktorieli e tejkalon vlerën maksimale të mundshme të tipit `int`.

```
// Programi p25
#include <iostream>
#include <iomanip>
#include <climits>
using namespace std;
```

```
int main()
{
      int i,n;
      float F;
      char h[]="-----------------------------";
      cout << "Vlera e variablës n: ";
      cin >> n;
      cout << "\n  Numri      Faktorieli\n"
           << h
             << endl;
      F=1;
      i=1;
      while (i<=n)
      {
            if ((F*i)>INT_MAX)
            {
                  cout << "\nVlera e faktorielit e madhe\n";
                  break;
            }
            F=F*i;
            cout << setw(5)
                 << i
                 << fixed
                 << setprecision(1)
                 << setw(18)
                 << F
                 << endl;
            i++;
      }
      cout << h
           << endl;
return 0;
}
```

Në program, për llogaritje të faktorielit është shfrytëzuar unaza `while`, ku, siç shihet, para unazës janë përcaktuar vlerat fillestare, si ajo e faktorielit (`F=1`) ashtu edhe ajo e variablës së unazës (`i=1`). Brenda unazës, përmes shprehjes:

```
F=F*i;
```

kryhen shumëzimet e nevojshme për llogaritjen e vlerës së faktorielit, të cilat pastaj shtypen duke e shfrytëzuar komandën `cout`. Por, para se të llogaritet vlera vijuese e faktorielit, për vlerën aktuale të variablës së unazës, përmes komandës:

```
if ((F*i)>INT_MAX)
{
```

```
            cout << "\nVlera e faktorielit e madhe\n";
            break;
      }
```

kontrollohet se mos ndoshta kjo vlerë e tejkalon vlerën kufitare maksimale të tipit `int`, duke e krahasuar atë me konstanten `INT_MAX`, e cila e paraqet vlerën maksimale të mundshme. Nëse plotësohet kushti i shkruar brenda kllapave të komandës `if`, në vazhdim të komandës është paraparë shtypja e mesazhit:

```
Vlera e faktorielit e madhe
```

dhe, njëkohësisht, përmes komandës `break` edhe dalja nga unaza. Komanda e parë e cila ekzekutohet pas daljes nga unaza është komanda:

```
cout << h
     << endl;
```

me të cilën shtypen vizat në fund të tabelës me vlerat e faktorielëve, e cila me atë rast përpilohet. Kështu, p.sh., nëse programi i dhënë ekzekutohet dhe përmes tastierës kompjuterit i jepet vlera `n=50`, tabela do ta ketë pamjen e dhënë në *Fig.5.51*.

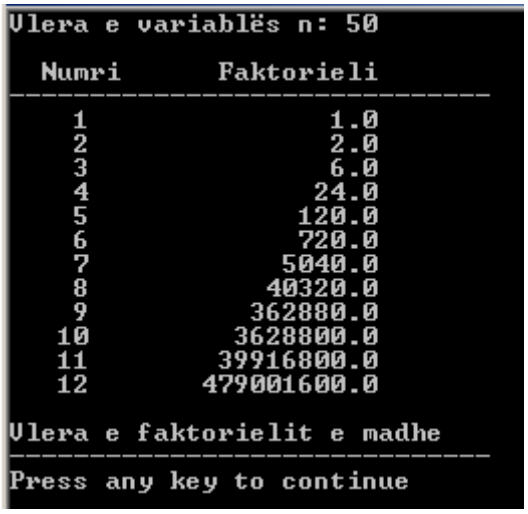


Fig.5.51
Rezultati i programit p25

Nga tabela e dhënë shihet se pas vlerës së faktorielit për `i=12` plotësohet kushti për tejkalimin e vlerës maksimale të mundshme të konstantes `INT_MAX` dhe prandaj ekzekutimi i mëtejmë i unazës ndërpritet.

Siç është shpjeguar edhe në kapitullin e dytë, gjatë shfrytëzimit të konstanteve ku ruhen vlerat kufitare të tipeve të ndryshme, në ballinën e programit duhet të vendoset komanda paraprocesorike:

```
#include <climits>
```

Nëse brenda unazës së programit të dhënë nuk vendoset komanda `if`, ekzekutimi i unazës do të vazhdojë edhe më tutje, sepse variabla `F` është deklaruar e tipit `float` dhe në të mund të ruhen vlera më të mëdha se vlera e konstantes `INT_MAX`.

Unazat do-while

Përveç me komandat `for` dhe `while`, unazat mund të realizohen edhe duke i shfrytëzuar komandat `do` e `while`, të cilat në formë të përgjithshme shkruhen kështu:

```
i=f;
do
{
   a;
   i=i+h;
}
while(i<=p);
```

ku janë:

`i` - variabla e unazës.
`f` - vlera fillestare e variablës së unazës.
p - vlera përfundimtare e variablës së unazës.
`h` - hapi me të cilin ndryshohen vlerat e variablës `i`.
`a` - komandat e përfshira brenda unazës.

Unazat e tilla shkurt quhen edhe unaza `do-while`. Përmes tyre, ekzekutimi i komandave `a`, të përfshira brenda kllapave të unazës, do të përsëritet derisa të plotësohet kushti `i<=p`.

Nëse unaza `do-while` paraqitet përmes bllok-diagramit, në formë të përgjithshme do ta kemi pamjen e dhënë në *Fig.5.52*.

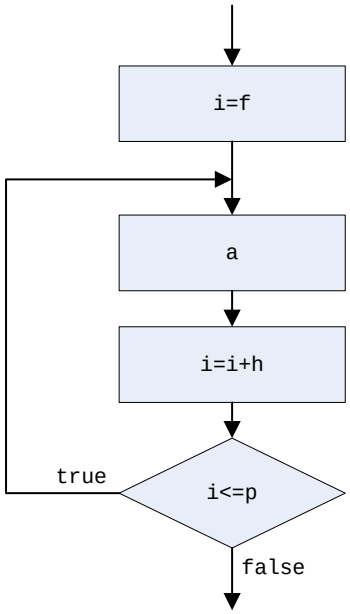


Fig.5.52
Forma e përgjithshme e unazës do-while e paraqitur përmes bllok-diagramit

Për dallim nga unazat e realizuara me komandat `for` dhe `while`, tek unaza `do-while` së pari ekzekutohen komandat të cilat gjenden brenda unazës dhe pastaj kontrollohet se a është plotësuar kushti `i<=p`. Prandaj, duhet pasur kujdes gjatë shfrytëzimit të kësaj unaze, që të eliminohet mundësia e ekzekutimit të saj për vlerën fillestare `i=f`, në rastet kur kjo vlerë është e palogjikshme.

Shembull Programi përmes së cilit nga numrat natyrorë mes `1` dhe `m` llogaritet shuma e katrorëve të numrave tekë dhe e kubeve të numrave çift.

```
// Programi do1
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
      int i,m;
      double s;
      cout << "\nVlera e variablës m: ";
```

```
      cin >> m;
      s=0;
      i=1;
      do
      {
            s=s+pow(i,2)+pow(i+1,3);
            i=i+2;
      }
      while (i<=m);
      cout << "Shuma e kërkuar është s="
           << s
           << "\n";
return 0;
}
```

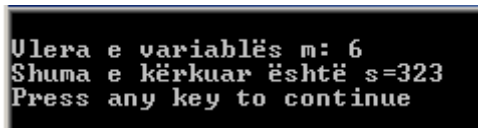
Në program, brenda unazës `do-while` është vendosur komanda:

```
s=s+pow(i,2)+pow(i+1,3);
```

e cila do të ekzekutohet për vlera të ndryshme të variablës `i` mes vlerës fillestare `i=1` dhe vlerës përfundimtare `m` (kushti `i<=m`), duke e rritur ate me hapin `2` (shprehja `i=i+2`). Në këtë shprehje, për llogaritjen e katrorëve dhe të kubeve të numrave `i` dhe `i+1` është shfrytëzuar funksioni `pow`, i cili gjendet në pakon e funksioneve të modulit `math.h`, për çka do të bëhet fjalë më vonë.

Pas ekzekutimit të programit të dhënë, për vlerën hyrëse `m=6`, e cila kompjuterit i jepet përmes tastierës, në ekran do ta kemi pamjen e dhënë në *Fig.5.53*.

Fig.5.53
Rezultati i programit do1



Shuma e fituar në fakt është llogaritur përmes shprehjes:

$$s = 1^2 + \mathbf{2^3} + 3^2 + \mathbf{4^3} + 5^2 + \mathbf{6^3} = 323$$

Programi i dhënë mund të realizohet duke e shfrytëzuar unazën `while`, ose unazën `for`. Pamja e tij gjatë realizimit përmes unazës `while` është dhënë në vijim.

```
// Programi do1a
#include <iostream>
#include <math.h>
```

```
using namespace std;

int main()
{
      int i,m;
      double s;
      cout << "\nVlera e variablës m: ";
      cin >> m;
      s=0;

      i=1;
      while (i<=m)
      {
            s=s+pow(i,2)+pow(i+1,3);
            i=i+2;
      }

      cout << "Shuma e kërkuar është s="
           << s
           << "\n";
return 0;
}
```

Nëse analizohet ky version i programit, do të shihet se dallimi qëndron vetëm në zhvendosjen e komandës `while` në pozitën e komandës `do`, duke mos ndryshuar asgjë tjetër.

Versioni i programit i realizuar me unazën `do-while` do të thjeshtohet krejtësisht, nëse në vend të saj shfrytëzohet unaza `for`, gjë që shihet në vijim.

```
// Programi do1b
#include <iostream>
#include <math.h>
using namespace std;

int main()
{
      int i,m;
      double s;
      cout << "\nVlera e variablës m: ";
      cin >> m;
      s=0;

      for (i=1;i<=m;i=i+2)
            s=s+pow(i,2)+pow(i+1,3);

      cout << "Shuma e kërkuar është s="
           << s
           << "\n";
```

```
return 0;
}
```

Brenda unazës `do-while` mund të paraqiten edhe komanda të tjera, siç është, p.sh., komanda për degëzim `if`.

Shembull Programi përmes së cilit llogaritet prodhimi i katrorëve të anëtarëve negativë të vektorit `A(m)`.

```
// Programi do2
#include <iostream>
using namespace std;
int main()
{
      const int m=8;
      int i,A[m]={4,-7,2,5,-3,8,-6,1};
      double p;
      p=1;

      i=0;
      do
      {
            if (A[i]<0)
                  p=p*(A[i]*A[i]);
            i++;
      }
      while (i<m);

      cout << "Prodhimi p= "
           << p
           << "\n";
return 0;
}
```

Nëse ekzekutohet programi i dhënë, si rezultat në ekran shtypet:

```
Prodhimi p=15876
```

gjë që është fituar përmes llogaritjes:

$$p = (-7)^2 * (-3)^2 * (-6)^2 = 15876$$

Unaza të ndërthurura

Si tek unazat `for` dhe `while`, edhe brenda unazës `do-while` mund të përfshihen unaza të tjera, por të cilat mes vete nuk priten. Këto unaza mund të realizohen si unaza `do-while`, ose edhe si unaza `for` e `while`.

Shembull Programi përmes së cilit gjendet shuma e anëtarëve të matricës së dhënë `R(m,n)`.

```
// Programi p27
#include <iostream>
using namespace std;
int main()
{
      const int m=4,n=3;
      int i,j;
      int R[m][n]={{4,7,2},
                   {5,9,3},
                   {6,1,12},
                   {8,15,4}};
      float s;
      s=0;
      i=0;
      do
      {
            j=0;
            do
            {
                  s=s+R[i][j];
                  j++;
            }
            while (j<n);
            i++;
      }
      while (i<m);
      cout << "Shuma e llogaritur s="
           << s
             << "\n";
return 0;
}
```

Në program, me qëllim të mbledhjes së anëtarëve të matricës së dhënë `R(m,n)`, janë shfrytëzuar dy unaza `do-while`. Përmes variabës `i` të unazës së parë zgjedhen rreshtat e matricës. Kurse, variabla `j` e unazës së dytë (unazës së brendshme), e cila përfshihet në trupin e unazës së parë (unazës së jashtme),

shfrytëzohet për zgjedhjen e kolonave të matricës. Rezultati që fitohet pas ekzekutimit të programit në fjalë është:

```
Shuma e llogaritur s=76
```

Programi i dhënë mund të realizohet edhe përmes unazave `while`, ashtu siç shihet në vijim.

```
// Programi p28
#include <iostream>
using namespace std;
int main()
{
      const int m=4,n=3;
      int i,j;
      int R[m][n]={{4,7,2},
                   {5,9,3},
                   {6,1,12},
                   {8,15,4}};
      float s;
      s=0;
      i=0;
      while (i<m)
      {
            j=0;
            while (j<n)
            {
                        s=s+R[i][j];
                        j++;
            }
            i++;
      }
      cout << "Shuma e llogaritur s="
           << s
           << "\n";
return 0;
}
```

Nëse krahasohet ky program me versionin paraprak të tij, vërehet se dallimi është vetëm në vendosjen e komandave `while` në fillim të unazave, duke mos ndryshuar asgjë në pjesët e tjera të programit.

Por, nëse programi i dhënë realizohet përmes unazave `for`, ai do të jetë shumë më i thjeshtë, gjë që shihet në vijim.

```
// Programi p29
#include <iostream>
using namespace std;
```

```
int main()
{
      const int m=4,n=3;
      int i,j;
      int R[m][n]={{4,7,2},
                   {5,9,3},
                   {6,1,12},
                   {8,15,4}};
      float s;
      s=0;

      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  s=s+R[i][j];

      cout << "\nShuma e llogaritur s="
             << s
             << "\n";
return 0;
}
```

Unazat që shfrytëzohen brenda një programi nuk është e domosdoshme të jenë vetëm të një tipi. Kështu, p.sh., te versionet e programit të dhënë më sipër, nëse unaza e jashtme realizohet përmes unazës `do-while`, pjesa e programit me unazat do të duket si në vijim.

```
      i=0;
      do
      {
            for (j=0;j<n;j++)
                  s=s+R[i][j];
            i++;
      }
      while (i<m);
```

Rezultati që fitohet pas ekzekutimit të këtij versioni të programit do të jetë i njëjtë me atë që u dha në versionin e parë të tij, gjatë realizimit përmes unazave do-while.

Brenda unazave `do-while` mund të ketë edhe komanda të zakonshme për degëzim.

Shembull

Programi përmes së cilit gjendet anëtari më i madh në matricën e dhënë `A(m,n)`.

```
// Programi do3
```

```
#include <iostream>
using namespace std;
int main()
{
      const int m=4,n=5;
      int i,j,x;
      int A[m][n]={{6,3,5,-2,4},
                   {1,5,2,7,15},
                   {7,8,5,4,13},
                   {11,5,8,2,7}};
      x=A[0][0];

      i=0;
      do
      {
            j=0;
            do
            {
                  if (A[i][j]>x)
                        x=A[i][j];
                  j++;
            }
            while (j<n);
            i++;
      }
      while (i<m);

      cout << "Anëtari më i madh x="
           << x
           << "\n";
      return 0;
}
```

Këtu, si variabël e unazës së jashtme është marrë indeksi i rreshtave `i`, kurse unaza e brendshme e shfrytëzon indeksin e kolonave `j` të matricës. Anëtari më i madh ruhet te variabla x. Përmes shprehjes:

```
x=A[0][0];
```

kësaj variable fillimisht i shoqërohet vlera e anëtarit të parë matricës. Rezultati që shtypet në ekran, pas ekzekutimit të programit të dhënë, është:

```
Anëtari më i madh x=15
```

Nëse te programi i dhënë më sipër për realizimin e unazave shfrytëzohet komanda `for`, programi do të duket si në vijim.

```
// Programi do4
```

```
#include <iostream>
using namespace std;
int main()
{
  const int m=4,n=5;
  int i,j,x;
  int A[m][n]={ {6,3,5,-2,4},
                {1,5,2,7,15},
                {7,8,5,4,13},
                {11,5,8,2,7}};

  x=A[0][0];
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          if (A[i][j]>x)
             x=A[i][j];

  cout << "Anëtari më i madh x="
       << x
       << "\n";
  return 0;
}
```

Nëse krahasohen dy programet në fjalë, edhe këtu qartë shihet se realizimi i unazave përmes komandës `for` është më i thjeshtë, si dhe ka një dukshmëri më të madhe të veprimeve që kryhen gjatë gjetjes së anëtarit të kërkuar të matricës.

Brenda unazave të ndërthurura mund të paraqiten komanda me degëzime të përbëra.

Shembull Programi përmes së cilit radhiten sipas madhësisë anëtarët e vektorit të dhënë `A(m)`, duke filluar prej anëtarit më të madh.

```
// Programi p26
#include <iostream>
using namespace std;
int main()
{
      const n=8;
      int i,j,x,A[n]={-5,8,-2,6,3,15,7,9};
      cout << "\nVektori i dhënë\n\n";
      cout << "A=[  ";

      for (i=0;i<n;i++)
```

```
            cout << A[i]
                 << "  ";

      cout << "]\n";

      i=0;
      do
      {
            j=i+1;
            do
            {
               if (A[i]<A[j])
                     {
                        x=A[i];
                        A[i]=A[j];
                        A[j]=x;
                     }
               j++;
            }
            while (j<n);
            i++;
      }
      while (i<n-1);

      cout << "\nVektori pas radhitjes\n\n";
      cout << "A=[  ";

      for (i=0;i<n;i++)
            cout << A[i]
                 << "  ";

      cout << "]\n\n";
      return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që paraqitet në ekran do të duket si në *Fig.5.54*.

```
Vektori i dhënë
A=[  -5  8  -2  6  3  15  7  9  ]
Vektori pas radhitjes
A=[  15  9  8  7  6  3  -2  -5  ]
Press any key to continue
```

Fig.5.54
Rezultati i programit p26

Në program, për çdo vlerë të variablës `i`, përcaktohet vlera maksimale në pozitën përkatëse të vektorit. Prandaj, edhe variablës së unazës së dytë i jepet vlera fillestare `j=i+1`, ashtu që në krahasim të marrin pjesë vetëm anëtarët e vektorit të cilët ende nuk janë radhitur. Nëse gjatë krahasimit të dy anëtarëve të vektorit përmes komandës:

```
if (A[i]<A[j])
```

konstatohet se shprehja brenda kllapave ka vlerë `true`, vlerat e dy anëtarëve të vektorit duhet t'i ndërrojnë vendet. Për këtë qëllim shfrytëzohen shprehjet:

```
x=A[i];
A[i]=A[j];
A[j]=x;
```

të cilat janë vendosur brenda degës së përbërë të komandës `if`.

Kapërcimi i komandave brenda unazës

Tek unazat `do-while`, për kapërcimin e pjesëve të caktuara të komandave brenda unazës mund të shfrytëzohet komanda e kapërcimit pa kusht `goto`.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y = 2x + 3 \prod_{\substack{i=2 \\ (i\neq 3,5,6)}}^{n} (2i-1)$$

Vlerat e variablave `n` dhe `x` kompjuterit i jepen përmes tastierës si vlera hyrëse.

```
// Programi p30
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
```

```
      int i,n;
      double x,y,p;
      char g[]="------------------";
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << endl;
      cout << "\n    i         p\n"
           << g
           << endl;
      p=1;

      i=2;
      do
      {
            if ((i==3) || (i==5) || (i==6))
                        goto Vijues;
                  else
                  {
                        p=p*(2*i-1);
                        cout << setw(5)
                             << i
                             << setw(10)
                             << p
                             << endl;
                  }
      Vijues:
            i++;
      }
      while (i<=n);

      y=2*x+3*p;
            cout << g
                 << "\nRezultati y="
                 << y
                 << endl;
return 0;
}
```

Siç mund të shihet edhe në program, për kapërcimin e anëtarëve të prodhimit për vlerat e variablës së unazës `i=3`, `5` dhe `6` është shfrytëzuar komanda e kapërcimit pa kusht `goto Vijues`. Por, përmes kësaj komande kapërcehet te pjesa e fundit e unazës:

```
      Vijues:
            i++;
```

me qëllim që të vazhdojë rritja e vlerës së variablës së unazës.

Pas ekzekutimit të programit në fjalë, për vlera hyrëse të caktuara, në ekran do ta kemi pamjen e dhënë në *Fig.5.55*.

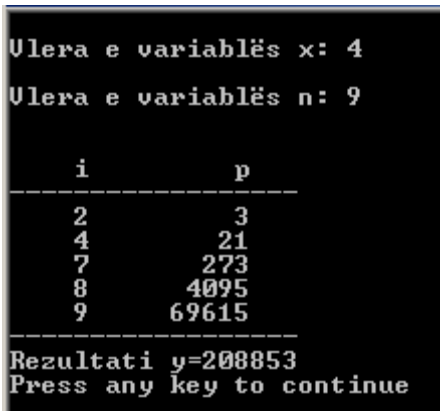


Fig.5.55
Rezultati i programit p30

Nga tabela e dhënë shihet qartë se është kapërcyer shumëzimi me anëtarët përkatës të variablës `i`, kur ajo i merr vlerat `3`, `5` dhe `6`.

Tek unaza `do-while`, kapërcimi nuk mund të bëhet me komandën `continue`, si tek unaza `for`. Kjo është e pamundur sepse përmes kësaj komande shkohet në fund të unazës, duke e kapërcyer edhe komandën për rritje të vlerës së variablës së unazës. Në këtë mënyrë procedura e llogaritjes shkon në pafundësi dhe për ndërprerje të ekzekutimit të programit, p.sh., në tastierë duhet të shtypet kombinimi i tasteve `Ctrl+Break`.

Dalja nga unaza

Sipas nevojës, ekzekutimi i komandave brenda trupit të unazës `do-while` mund të ndërpritet, duke dalë prej saj përmes komandës `goto`, ose komandës `break`.

Shembull

Programi përmes së cilit në çdo rresht të matricës së dhënë `R(m,n)` gjendet anëtari i parë negativ, duke u nisur prej fillimit të tij.

```
// Programi p31
#include <iostream>
#include <iomanip>
using namespace std;
int main()
```

```
{
      const int m=4,n=5;
      int i,j;
      char t[]="--------------------";
      int R[m][n]={{3,-4,7,-2,9},
                   {5,8,16,11,14},
                   {-17,4,-1,6,2},
                   {6,8,3,1,-15}};
      cout << "\n Indeksi     Anëtari\n"
           << t
           << endl;

      i=0;
      do
      {
            j=0;
            do
            {
                  if (R[i][j]<0)
                  {
                        cout << setw(5)
                             << i
                             << setw(13)
                             << R[i][j]
                             << endl;
                        goto Dalje;
                  }
                  else
                  j++;
            }
            while (j<n);
Dalje:
            i++;
      }
      while (i<m);

      cout << t
           << "\n";
return 0;
}
```

Përmes dy unazave të shfrytëzuara në program, kalohet nëpër rreshtat e matricës së dhënë. Gjatë kësaj, duke e shfrytëzuar pjesën e programit:

```
      if (R[i][j]<0)
      {
            cout << setw(5)
                 << i
```

```
                << setw(13)
                << R[i][j]
                << endl;
          goto Dalje;
      }
```

kontrollohen anëtarët e matricës se a kanë vlera negative. Anëtari i parë negativ që gjendet, në fakt është anëtari i parë negativ në rreshtin aktual (rreshtin me indeks `i`), sepse, pasi të shtypet indeksi dhe vlera e gjetur e anëtarit të matricës, dilet nga unaza e brendshme duke e shfrytëzuar komandën e kapërcimit pa kusht `goto Dalje`. Me të në fakt ndërpritet shqyrtimi i anëtarëve të tjerë të rreshtit aktual të matricës, dhe kalohet në rreshtin vijues të matricës sepse labela `Dalja:` është vendosur para komandës me të cilën rritet indeksi i rreshtave të matricës.

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket ashtu siç shihet në *Fig.5.56*.

```
 Indeksi     Anëtari
--------------------
    0           -4
    2          -17
    3          -15
--------------------
Press any key to continue
```

Fig.5.56
Rezultati i programit p31

Siç shihet nga tabela e dhënë, te rreshti me indeks `i=1` nuk është gjetur asnjë anëtar negativ.

Programi mund të plotësohet ashtu që, nëse në ndonjë rresht nuk gjendet asnjë anëtar negativ, në kolonën me vlera të anëtarëve të matricës të shtypet, p.sh., mesazhi `Nuk ka`. Për këtë qëllim, në program duhet të shtohet komanda për shtypje të mesazhit në fjalë, menjëherë pas daljes nga unaza e brendshme, ashtu siç shihet në vijim.

```
      }
      while (j<n);
      cout << setw(5)
           << i
           << "         Nuk ka\n";
Dalje:
      i++;
      }
```

Në këtë mënyrë, kur shqyrtimi i anëtarëve të rreshtave të veçantë të matricës përfundon pa e gjetur asnjë anëtar negativ, nga unaza e brendshme do

të dilet në rrugë normale, pa e kapërcyer fundin e unazës përmes komandës `goto Dalje`.

Nëse pas ndryshimit në fjalë ekzekutohet programi i dhënë, rezultati në ekran do të duket si në *Fig.5.57*.

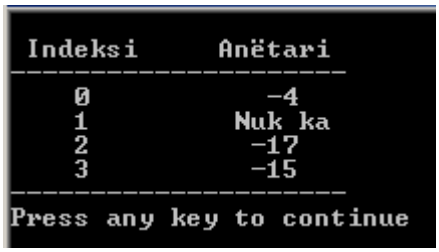


Fig.5.57
Rezultati i versionit të modifikuar të programit p31

Për dalje nga unaza, në vend të komandës `goto` mund të shfrytëzohet edhe komanda `break`. Kështu, p.sh., te programi `p31` i dhënë më sipër do të dilet nga unaza e brendshme pas kushtit të përfshirë në komandën `if`, nëse në vend të komandës `goto Dalje` vendoset komanda `break`. Gjatë kësaj, labela `Dalje:` në pjesën vijuese të programit është e panevojshme.

6

Funksionet

Gjatë shkruarjes së programeve të ndryshme përdoren biblioteka me funksione të gatshme, të përcaktuara nga përpiluesi i kompajlerit të gjuhës `C++`. Por, edhe vetë shfrytëzuesi mund të përcaktojë funksione të ndryshme dhe t'i shfrytëzojë ato sipas nevojës.

Një grumbull i komandave të caktuara brenda programit mund të përsëritet më shumë herë. Me qëllim të thjeshtimit të shkruarjes së programit, grumbulli i tillë deklarohet si pjesë e veçantë, e cila quhet *nënprogram*. Pastaj, nënprogrami thirret përmes emrit të tij, sa herë që nevojitet.

Në gjuhën `C++` shfrytëzohen vetëm tipe të nënprogrameve që njihen si *funksione* (ang. function). Prandaj, zakonisht, në këtë gjuhë, kur flitet për nënprograme, mendohet në përcaktimin dhe në shfrytëzimin e funksioneve të ndryshme.

Në fillim të librit është përmendur funksioni `main()`, i cili paraqitet në çdo program. Gjithashtu, në pjesët paraprake janë përdorur disa funksione matematikore të cilat gjenden në kuadër të modulit `math.h`. Kurse në fund të këtij kapitulli jepen shembuj të tjerë të shfrytëzimit të funksioneve matematikore nga moduli në fjalë. Njëkohësisht, shpjegohet përdorimi i disa funksioneve nga biblioteka përkatëse për punë me stringje.

Definimi dhe thirrja e funksioneve

Nënprogramet përmes së cilave definohen funksione të zakonshme shkruhen në kuadër të programit tek i cili shfrytëzohen. Nënprogrami me titullin `main()`, ku thirren funksionet, njihet edhe si *program kryesor*.

Gjatë ekzekutimit të programit, në momentin kur kompjuteri e takon emrin e një funksioni, ekzekutimi vazhdon me komandat e përfshira brenda nënprogramit. Kur përfundon ekzekutimi i komandave të nënprogramit, procedura e ekzekutimit vazhdon me komandat vijuese të programit, të cilat gjenden menjëherë pas emrit të funksionit.

Në formë të përgjithshme, definimi i një funksioni duket kështu:

```
tipi emri(t1 f1,t2 f2,...,tn fn)
{
   x;
   return r;
}
```

ku janë:

`tipi` - tipi i rezultatit të funksionit.
`emri` - emri i funksionit, i zgjedhur lirisht si identifikator.
`f1, f2, ..., fn` - parametrat formalë.
`t1,t2,...,tn` - tipet e parametrave formalë.
`x` - komanda të ndryshme brenda trupit të funksionit.
`r` - rezultati të cilin e jep funksioni.

Njëkohësisht, në fillim të programit kryesor, ku definohet dhe shfrytëzohet funksioni, jepet *prototipi i funksionit* (ang. function prototipe), i cili përfundon me pikëpresje (`;`). Forma e përgjithshme e prototipit të funksionit shkruhet si edhe titulli i tij, kështu:

```
tipi emri(t1 f1,t2 f2,...,tn fn);
```

ose, edhe pa i shkruar parametrat formalë brenda kllapave:

```
tipi emri(t1,t2,...,tn);
```

Por, meqë varianti i parë tek i cili, përveç tipeve, shihen edhe parametrat, është më komplet, në vijim do të shfrytëzohet vetëm ky variant. Përmes prototipit të funksionit, kompajleri njoftohet me emrin e funksionit që definohet, numrin, tipin dhe radhën e shkruarjes së parametrave formalë brenda kllapave të tij.

Për ta shfrytëzuar funksionin në llogaritje të ndyshme, ai thirret përmes emrit të tij. Gjatë kësaj, parametrat formalë:

```
f1, f2, ..., fn
```

zëvendësohen me parametrat aktualë përkatës:

```
a1, a2, ..., an
```

kështu:

```
emri(a1,a2,...,an)
```

Parametrat e shënuar brenda kllapave në titullin e funksionit gjatë definimit të tij, quhen *parametra formalë*, sepse përmes tyre tregohet forma e llogaritjeve që kryhen brenda funksionit. Kurse, parametrat me të cilët zëvendësohen ato formalë gjatë thirrjes së nënprogramit, siç u përmend edhe më sipër, quhen *parametra aktualë*.

Parametrat formalë dhe ato aktualë duhet të përputhen mes vete për nga:

- *numri* - sa ka parametra formalë aq duhet të ketë edhe parametra aktualë;
- *tipi* - tipin e njëjtë duhet ta kenë parametrat formalë dhe parametrat aktualë përkatës;
- *radha e shkruarjes* - parametrit formalë në pozitë të caktuar i përgjigjet parametër aktual me pozitë të njëjtë.

Shembull Programi përmes së cilit llogaritet sipërfaqja e katërkëndëshit me brinjët `a` dhe `b`, duke e shfrytëzuar funksionin `Dita`.

a. Varianti pa nënprogram

```
// Programi funk1a
#include <iostream>
using namespace std;
int main()
{
      float a,b;
      double s;
      cout << "\nBrinja a: ";
      cin >> a;
      cout << "\nBrinja b: ";
      cin >> b;

      s=a*b;

      cout << "\nSipërfaqja s="
           << s
           << "\n\n";
return 0;
}
```

Në programin e dhënë, pasi përmes tastierës kompjuterit t'i jepen vlerat e brinjëve të katërkëndëshit, me shprehjen e zakonshme llogaritet vlera e sipërfaqes së katërkëndëshit. Pastaj, duke e shfrytëzuar komandën për shtypje, në ekran shtypet rezultati, i cili për vlera hyrëse të caktuara duket si në *Fig.6.1*.

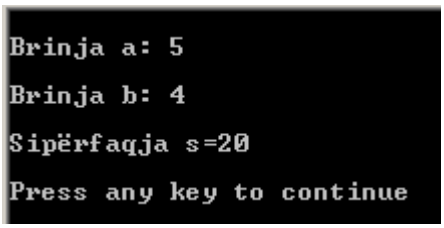


Fig.6.1
Rezultati i programit funk1a

b. Varianti me nënprogram

```
// Programi funk1b
#include <iostream>
using namespace std;
double Dita(float x,float y);
int main()
{
      float a,b;
      double s;
      cout << "\nBrinja a: ";
      cin >> a;
      cout << "\nBrinja b: ";
      cin >> b;

      s=Dita(a,b);

      cout << "\nSipërfaqja s="
           << s
           << "\n\n";
return 0;
}

// Nënprogrami Dita

double Dita(float x,float y)
{
      return x*y;
}
```

Nëse analizohet verzioni i dhënë i programit, do të shihet se në fillim të programit, menjëherë pas komandës:

```
#include <iostream>
```

është vendosur prototipi i funksionit `Dita`:

```
double Dita(float x,float y);
```

i tipit `double`, me parametrat formalë `x` dhe `y`, të cilët janë deklaruar të tipit `float`. Njëkohësisht, në fund të programit, është vendosur nënprogrami `Dita`:

```
// Nënprogrami Dita

double Dita(float x,float y)
{
      return x*y;
}
```

Siç shihet më sipër, nënprogrami fillon me titullin e funksionit:

```
double Dita(float x,float y)
```

i cili është plotësisht i njëjtë me prototipin e vendosur në fillim të programit. Brenda trupit të nënprogramit gjendet vetëm komanda:

```
      return x*y;
```

përmes së cilës gjatë thirrjes së tij, në programin kryesor, përcillet vlera e llogaritur e prodhimit `x*y`.

Në programin kryesor, funksioni thirret përmes emrit `Dita`, kështu:

```
      s=Dita(a,b);
```

duke i zëvendësuar parametrat formalë `x` dhe `y`, me brinjët e katërkëndëshit `a` dhe `b`, të cilat paraqiten si parametra aktualë. Gjatë ekzekutimit të këtij versioni të programit, rezultati në ekran do të duket plotësisht njëlloj si ai që u dha në *Fig.6.1*.

Siç u tha edhe më parë, prototipi i funksionit në fillim të programit mund të deklarohet edhe duke i shënuar vetëm tipet e parametrave formalë, por jo edhe parametrat përkatës. Kështu, në shembullin e programit të dhënë më sipër, prototipit i funksionit mund të deklarohet edhe në këtë mënyrë:

```
double Dita(float,float);
```

Gjatë thirrjes së funksionit, parametrat aktualë mund të jenë vlera numerike, ose edhe shprehje. P.sh., te programi i mësipërm, gjatë llogaritjes së sipërfaqes, mund të shkruajmë:

```
      s=Dita(5,4);
```

ku numri `5` i përgjigjet brinjës `a`, kurse numri `4` - brinjës `b`.

Nënprogrami nga shembulli i mësipërm mund të shkruhet edhe kështu:

```
// Nënprogrami Dita

double Dita(float x,float y)
{
      double z;
      z=x*y;
      return z;
}
```

ku, siç shihet, sipërfaqja është përcaktuar duke e shfrytëzuar variablën ndihmëse `z`, e cila pastaj shkruhet në vazhdim të komandës `return`. Këtu, meqë brenda funksionit paraqitet variabla e re `z`, ajo është deklaruar në fillim të tij si variabël e tipit `double`.

Funksioni mund të definohet edhe në fillim të programit.

Gjatë definimit të funksionit, si parametra formalë mund të merren edhe variabla të njëjta me ato të cilat do të paraqiten si parametra aktualë, kur ai thirret në program.

Shembull

Versioni i programit funk1b, tek i cili gjatë definimit të funksionit `Dita`, parametrat formalë për brinjët e katërkëndëshit janë marrë si `a` dhe `b`, dhe përputhen me parametrat aktualë gjatë thirrjes së tij.

```
// Programi funk1c
#include <iostream>
using namespace std;
double Dita(float a,float b);
int main()
{
      float a,b;
      double s;
      cout << "\nBrinja a: ";
      cin >> a;
      cout << "\nBrinja b: ";
      cin >> b;
      s=Dita(a,b);
      cout << "\nSipërfaqja s="
           << s
           << "\n\n";
}

// Nënprogrami Dita
```

```
double Dita(float a,float b)
{
      double z;
      z=a*b;
      return z;
}
```

Këtu shihet se parametrat formalë te nënprogrami janë variablat a dhe b. Gjatë thirrjes së funksionit, për ta llogaritur vlerën e sipërfaqes s, si parametra aktualë gjithashtu paraqiten variablat a dhe b.

Nëse funksioni që definohet ka më shumë degë, në nënprogramin përkatës mund shfrytëzohen disa komanda return.

Shembull

Programi funkA përmes së cilit llogaritet vlera e funksionit:

$$y = \begin{cases} 2x+1 & \text{për } x \le a \\ 3x-4 & \text{për } x > a \end{cases}$$

duke e shfrytëzuar nënprogramin vleraY. Vlera e variablës a përcaktohet si konstante në fillim të programit, kurse vlera e variablës x kompjuterit i jepet përmes tastierës si vlerë hyrëse.

a. Versioni i nënprogramit me një komandë `return`

```
// Programi funkA
#include <iostream>
using namespace std;
float vleraY(float x,float a);
int main()
{
      const float a=4.5;
      float x,y;
      cout << "\nVlera e variablës x: ";
      cin >> x;

      y=vleraY(x,a);

      cout << "\nVlera e llogaritur y="
           << y
           << "\n\n";
}

// Nënprogrami vleraY
```

```
float vleraY(float x,float a)
{
      float y;
      if (x<=a)
            y=2*x+1;
      else
            y=3*x-4;
      return y;
}
```

Këtu, gjatë përcaktimit të funksionit në nënprogram edhe pse paraqiten dy degë, si zakonisht shfrytëzohet vetëm një degë e komandës `return`.

Nëse gjatë përcaktimit të vlerës së variablës `a` si konstante nuk deklarohet edhe tipi i saj `float`, përmes deklarimit:

```
const float a=4.5;
```

gjatë shfrytëzimit si parametër aktual te funksioni, kompjuteri na njofton për mundësinë e gabimit për shkak të konvertimit në tipin `float` (ashtu siç është deklaruar parametri formal përkatës në nënprogram).

Rezultati i programit të dhënë për vlerën e variablës hyrëse `x=3` do të duket si në *Fig.6.2*.

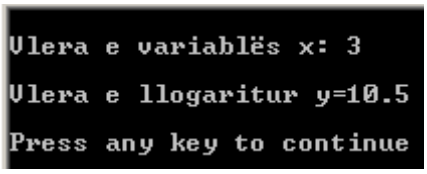


Fig.6.2
Rezultati i programit funkA

b. Versioni i nënprogramit me dy komanda `return`

Programi i dhënë më sipër mbetet i pandryshuar, kurse nënprogrami përmes së cilit përcaktohet llogaritja e vlerës së funksionit `y` do të duket si në vijim.

```
// Nënprogrami vleraY

float vleraY(float x,float a)
{
      if (x<=a)
            return 2*x+1;
      else
            return 3*x-4;
}
```

Siç shihet nga nënprogrami i dhënë, këtu për çdo shprehje të funksionit `y` është shfrytëzuar një komandë e veçantë `return`. Kjo formë e shkruarjes së nënprogramit mund të shfrytëzohet në rastet kur në degët e veçanta rezultati merret përmes shprehjeve ose variablave të ndryshme.

Llogaritje komplekse brenda funksionit

Brenda funksionit mund të paraqiten edhe llogaritje më komplekse. Por, në fund, variabla në të cilën përfshihet rezultati i të gjitha llogaritjeve përsëri duhet të shënohet te komanda `return`.

Shembull Programi përmes së cilit llogaritet prodhimi i numrave natyrorë mes `m` dhe `n`, duke e shfrytëzuar nënprogramin `prodhimi`.

a. Pa nënprogram

```
// Programi funk2
#include <iostream>
using namespace std;
int main()
{
      double p;
      int i,m,n;
      cout << "\nVlera e variablës m: ";
      cin >> m;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      p=1;
      for (i=m;i<=n;i++)
            p=p*i;

      cout << "\nProdhimi p="
           << p
           << "\n\n";
return 0;
}
```

Në program, gjatë llogaritjes së prodhimit:

$$p = \prod_{i=m}^{n} i$$

është shfrytëzuar unaza e zakonshme `for`. Nëse ekzekutohet programi i dhënë, për vlerat hyrëse `m=3` dhe `n=9`, rezultati në ekran do të duket si në *Fig.6.3*.

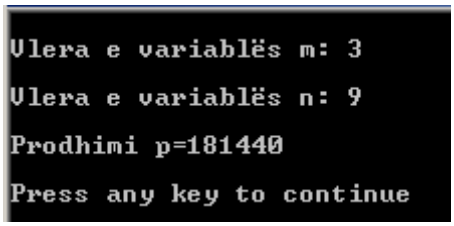


Fig.6.3
Rezultati i programit funk2

Në shembullin e vlerave hyrëse, që shihen më sipër, prodhimi në fakt është llogaritur kështu:

```
p = 3 * 4 * 5 * 6 * 7 *8 * 9 = 181440
```

b. Me nënprogram

```
// Programi funk2
#include <iostream>
double prodhimi(int m,int n);
using namespace std;
int main()
{
      double p;
      int m,n;
      cout << "\nVlera e variablës m: ";
      cin >> m;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      p=prodhimi(m,n);
      cout << "\nProdhimi p="
           << p
           << "\n\n";
return 0;
}

// Nënprogrami prodhimi

double prodhimi(int m,int n)
{
      double p;
      int i;
      p=1;
      for (i=m;i<=n;i++)
              p=p*i;
      return p;
}
```

Gjatë definimit të llogaritjes së prodhimit, në nënprogram janë shfrytëzuar parametrat formalë `m` dhe `n`. Por, njëkohësisht, brenda tij janë përdorur edhe variablat ndihmëse `p` dhe `i`, tipet e të cilave deklarohen në fillim të tij. Në fund, te komanda `return` është shënuar variabla `p`, në të cilën ruhet rezultati i llogaritjes.

Funksioni mund të thirret edhe direkt në komandën për shtypje të rezultateve, p.sh., ashtu siç është dhënë në vijim.

```
// Programi funk3
#include <iostream>
double prodhimi(int m,int n);
using namespace std;
int main()
{
      int m,n;
      cout << "\nVlera e variablës m: ";
      cin >> m;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nProdhimi p="
           << prodhimi(m,n)
           << "\n\n";
return 0;
}
```

Në program më nuk është shfrytëzuar variabla e prodhimit `p`, sepse vlera përkatëse llogaritet dhe shtypet brenda komandës për shtypje. Këtu, nënprogrami mbetet i pandryshuar nga ai që u dha më parë. Nëse ekzekutohet programi i dhënë, për vlerat hyrëse `m=3` dhe `n=9`, rezultati në ekran do të duket njëlloj me atë që u dha në *Fig.6.3*.

Thirrja e nënprogramit mund të bëhet edhe brenda unazave, ashtu që si parametra aktualë të shfrytëzohen edhe variablat e unazave.

Shembull

Programi përmes së cilit përpilohet tabela e faktorielëve të numrave natyrorë mes vlerave `1` dhe `k`. Vlera e variablës `k` kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi funk4
#include <iostream>
#include <iomanip>
double prodhimi(int m,int n);
using namespace std;
int main()
{
      double F;
```

```
      int k,x;
      char h[]="-------------------";
      cout << "\nVlera e variablës k: ";
      cin >> k;
      cout << "\n    x         F\n"
           << h
           << "\n";

      for (x=1;x<=k;x++)
      {
            F=prodhimi(1,x);
            cout << setw(5)
                 << x
                 << setw(10)
                 << F
              << "\n";
      }

      cout << h
           << endl;
return 0;
}

// Nënprogrami prodhimi
double prodhimi(int m,int n)
{
      double p;
      int i;
      p=1;
      for (i=m;i<=n;i++)
              p=p*i;
      return p;
}
```

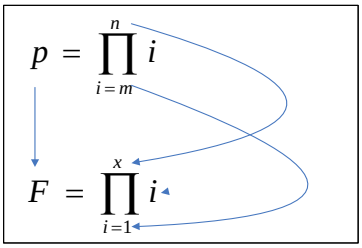
Këtu, për llogaritje të faktorielit për çdo vlerë të variablës së unazës `x`, është shfrytëzuar funksioni `prodhimi`, i përcaktuar përmes nënprogramit. Meqë duhet të llogaritet faktorieli për çdo vlerë të variablës `x`, shprehja përkatëse për llogaritje:

```
F=prodhimi(1,x)
```

është vendosur brenda unazës dhe si parametër i dytë brenda kllapave të funksionit është shfrytëzuar variabla e unazës `x`. Duke i pasur parasysh parametrat formalë m dhe n te nënprogrami, si dhe parametrat aktualë `1` dhe `x` brenda kllapave të funksionit, përmes shprehjes në fjalë llogaritet vlera `x!`, kështu:

$$F = \prod_{i=1}^{x} i$$

Gjatë kësaj, zëvendësimi i parametrave formalë `m` e `n`, që shfrytëzohen te funksioni `prodhimi`, me parametrat aktualë përkatës `1` dhe `x`, skematikisht mund të paraqitet në këtë mënyrë:



Nëse ekzekutohet programi i dhënë për vlerën hyrëse k=8, rezultati në ekran do të duket si në *Fig.6.4*.

```
Vlera e variablës k: 8

    x          F
--------------------
    1          1
    2          2
    3          6
    4         24
    5        120
    6        720
    7       5040
    8      40320
--------------------
Press any key to continue
```

Fig.6.4
Rezultati i programit funk4

Funksioni pa rezultat dalës

Funksioni që definohet përmes një nënprogrami mund të shfrytëzohet për llogaritje dhe shtypje të rezultateve, pa dhënë asnjë rezultat dalës. Në raste të tilla, komanda `return` shkruhet pa variablën përcjellëse në fund të saj, kurse para emrit të funksionit shkruhet fjala `void`.

Shembull

Programi i cili e shfrytëzon funksionin `jeta` për shtypje në ekran të tekstit `Koha e bukur`.

```
// Programi funk5
#include <iostream>
using namespace std;
void jeta(int m,int n);
```

```
int main()
{
     jeta(1,10);
return 0;
}

// Nënprogrami jeta
void jeta(int m,int n)
{
     int i;
     for (i=m;i<=n;i++)
         cout << "Koha e bukur\n";
     return;
}
```

Këtu, meqë nënprogrami nuk jep si rezultat asnjë vlerë, para emrit të nënprogramit është shkruar fjala `void`, si dhe në fund të komandës `return` te nënprogrami nuk është shënuar asgjë. Brenda unazës së nënprogramit është vendosur komanda për shtypje të fjalisë `Koha e bukur`. Vlerat e variablës së unazës `i` brenda nënprogramit lëvizin mes vlerës fillestare `i=m` dhe vlerës përfundimtare `i=n`. Pasi që gjatë shfrytëzimit të funksionit `jeta` në programin kryesor parametrave formalë `m` dhe `n` u janë shoqëruar parametrat aktualë `1` dhe `10`, gjatë ekzekutimit të programit në ekran fjalia do të shtypet `10` herë, ashtu siç shihet në *Fig.6.5*.

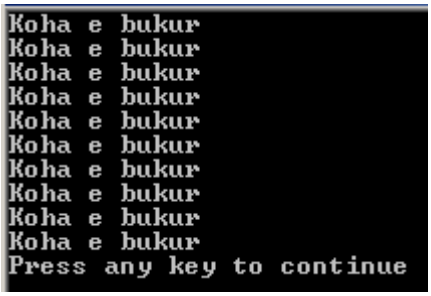


Fig.6.5
Rezultati i programit funk5

Rezultati në ekran do të duket plotësisht njëlloj, nëse gjatë thirrjes së funksionit, p.sh., si parametra aktualë merren vlerat `m=11` dhe `n=20`, në këtë mënyrë:

```
jeta(11,20);
```

sepse edhe në këtë rast ekzekutimi i trupit të unazës do të përsëritet `10` herë.

Funksioni pa parametra formalë

Funksioni mund të mos ketë parametra formalë, nëse rezultatet e tij janë fikse. Të tillë, p.sh., mund të jenë funksionet për shtypje të mesazheve të caktuara.

Shembull Programi i cili e shfrytëzon funksionin `libri` për shtypje të përsëritur në ekran të tekstit `Libri dhe lapsi`.

```
// Programi funk6
#include <iostream>
#include <iomanip>
using namespace std;
void libri();
int main()
{
      libri();
return 0;
}

// Nënprogrami libri
void libri()
{
      int i;
      for (i=1;i<=10;i++)
      {
            cout << setw(40-2*i)
                 << "Libri dhe lapsi\n";
      }
      return;
}
```

Nëse ekzekutohet programi i dhënë, rezultati do të duket si në *Fig.6.6*.

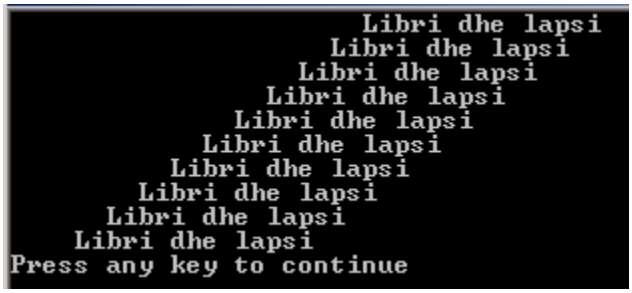


Fig.6.6
Rezultati i programit funk6

Këtu, brenda unazës së nënprogramit është përcaktuar që të shtypet `10` herë fjalia `Libri dhe lapsi`, sa është numri i vlerave të ndryshme të variablës së unazës. Meqë numri i rreshtave që shtypen është fiks, brenda kllapave të funksionit nuk është shënuar asnjë parametër.

Më shumë thirrje të një funksioni

Në pjesën paraprake, te programi `funk4`, u shpjegua thirrja e përsëritur e funksionit brenda një unaze. Por, sipas nevojës, brenda programit mund të ketë më shumë thirrje të pavarura të funksionit.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y = 3x + (n+1)! + (2n)! + \frac{n!}{x}$$

Faktorielët të cilët paraqiten në shprehje llogariten përmes nënprogramit `fakt`, me të cilin definohet llogaritja e vlerës së funksionit `F=m!`.

a. Pa nënprogram

```
// Programi funk7a
#include <iostream>
using namespace std;
int main()
{
      double a,b,c,x,y;
      int i,n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      a=1;
      for (i=1;i<=(n+1);i++)
            a=a*i;

      b=1;
      for (i=1;i<=(2*n);i++)
            b=b*i;

      c=1;
      for (i=1;i<=n;i++)
```

```
            c=c*i;

      y=3*x+a+b+c/x;

      cout << "\nVlera e llogaritur y="
           << y
           << "\n\n";
return 0;
}
```

Në program, për llogaritje të vlerave të faktorielëve, që paraqiten në shprehje, janë shfrytëzuar tri unaza të veçanta. Nëse gjatë ekzekutimit të programit përmes tastierës kompjuterit i jepen vlerat x=4 dhe n=3, rezultati në ekran do të duket si në *Fig.6.7*.

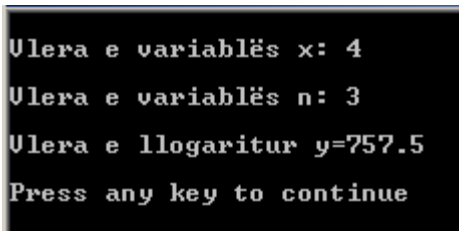


Fig.6.7
Rezultati i programit funk7a

b. Me nënprogram

```
// Programi funk7b
#include <iostream>
double fakt(int m);
using namespace std;
int main()
{
      double a,b,c,x,y;
      int n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      a=fakt(n+1);
      b=fakt(2*n);
      c=fakt(n);

      y=3*x+a+b+c/x;

      cout << "\nVlera e llogaritur y="
           << y
           << "\n\n";
```

```
return 0;
}


// Nënprogrami fakt
double fakt(int m)
{
      double F;
      int i;
      F=1;
      for (i=1;i<=m;i++)
            F=F*i;
      return F;
}
```

Këtu, për llogaritje të vlerave të faktorielëve është shfrytëzuar funksioni `fakt`. Gjatë `3` thirrjeve të funksionit, për t'i llogaritur vlerat `a`, `b` dhe `c`, parametri formal `m`, që shfrytëzohet te funksioni, është zëvendësuar me `3` parametrat aktualë:

```
n+1
2*n
n
```

Nëse ekzekutohet programi `funk7b`, rezultati në ekran do ta ketë pamjen e dhënë në *Fig.6.7*.

Funksionet mund të thirren direkt në shprehjet e ndryshme matematikore, në të cilat nevojiten vlera që llogariten përmes tyre.

Shembull Versioni i programit `funk7b`, pas eliminimit të variablave `a`, `b` dhe `c` si të panevojshme.

```
// Programi funk7c
#include <iostream>
double fakt(int m);
using namespace std;
int main()
{
      double x,y;
      int n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;
```

```
      y=3*x+fakt(n+1)+fakt(2*n)+fakt(n)/x;

      cout << "\nVlera e llogaritur y="
           << y
           << "\n\n";
return 0;
}

// Nënprogrami fakt
double fakt(int m)
{
      double F;
      int i;
      F=1;
      for (i=1;i<=m;i++)
            F=F*i;
      return F;
}
```

Në praktikë, zakonisht shfrytëzohet kjo formë e thirrjes së funksioneve, pa shtuar variabla ndihmëse gjatë llogaritjeve të ndryshme.

Plotësisht njëlloj thirren funksionet të cilat përmbajnë më shumë parametra formalë.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$h = x - 2\sum_{i=2}^{n+1}(2i+3) + 4\sum_{i=1}^{n}(i-5)$$

duke shfrytëzuar një nënprogram për llogaritjen e shumës.

a. Pa nënprogram

```
// Programi funk8a
#include <iostream>
using namespace std;
int main()
{
      float x;
      double s1,s2,h;
      int i,n;
      cout << "\nVlera e variablës x: ";
```

```
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      s1=0;
      for (i=2;i<=(n+1);i++)
            s1=s1+(2*i+3);

      s2=0;
      for (i=1;i<=n;i++)
            s2=s2+(i-5);

      h=x-2*s1+4*s2;

      cout << "\nVlera e llogaritur h="
           << h
           << "\n\n";
return 0;
}
```

Në program, për llogaritje të çdo shume është shfrytëzuar një unazë e veçantë. Në fund, rezultatet e dy shumave `s1` dhe `s2` janë shfrytëzuar në shprehjen për llogaritjen e vlerës së funksionit `h`. Nëse ekzekutohet programi, për vlera hyrëse të caktuara, rezultati në ekran do të duket ashtu siç është dhënë në *Fig.6.8*.

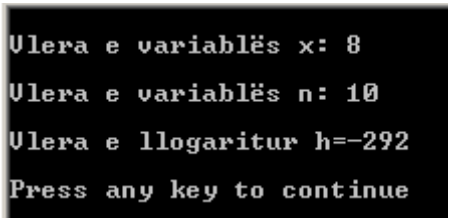


Fig.6.8
Rezultati i programit funk8a

b. Me nënprogram

```
// Programi funk8b
#include <iostream>
double shuma(int m,int n,float a,float b);
using namespace std;
int main()
{
      float x;
      double h;
      int n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;
```

```
      h=x-2*shuma(2,n+1,2,3)+4*shuma(1,n,1,-5);

      cout << "\nVlera e llogaritur h="
           << h
           << "\n\n";

return 0;
}

// Nënprogrami shuma

double shuma(int m,int n,float a,float b)
{
      float s;
      int i;
      s=0;
      for (i=m;i<=n;i++)
            s=s+(a*i+b);
      return s;
}
```

Në nënprogramin `shuma`, gjatë definimit të funksionit për llogaritjen e shumës është shfrytëzuar shprehja:

$$s = \sum_{i=m}^{n} (a \cdot i + b)$$

Me qëllim të krahasimit me këtë shprehje të dy shumave që paraqiten te funksioni h, ato mund të shkruhen në format:

$$s1 = \sum_{i=2}^{n+1} (2 \cdot i + 3)$$

$$s2 = \sum_{i=1}^{n} (1 \cdot i + (-5))$$

Pas kësaj, lehtë shihet se parametrat formalë:

```
m      n      a      b
```

gjatë llogaritjes së shumave përmes funksionit `shuma` janë zëvendësuar me parametrat aktualë përkatës:

```
2    n+1      2      3
```

dhe

```
1     n     1     -5
```

gjë që shihet edhe gjatë thirrjes së funskionit brenda shprehjes h:

```
shuma(2,n+1,2,3)
shuma(1,n,1,-5)
```

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket plotësisht njëlloj me atë që u dha në *Fig.6.8*.

Në një program njëkohësisht mund të ketë më shumë llogaritje të veçanta. Gjatë kësaj është më racionale të tentohet që për pjesët që përsëriten të përpilohet nënprogrami përkatës.

Shembull

Programi përmes së cilit llogariten vlerat e funksioneve:

$$y = 3x + (n+2)!$$

$$z = 2y - (n+1)! + \frac{4!}{3}$$

duke e shfrytëzuar një nënprogram për llogaritjen e faktorielëve. Vlera e variablës x llogaritet si shumë e numrave natyrorë mes 1 dhe n, kurse numri n kompjuterit i jepet përmes tastierës si vlerë hyrëse.

```
// Programi funk9
#include <iostream>
double fakt(int m);
using namespace std;
int main()
{
      double x,y,z;
      int i,n;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      x=0;
      for (i=1;i<=n;i++)
            x=x+i;

      y=3*x+fakt(n+2);
      z=2*y-fakt(n+1)+fakt(4)/3;

      cout << "\nVlera e llogaritur y="
```

```
            << y
            << "\n\nVlera e llogaritur z="
            << z
            << "\n\n";
return 0;
}

// Nënprogrami fakt

double fakt(int m)
{
      double F;
      int i;
      F=1;
      for (i=1;i<=m;i++)
            F=F*i;
      return F;
}
```

Në program, vlera e variablës `x` është llogaritur pa nënprogram, duke e shfrytëzuar shprehjen:

$$x = \sum_{i=1}^{n} i$$

Kurse, përmes nënprogramit `fakt` është definuar llogaritja e faktorielit `F=m!`, ku `m` është marrë si parametër formal. Pastaj, për llogaritjen e vlerave të faktorielëve, që paraqiten në të dy shprehjet, `3` herë është shfrytëzuar funksioni `fakt`, duke e zëvendësuar parametrin formal `m` me parametrat aktualë përkatës: `n+2`, `n+1` dhe `4`.

Nëse ekzekutohet programi i dhënë, për vlerën hyrëse `n=5`, rezultati në ekran do të duket si në *Fig.6.9*.

```
Vlera e variablës n: 5
Vlera e llogaritur y=5085
Vlera e llogaritur z=9458
Press any key to continue
```

Fig.6.9
Rezultati i programit funk9

Disa nënprograme njëkohësisht

Në kuadër të një programi njëkohësisht mund të shfrytëzohen disa nënprograme për përcaktimin e funksioneve. Radha e vendosjes së nënprogrameve brenda programit nuk ka rëndësi.

Shembull

Programi përmes së cilit llogariten vlerat e shprehjeve:

$$g = \frac{x}{3} + (2n+1)!$$

$$h = x + \frac{n!}{2} - 3\sum_{i=1}^{n+1}(4i)$$

Për llogaritje të faktorielit dhe të shumës janë shfrytëzuar dy funksione të veçanta. Vlerat e variablave x dhe n kompjuterit i jepen si vlera hyrëse përmes tastierës.

```
// Programi funk10
#include <iostream>
double fakt(int m);
double shuma(int m,float x);
using namespace std;
int main()
{
      double x,g,h;
      int n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      g=x/3+fakt(2*n+1);
      h=x+fakt(n)/2-3*shuma(n+1,4);

      cout << "\nVlera e llogaritur g="
           << g
           << "\n\nVlera e llogaritur h="
           << h
           << "\n\n";
```

```
return 0;
}

// Nënprogrami fakt

double fakt(int m)
{
      double F;
      int i;
      F=1;
      for (i=1;i<=m;i++)
            F=F*i;
      return F;
}

// Nënprogrami shuma

double shuma(int m,float k)
{
      double s;
      int i;
      s=0;
      for (i=1;i<=m;i++)
            s=s+(k*i);
      return s;
}
```

Përmes dy nënprogrameve definohen llogaritjet e vlerave të funksioneve:

$$F = m!$$

$$s = \sum_{i=1}^{m} (k \cdot i)$$

Këtu, si edhe te shembujt paraprakë, funksionet janë quajtur me emrat `fakt` dhe `shuma`, gjë që paraqet një zgjedhje të lirë të emrave. Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran për vlerat hyrëse `x=5` dhe `n=3` do të duket si në *Fig.6.10*.

```
Vlera e variablës x: 5
Vlera e variablës n: 3
Vlera e llogaritur g=5041.67
Vlera e llogaritur h=-112
Press any key to continue
```

Fig.6.10
Rezultati i programit funk10

Emrat e funksioneve mund të zgjedhen edhe të tillë që aspak të mos ndërlidhen me llogaritjet e përfshira brenda tyre.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$z = 4x - 3y + 1$$

ku `x` është numri më i madh mes `3` numrave të dhënë `a`, `b` dhe `c`, kurse `y` e paraqet mbetjen nga pjesëtimi i numrave të plotë `r` dhe `t`. Vlerat e variablave `x` dhe `y` llogariten përmes nënprogrameve `dita` dhe `nata`.

```
// Programi funk11
#include <iostream>
using namespace std;
double dita(double a,double b,double c);
int nata(int r,int t);
int main()
{
      double x,y,z;
      x=dita(57,83,17);
      cout << "\nVlera më e madhe x="
           << x;
      y=nata(9,5);
      cout << "\n\nMbetja nga pjesëtimi y="
           << y;
      z=4*x-3*y+1;
      cout << "\n\nVlera e funksionit z="
           << z
           << "\n\n";
      return 0;
}

// Nënprogrami dita

double dita(double a,double b,double c)
{
```

```
      double x;
      if ((a>b) && (a>c))
            x=a;
      else
            if (b>c)
                  x=b;
            else
                  x=c;
      return x;
}


// Nënprogrami nata

int nata(int r, int t)
{
      int y;
      y=r%t;
      return y;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.6.11*.

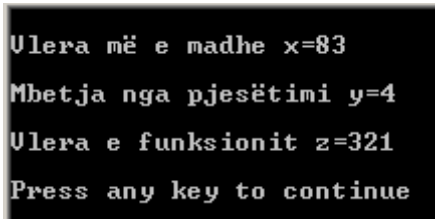


Fig.6.11
Rezultati i programit funk11

Këtu, gjatë thirrjes së funksionit `dita`, parametrat formalë `a`, `b` dhe `c` janë zëvendësuar me vlerat përkatëse `57`, `83` dhe `17`. Kurse, te funksioni `nata`, në vend të parametrave formalë `r` dhe `t` janë shfrytëzuar vlerat `9` dhe `5`.

Shembull Versioni i programit `funk11`, tek i cili vlera e funksionit `z` llogaritet pa i veçuar si funksione të veçanta variablat `x` dhe `y`.

```
// Programi funk11a
#include <iostream>
using namespace std;
double dita(double a,double b,double c);
```

```
int nata(int r,int t);
int main()
{
      double z;

      z=4*dita(57,83,17)-3*nata(9,5)+1;

      cout << "\nVlera e funksionit z="
           << z
           << "\n\n";
return 0;
}
```

Këtu, pjesa e nënprogrameve, që shihen më sipër te programi `funk11`, mbetet e pandryshuar.

Nëse ekzekutohet programi i dhënë, në ekran do të shtypet vetëm rreshti i tretë me rezultate, i cili shihet në *Fig.6.11*.

Vektorët në nënprograme

Brenda nënprogrameve mund të operohet edhe me anëtarë të vektorëve, të cilët paraqiten si parametra formalë.

Llogaritjet me vektorë

Nëse në nënprograme shfrytëzohen vektorët, në titull duhet të deklarohen si parametra formalë, me kllapat e indekseve, pa e dhënë edhe numrin e anëtarëve të tyre.

Shembull

Programi përmes së cilit gjendet shuma e anëtarëve të vektorit `A(m)`, duke e shfrytëzuar gjatë llogaritjes nënprogramin përkatës `vektorSH`.

```
// Programi funkV1
#include <iostream>
#include <iomanip>
using namespace std;
int vektorSH(int X[],int n);
int main()
{
      const int m=5;
      int i,g;
      int A[m]={17,4,12,9,3};
      char t[]="--------------------";
      cout << "\n    i       A[i]\n"
```

```
            << t
            << endl;
      for (i=0;i<m;i++)
             cout << setw(5)
                  << i
                  << setw(10)
                  << A[i]
                  << endl;

      g=vektorSH(A,m);

      cout << t
           << "\n    Shuma:   "
           << g
           << "\n\n";
return 0;
}

// Nënprogrami vektorSH

int vektorSH(int X[],int n)
{
      int i,s;
      s=0;
      for (i=0;i<n;i++)
            s=s+X[i];
return s;
}
```

Siç shihet nga programi i dhënë, prototipi i nënprogramit është deklaruar menjëherë në fillim të programit përmes shprehjes:

```
int vektor(int X[],int n);
```

duke e deklaruar edhe vektorin si parametër formal. Gjatë kësaj, vektori është deklaruar pa e shënuar brenda kllapave të mesme numrin e anëtarëve të tij.

Në programin kryesor, gjatë thirrjes së nënprogramit përmes shprehjes:

```
g=vektorSH(A,m);
```

vektori A paraqitet si parametër aktual, në pozitën e parametrit të parë (në të njëjtën pozitë ku është edhe vektori X si parametër formal), por këtu pa kllapat e mesme të indeksit.

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.6.12*.

```
    i       A[i]
--------------------
    0         17
    1          4
    2         12
    3          9
    4          3
--------------------
    Shuma:   45

Press any key to continue
```

Fig.6.12
Rezultati i programit funkV1

Siç shihet në programin kryesor dhe në nënprogram, gjatë shfrytëzimit të anëtarëve të vektorit, indeksi ndryshon mes vlerës `0` dhe `n-1`, përkatësisht brenda kllapave të unazave `for` vlera fillestare e indeksit është marrë `i=0`, dhe shkon deri te vlera kufitare `i<n`.

Nënprogrami mund të shfrytëzohet edhe për gjetje të anëtarit të caktuar të vektorit.

Shembull

Programi përmes së cilit gjendet anëtari më i madh `x` në vektorin e dhënë `A(m)`, duke e shfrytëzuar nënprogramin `max`.

```
// Programi funkV4
#include <iostream>
#include <iomanip>
using namespace std;
int max(int A[],int m);
int main()
{
      const int m=7;
      char t[]="------------------";
      int i,x,A[m]={5,8,-4,7,14,9,3};
      cout << "\nAnëtarët e vektorit\n"
           << "    i    A[i]\n"
           << t
           << endl;
      for (i=0;i<m;i++)
            cout << setw(5)
                 << i
                 << setw(7)
                 << A[i]
                 << endl;
      cout << t
           << endl;

      x=max(A,m);

      cout << "\nAnëtari më i madh x="
           << x
```

```
          << "\n\n";
return 0;
}

// Nënprogrami max

int max(int A[],int m)
{
      int i,x;
      x=A[0];
      for (i=1;i<m;i++)
            if (A[i]>x)
                  x=A[i];
      return x;
}
```

Këtu, pasi te nënprogrami të gjendet anëtari më i madh x, për t'u përcjellë rezultati i fituar në programin kryesor, variabla `x` është shënuar në vazhdim të komandës `return`. Kurse gjatë shfrytëzimit të funksionit në programin kryesor te shprehja:

```
x=max(A,m);
```

rezultati është ruajtur te variabla e njëjtë (`x`) si edhe në nënprogram, gjë që nuk është e domosdoshme.

Nëse ekzekutohet programi `funkV4`, në ekran do ta kemi pamjen e dhënë në *Fig.6.13*.

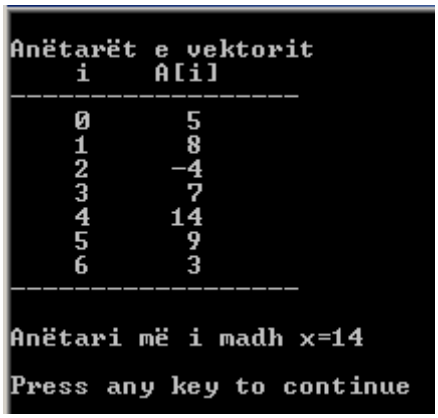


Fig.6.13
Rezultati i programit funkV4

Nëse gjatë thirrjes së nënprogramit, në vend të parametrit aktual `m` shënohet numri `4`, përkatësisht nëse shfrytëzohet shprehja:

```
x=max(A,4);
```

do të fitohet vlera x=8. Kjo është rezultat i asaj se vlera e anëtarit më të madh në këtë rast kërkohet në mesin e 4 anëtarëve të parë të vektorit.

Nënprogrami mund të shfrytëzohet edhe gjatë numërimit të anëtarëve të caktuar të vektorit.

Shembull

Programi përmes së cilit numërohen anëtarët pozitivë të vektorit R(m), të cilët janë më të vegjël se numri pozitiv z. Gjatë numërimit shfrytëzohet nënprogrami numriP.

```
// Programi funkV5
#include <iostream>
#include <iomanip>
using namespace std;
int numriP(int R[],int m,float z);
int main()
{
      const int m=8;
      char t[]="------------------";
      int i,g,R[m]={4,-6,9,3,15,7,5,8};
      float z;
      cout << "Vlera z: ";
      cin >> z;
      cout << "\nAnëtarët e vektorit\n\n"
           << "    i    R[i]\n"
           << t
           << endl;
      for (i=0;i<m;i++)
            cout << setw(5)
                 << i
                 << setw(7)
                 << R[i]
                 << endl;
      cout << t
           << endl;

      g=numriP(R,m,z);

      cout << "\nNumri i kërkuar g="
           << g
           << "\n\n";
return 0;
}
```

```
// Nënprogrami numriP

int numriP(int R[],int m,float z)
{
      int i,g;
      g=0;
      for (i=0;i<m;i++)
            if (R[i]>0)
                  if (R[i]<z)
                        g=g+1;
      return g;
}
```

Brenda unazës së nënprogramit, përmes komandave:

```
if (R[i]>0)
      if (R[i]<z)
            g=g+1;
```

është përcaktuar që nëse njëkohësisht plotësohen dy kushtet e dhëna (anëtari i vektorit është numër pozitiv dhe më i vogël se numri `z`), numëratori g rritet për 1. Komandat në fjalë mund të shkruhen edhe kështu:

```
if ((R[i]>0) && (R[i]<z))
      g=g+1;
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepet vlera hyrëse `z=8`, pas thirrjes së funksionit `numriP` përmes shprehjes:

```
g=numriP(R,m,z);
```

rezultati në ekran do të duket si në *Fig.6.14*.

Fig.6.14
Rezultati i programit funkV5

Rezultat i njëjtë do të fitohet edhe nëse vlera hyrëse `z`, e cila shfrytëzohet si parametër aktualë, shënohet direkt brenda kllapave të funksionit gjatë thirrjes së tij, kështu:

```
g=numriP(R,m,8);
```

Nënprogramet mund të shfrytëzohen edhe për radhitjen sipas madhësive të anëtarëve të vektorëve. Gjatë kësaj, vlerat e anëtarëve te vektori dalës nga nënprogrami do t'i ndryshojnë vendet e tyre.

Shembull

Programi përmes së cilit radhiten sipas madhësisë anëtarët e vektorit `A(m)`, duke shkuar prej vlerës më të madhe kah ajo më e vogla. Gjatë radhitjes shfrytëzohet nënprogrami `radha`.

```
// Programi funkV6
#include <iostream>
#include <iomanip>
using namespace std;
void radha(int A[],int m);
int main()
{
      const int m=8;
      int i,A[m]={4,-6,9,3,15,7,5,8};
      cout << "\nVektori i dhënë\n\n"
           << "A={";
      for (i=0;i<m;i++)
            cout << setw(4)
                 << A[i];
      cout << " }\n";

      radha(A,m);

      cout << "\nVektori i sortuar\n\n"
           << "A={";
      for (i=0;i<m;i++)
            cout << setw(4)
                 << A[i];
      cout << " }\n\n";
return 0;
}

// Nënprogrami radha
```

```
void radha(int A[],int m)
{
      int i,j,x;
      for (i=0;i<m-1;i++)
            for (j=i+1;j<m;j++)
                  if (A[i]<A[j])
                  {
                        x=A[i];
                        A[i]=A[j];
                        A[j]=x;
                  }
return;
}
```

Në nënprogram, gjatë ndërrimit të vendeve të anëtarëve të vektorit është shfrytëzuar variabla ndihmëse `x`, tek e cila ruhet përkohësisht njëra nga vlerat që ndërrohen.

Nëse ekzekutohet programi i dhënë, në ekran do të shihet vektori i dhënë dhe vektori i sortuar, ashtu siç shihet në *Fig.6.15*.

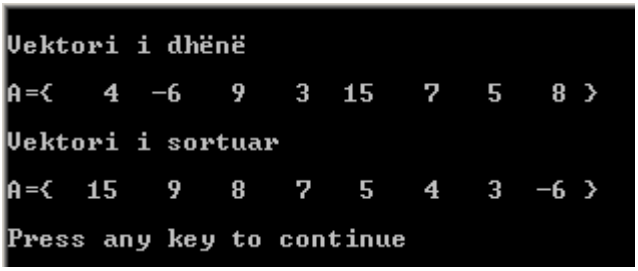


Fig.6.15
Rezultati i programit funkV6

Nënprogrami mund të shfrytëzohet edhe për shtypjen e anëtarëve të vektorit.

Shembull

Programi përmes së cilit gjendet shuma e anëtarëve pozitivë të vektorit `A(m)` dhe e katrorëve të anëtarëve negativë të tij, duke e shfrytëzuar gjatë llogaritjes nënprogramin përkatës `jeta`.

```
// Programi funkV7
#include <iostream>
#include <iomanip>
using namespace std;
int jeta(int A[],int n);
int main()
{
```

```
      const int n=8;
      int s;
      int A[n]={2,-6,4,8,-3,7,-5,9};
      char t[]="------------------";
      cout << "\nAnëtarët e vektorit\n"
             << "\n    i       A[i]\n"
           << t
           << endl;

      s=jeta(A,n);

      cout << t
           << "\n   Shuma:   "
           << s
           << "\n\n";
return 0;
}

// Nënprogrami jeta

int jeta(int A[],int n)
{
      int i,s;

      s=0;
      for (i=0;i<n;i++)
            if (A[i]<0)
                  s=s+A[i]*A[i];
            else
                  s=s+A[i];

      for (i=0;i<n;i++)
            cout << setw(5)
                 << i
                 << setw(10)
                 << A[i]
                 << endl;
return s;
}
```

Siç shihet më sipër, brenda nënprogramit, pas llogaritjes së shumës së kërkuar, përmes unazës përkatëse janë shtypur anëtarët e vektorit, përkatësisht pjesa mes dy rreshtave me viza te tabela e dhënë në *Fig.6.16*.

```
Anëtarët e vektorit

    i       A[i]
------------------
    0          2
    1         -6
    2          4
    3          8
    4         -3
    5          7
    6         -5
    7          9
------------------
   Shuma:   100

Press any key to continue
```

Fig.6.16
Rezultati i programit funkV7

Pjesët e tjera të tabelës, përfshirë edhe rezultatin në fund të saj, janë shtypur në programin kryesor.

Këtu, vlera e shumës është llogaritur përmes shprehjes:

$s=2+(-6)^2+4+8+(-3)^2+7+(-5)^2+9=100$

Përveç anëtarëve të vektorit, në nënprogram mund të shtypen edhe rezultatet që fitohen gjatë llogaritjes, duke i shfrytëzuar vlerat e anëtarëve të tij.

Shembull Versioni i programit `funkV7` tek i cili në nënprogram shtypet edhe rezultati i llogaritjes.

```
// Programi funkV7a
#include <iostream>
#include <iomanip>
using namespace std;
void jeta(int A[],int n,char t[]);
int main()
{
      const int n=8;
      int A[n]={2,-6,4,8,-3,7,-5,9};
      char t[]="-------------------";
      cout << "\nAnëtarët e vektorit\n"
           << "\n    i       A[i]\n"
           << t
           << endl;

      jeta(A,n,t);

return 0;
```

```
}

// Nënprogrami jeta
void jeta(int A[],int n,char t[])
{
      int i,s;

// ------------- llogaritja e shumës ---------

      s=0;
      for (i=0;i<n;i++)
            if (A[i]<0)
                  s=s+A[i]*A[i];
            else
                  s=s+A[i];

// ------------- shtypja e vektorit ----------

      for (i=0;i<n;i++)
            cout << setw(5)
                 << i
                 << setw(10)
                 << A[i]
                 << endl;
      cout << t
           << "\n   Shuma:   "
           << s
           << "\n\n";
return;
}
```

Në këtë version të programit shuma e llogaritur është paraparë të shtypet në nënprogram. Në këtë mënyrë, gjatë shfrytëzimit të funksionit `jeta`, shuma `s` nuk duhet të përcillet në programin kryesor, prandaj edhe thirrja e nënprogramit bëhet thjesht, duke e shënuar emrin e funksionit në program. Si rrjedhojë, para emrit të funksionit është shënuar fjala `void`, kurse te komanda `return` e nënprogramit nuk figuron më variabla e shumës `s`. Meqë në nënprogram është paraparë të shtypet edhe rreshti i fundit me viza, si parametër i tretë brenda kllapave të funksionit është përfshirë edhe variabla `t`, e cila në fakt paraqet një vektor të tipit `char`.

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket plotësisht njëlloj me atë që shihet në *Fig.6.16*.

Llogaritje duke shfrytëzuar disa vektorë

Gjatë llogaritjeve të ndryshme në nënprograme mund të shfrytëzohen njëkohësisht vlerat që merren nga disa vektorë. Nënkuptohet se emrat e vektorëve duhet të paraqiten si parametra brenda kllapave të funksioneve përkatëse.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$z = 3x + 2\sum_{i=1}^{n}(3a_i - b_i^2)$$

ku brenda shumës paraqiten anëtarët e vektorëve `A(n)` dhe `B(n)`. Gjatë llogaritjes së shumës shfrytëzohet nënprogrami `shumaV`.

```
// Programi funkV8
#include <iostream>
#include <iomanip>
using namespace std;
float shumaV(int A[],int B[],int n);
int main()
{
      const int n=7;
      float s,x,y;
      int A[n]={3,6,9,-2,-7,8,5};
      int B[n]={2,7,-4,5,8,-1,3};
      int i;
      char t[]="--------------------------";
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nAnëtarët e vektorëve\n"
           << "\n    i       A[i]      B[i]\n"
           << t
           << endl;

      for (i=0;i<n;i++)
            cout << setw(5)
                 << i
                 << setw(10)
                 << A[i]
                 << setw(10)
                 << B[i]
                 << endl;

      s=shumaV(A,B,n);
```

```
      cout << t
           << "\nVlera e shumës s="
           << s
           << "\n";

      z=3*x+2*s;

      cout << "\nVlera e funksionit z="
           << z
           << "\n\n";

return 0;
}

// Nënprogrami shumaV
float shumaV(int A[],int B[],int n)
{
      int i;
      float s;

      s=0;
      for (i=0;i<n;i++)
            s=s+(3*A[i]-B[i]*B[i]);
return s;
}
```

Në programin e dhënë, gjatë llogaritjes së shumës janë shfrytëzuar anëtarët e dy vektorëve. Prandaj, në nënprogram, si parametra formalë paraqiten vektorët A dhe B. Njëkohësisht, kur thirret funksioni për ta llogaritur shumën, edhe si parametra aktualë paraqiten vektorët në fjalë. Në program dhe në nënprogram shuma është deklaruar e tipit `float`, sepse vlera e saj mund të jetë edhe numër më i madh për vlera të tjera të anëtarëve të vektorëve.

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket si në *Fig.6.17*.

```
Vlera e variablës x: 5.7

Anëtarët e vektorëve

    i       A[i]      B[i]
-------------------------
    0          3         2
    1          6         7
    2          9        -4
    3         -2         5
    4         -7         8
    5          8        -1
    6          5         3
-------------------------
Vlera e shumës s=-102

Vlera e funksionit z=-186.9

Press any key to continue
```

Fig.6.17
Rezultati i programit funkV8

Nëse duam ta kontrollojmë saktësinë e rezultatit të fituar, vlerat duhet të llogariten edhe me dorë. Por, që të humb sa më pak kohë, për testim mund të merren, p.sh., vetëm `3` anëtarët e parë të vektorëve. Për këtë qëllim, programi duhet të ekzekutohet pasi paraprakisht të merren `3` vlera e parametrit `n` gjatë thirrjes së funksionit, kështu:

```
s=shumaV(A,B,3);
```

Plotësisht njëlloj si gjatë shfrytëzimit të dy vektorëve në nënprogram veprohet edhe kur kemi të bëjmë me më shumë vektorë njëkohësisht.

Formimi i vektorit në nënprogram

Si rezultat i një nënprogrami mund të merren edhe anëtarët e vektorit. Gjatë kësaj, shkruarja e vektorit si parametër formal dhe si parametër aktual do të duket ashtu siç u dha më sipër. Por, këtu vektori do të paraqitet edhe në fund të komandës `return`, me kllapat e mesme të indeksit përkatës.

Shembull

Programi përmes së cilit formohet vektori `A(m)`, duke i përcaktuar anëtarët e tij përmes shprehjes:

$$a_i = 3 + 4i$$

```
// Programi funkV2
#include <iostream>
#include <iomanip>
using namespace std;
void formimi(int X[],int n);
int main()
{
      const int m=5;
      int i,A[m];

      formimi(A,m);

      cout << "\nVektori i formuar\n\n"
           << "A={";
      for (i=0;i<m;i++)
            cout << setw(4)
                 << A[i];
```

```
      cout << " }\n\n";
return 0;
}

// Nënprogrami formimi

void formimi(int X[],int n)
{
      int i;
      for (i=0;i<n;i++)
            X[i]=3+4*i;
      return;
}
```

Për formimin e vektorit `A(m)` këtu është shfrytëzuar nënprogrami `formimi`, ku te funksioni përkatës si parametra formalë paraqiten vektori `X` dhe numri i anëtarëve të tij `n`.

Gjatë shfrytëzimit të funksionit në programin kryesor është shkruar komanda:

```
formimi(A,m);
```

Nëse krahasohen parametrat që paraqiten brenda kllapave me ata që janë shënuar në titullin e nënprogramit, do të shihet se parametrat formalë `X` dhe `n` janë zëvendësuar me parametrat aktualë `A` dhe `m`.

Rezultati i cili shtypet në ekran pas ekzekutimit të programit të dhënë do të duket si në *Fig.6.18*.

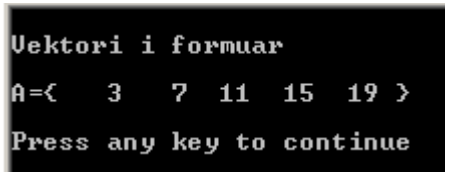


Fig.6.18
Rezultati i programit funkV2

Anëtarët e vektorit që formohet mund edhe të llogariten me shprehje më të përbëra.

Shembull

Programi përmes së cilit formohet vektori `Z(n)`, duke i llogaritur anëtarët e tij përmes shprehjes:

$$z_i = 2x + 4\sum_{k=0}^{i}(k+3)$$

Në program, për formimin e vektorit Z është shfrytëzuar nënprogrami `vektoriZ`. Vlera e variablës `x` kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi funkV3
#include <iostream>
#include <iomanip>
using namespace std;
void vektoriZ(float Z[],int n,float x);
int main()
{
      const int n=5;
      int i;
      float x,Z[n];
      cout << "\nVlera e variablës x: ";
      cin >> x;

      vektoriZ(Z,n,x);

      cout << "\nVektori i formuar\n\n"
           << "Z={";
      for (i=0;i<n;i++)
            cout << setw(7)
                 << Z[i];
      cout << " }\n\n";
return 0;
}

// Nënprogrami vektoriZ

void vektoriZ(float Z[],int n,float x)
{
      int i,k;
      float s;
      for (i=0;i<n;i++)
      {
            s=0;
            for (k=0;k<=i;k++)
                  s=s+(k+3);
            Z[i]=2*x+4*s;
      }
      return;
}
```

Përmes nënprogramit `vektorZ` këtu është përcaktuar formimi i vektorit `Z`, duke mos i dalluar parametrat formalë dhe ata aktualë, të cilët shfrytëzohen gjatë definimit dhe thirrjes së funksionit. Kjo shihet qartë, nëse krahasohet procedura e llogaritjes në nënprogram dhe komanda përmes së cilës thirret funksioni:

```
vektoriZ(Z,n,x);
```

Nëse ekzekutohet programi i dhënë, rezultati i cili shtypet në ekran do të duket si në *Fig.6.19*.

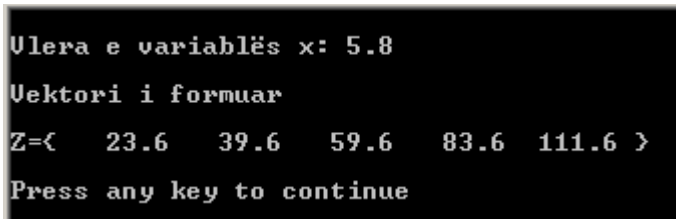


Fig.6.19 Rezultati i programit funkV3

Për t'i kontrolluar vlerat e fituara të anëtarëve të vektorit `Z`, p.sh., le ta marrim llogaritjen e anëtarit të dytë:

```
Z[1]=2*5.8+4*{(0+3)+(1+3)}=39.6
```

Nëse në nënprogramin e dhënë më sipër shfrytëzohen parametra formalë edhe për numrat që paraqiten në shprehjen e dhënë, gjatë shfrytëzimit të tij, nëse ata zëvendësohen me parametra aktualë të ndryshëm, do të fitohen vektorë me vlera të ndryshme të anëtarëve të tyre.

Shembull

Versioni `funkV3a` i programit `funkV3` tek i cili, gjatë formimit të vektorit `Z(n)`, për llogaritjen e anëtarëve të tij shfrytëzohet shprehja e përgjithshme:

$$z_i = ax + b\sum_{k=0}^{i}(k+c)$$

Vetëm vlera e variablës `x` kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi funkV3a
#include <iostream>
#include <iomanip>
using namespace std;
void vektoriZ(float Z[],int n,float x,int a,int b,int c);
int main()
{
      const int n=5;
      int i;
      float x,Z[n];
```

```
      cout << "\nVlera e variablës x: ";
      cin >> x;

      vektoriZ(Z,n,x,2,4,3);

      cout << "\nVektori i formuar\n\n"
           << "Z={";
      for (i=0;i<n;i++)
           cout << setw(4)
                << Z[i];
      cout << " }\n\n";
return 0;
}

// Nënprogrami vektoriZ

void vektoriZ(float Z[],int n,float x,int a,int b,int c)
{
      int i,k;
      float s;
      for (i=0;i<n;i++)
      {
           s=0;
           for (k=0;k<=i;k++)
                s=s+(k+c);
           Z[i]=a*x+b*s;
      }
      return;
}
```

Në nënprogram, brenda kllapave të funksionit, si parametra formalë janë shënuar edhe variablat `a`, `b` dhe `c`. Njëkohësisht, në programin kryesor, gjatë shfrytëzimit të funksionit `vektoriZ`, parametrat formalë janë zëvendësuar me vlerat `2`, `4` dhe `3`, të cilat janë marrë si parametra aktualë përkatës. Kështu, nëse ekzekutohet ky version i programit, për vlerën hyrëse `5.8` të variablës `x`, do të fitohet rezultati i dhënë në *Fig.6.19*.

Meqë nënprogrami është shkruar në një formë më të përgjithshme, ai mund të përdoret për krijimin e vektorëve me vlera të ndryshme të anëtarëve. Kështu, p.sh., nëse funksioni thirret me parametrat aktualë vijues:

```
vektoriZ(Z,n,x,5,2,1);
```

anëtarët e vektorit do të llogariten përmes shprehjes:

$$z_i = 5x + 2\sum_{k=0}^{i}(k+1)$$

sepse parametrat formalë `a`, `b` dhe `c` janë zëvendësuar me parametrat aktualë `5`, `2` dhe `1`.

Por, nëse funksioni thirret kështu:

```
vektoriZ(Z,n,x,3,x+1,2*x);
```

për llogaritje të anëtarëve të vektorit kompjuteri do ta shfrytëzojë shprehjen:

$$z_i = 3x + (x+1)\sum_{k=0}^{i}(k+2x)$$

sepse parametrat formalë `a`, `b` dhe `c` janë zëvendësuar me parametrat aktualë:

```
3
x+1
2*x
```

Funksioni i dhënë, në program mund të thirret edhe disa herë. Por, pas çdo thirrje duhet të shtypen, ose të shfrytëzohen për llogaritje të ndryshme anëtarët e vektorit, sepse me thirrjen vijuese të funksionit vlerat paraprake të anëtarëve të tij mbulohen me vlerat e reja.

Shfrytëzimi i disa vektorëve

Gjatë formimit të një vektori të ri mund të shfrytëzohen anëtarët e disa vektorëve, të cilët duhet të vendosen si parametra brenda kllapave të funksionit përkatës.

Shembull

Programi përmes së cilit nga anëtarët e vektorëve `X(m)` dhe `Y(m)` formohet vektori `Z(m)`, duke i llogaritur anëtarët e tij përmes shprehjes:

$$z_i = \begin{cases} 3x_i + d & \text{për } x_i < y_i \\ x_i - 2y_i & \text{për } x_i \geq y_i \end{cases}$$

Për formimin e vektorit të ri, në program shfrytëzohet nënprogrami `formimiZ`. Vlera e variablës `d` kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi funkV9
#include <iostream>
```

```
#include <iomanip>
using namespace std;
void formimiZ(int X[],int Y[],double Z[],float d,int m);
int main()
{
      const int m=8;
      int X[m]={6,3,-2,5,4,9,1,-7};
      int Y[m]={-2,1,5,8,7,3,4,6};
      int i;
      float d;
      double Z[m];
      char t[]="---------------------------";
      cout << "\nVlera e variablës d: ";
      cin >> d;
      cout << "\nAnëtarët e vektorëve\n"
           << "\n    i       X[i]      Y[i]\n"
           << t
           << endl;

      for (i=0;i<m;i++)
            cout << setw(5)
                 << i
                 << setw(10)
                 << X[i]
                 << setw(10)
                 << Y[i]
                 << endl;

      cout << t
           << "\n\nVektori i formuar"
           << "\n\nZ={";

      formimiZ(X,Y,Z,d,m);

      for (i=0;i<m;i++)
            cout << setw(5)
                 << Z[i];
      cout << " }\n\n";

return 0;
}

// Nënprogrami formimiZ

void formimiZ(int X[],int Y[],double Z[],float d,int m)
{
      int i;
      for (i=0;i<m;i++)
            if (X[i]<Y[i])
```

```
                Z[i]=3*X[i]+d;
            else
                Z[i]=X[i]-2*Y[i];
      return;
}
```

Këtu, brenda kllapave të titullit të nënprogramit, përveç dy vektorëve anëtarët e të cilëve shfrytëzohen gjatë përcaktimit të vlerave të vektorit Z, figuron edhe vetë vektori që formohet. Rezultati i cili fitohet pas ekzekutimit të programit të dhënë do të duket si në *Fig.6.20*.

```
Vlera e variablës d: 5

Anëtarët e vektorëve
    i       X[i]      Y[i]
--------------------------
    0          6        -2
    1          3         1
    2         -2         5
    3          5         8
    4          4         7
    5          9         3
    6          1         4
    7         -7         6
--------------------------

Vektori i formuar

Z={   10    1   -1   20   17    3    8  -16 }

Press any key to continue
```

Fig.6.20 Rezultati i programit funkV9

Matricat në nënprograme

Në nënprograme mund të shfrytëzohen edhe matricat. Gjatë deklarimit të prototipit të funksionit si dhe vetë nënprogramit, në grupin e parametrave formalë vendoset edhe emri i matricës. Por, këtu brenda kllapës së indeksit të dytë të matricës, me të cilin përcaktohet numri i kolonave të saj, duhet të shënohet konstantja përkatëse. Për këtë qëllim, paraprakisht numri i kolonave deklarohet si *konstante simbolike* (ang. symbolic constant) përmes komandës paraprocesorike `#define`, ose përmes deklarimit të zakonshëm `const`.

Shembull Programi përmes së cilit gjendet shuma e anëtarëve të matricës `A(m,n)`.

```
// Programi funkM1
#include <iostream>
#include <iomanip>
using namespace std;
#define n 3
double matrica(int X[][n],int m);
int main()
{
      const int m=4;
      int i,j;
      double r;
      char t[]="---------------";
      int A[m][n]={{5,8,2},
                   {7,3,9},
                   {6,1,5},
                   {3,2,4}};

      cout << "\n  Matrica A\n"
           << t
           << "\n";

      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(4)
                       << A[i][j] ;
            cout << endl;
      }

      cout << t
           << endl;
```

```
      r=matrica(A,m);

      cout << "Shuma e anëtarëve të matricës: "
               << r
               << "\n\n";
      return 0;
}

// Nënprogrami matrica

double matrica(int X[][n],int m)
{
      int i,j;
      double s;
      s=0;
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  s=s+X[i][j];
      }
      return s;
}
```

Këtu, në fillim të programit, numri i kolonave të matricës `n` është deklaruar si konstante përmes komandës paraprocesorike `#define`. Kjo konstante pastaj është shfrytëzuar gjatë deklarimit të prototipit të funksionit si dhe vetë funksionit. Numri i rreshtave të matricës `m` është marrë si parametër formalë i veçantë te prototipi i funksionit dhe gjatë përcaktimit të parametrave të vetë funksionit.

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në *Fig.6.21*.

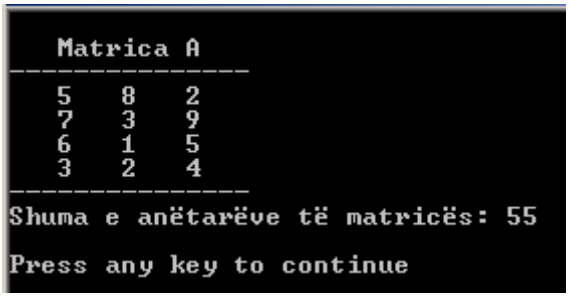


Fig.6.21
Rezultati i programit funkM1

Deklarimi i indeksit të dytë si konstante mund të bëhet edhe përmes komandës `const`, ashtu që rreshtat e parë të programit do të duken kështu:

```
// Programi funkm1
#include <iostream>
using namespace std;
const n=3;
double matrica(int X[][n],int m);
int main()
```

Prej këtu shihet se deklarimi në fjalë është vendosur para deklarimit të prototipit të funksionit dhe, meqë është vendosur edhe para programit kryesor, ky deklarim vlen edhe për programin kryesor.

Nënprogrami mund të shfrytëzohet gjatë formimit të matricave, ngjashëm si edhe gjatë formimit të vektorëve.

Shembull

Programi përmes së cilit formohet matrica `Z(m,n)`, duke i përcaktuar anëtarët e saj përmes shprehjes:

$$z_{ij} = \begin{cases} 2i + j & \text{për } i < j \\ 0 & \text{për } i = j \\ i + 3j & \text{për } i > j \end{cases}$$

Për formimin e matricës `Z` është shfrytëzuar nënprogrami `matricaZ`.

```
// Programi funkM2
#include <iostream>
#include <iomanip>
using namespace std;
const n=4;
void matricaZ(int A[][n],int m);
int main()
{
      const int m=5;
      int i,j;
      char t[]="-------------------";
      int Z[m][n];

      cout << "\n    Matrica e Z\n"
           << t
           << "\n";

      matricaZ(Z,m);
```

```
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(4)
                       << Z[i][j] ;
            cout << endl;
      }

      cout << t
           << "\n\n";

      return 0;
}

// Nënprogrami matricaZ

void matricaZ(int Z[][n],int m)
{
      int i,j;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  if (i<j)
                        Z[i][j]=2*i+j;
                  else
                        if (i==j)
                              Z[i][j]=0;
                        else
                              Z[i][j]=i+3*j;
      return;
}
```

Edhe këtu, gjatë vendosjes së matricës si parametër formal, kompjuterit duhet t'i jepet numri i kolonave n, duke e deklaruar vlerën e saj si konstante para programit kryesor. Nëse ekzekutohet programi i dhënë, matrica e formuar do të duket si në *Fig.6.22*.

```
    Matrica e Z
--------------------
   0   1   2   3
   1   0   4   5
   2   5   0   7
   3   6   9   0
   4   7  10  13
--------------------

Press any key to continue
```

Fig.6.22
Rezultati i programit funkM2

Nënprogrami mund të shfrytëzohet edhe gjatë numërimit të anëtarëve të caktuar në matricës.

Shembull Programi përmes së cilit gjendet numri i anëtarëve pozitivë `p` të matricës së dhënë `Z(m,n)`, duke e shfrytëzuar nënprogramin `matricaP`.

```
// Programi funkM3
#include <iostream>
#include <iomanip>
using namespace std;
const n=5;
int matricaP(int Z[][n],int m);
int main()
{
      const int m=4;
      int i,j,p;
      char t[]="----------------------";
      int Z[m][n]={{5,8,-2,4,7},
                   {4,-3,8,-6,1},
                   {5,6,-1,-2,9},
                   {-2,3,5,-4,6}};

      cout << "\n    Matrica Z\n"
           << t
           << "\n";

      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(4)
                       << Z[i][j] ;
            cout << endl;
      }

      cout << t
           << endl;

      p=matricaP(Z,m);

      cout << "Numri i anëtarëve pozitivë p="
           << p
           << "\n\n";
      return 0;
}

// Nënprogrami matricaP

int matricaP(int Z[][n],int m)
```

```
{
      int i,j,p;
      p=0;
      for (i=0;i<m;i++)
            for (j=0;j<n;j++)
                  if (Z[i][j]>=0)
                        p=p+1;
      return p;
}
```

Në nënprogram, brenda dy unazave, përmes së cilave gjenerohen indekset e anëtarëve të matricës `Z`, është vendosur komanda për degëzim `if`. Përmes kësaj komande, sa herë që anëtari i matricës është numër pozitiv, përmes shprehjes:

```
p=p+1;
```

numëratori i numrave pozitivë rritet për `1`.

Nëse ekzekutohet programi i dhënë, rezultati do të duket si në *Fig.6.23*.

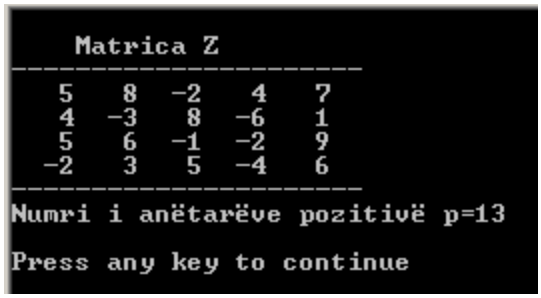


Fig.6.23
Rezultati i programit funkM3

Një funksion mund të thirret edhe disa herë brenda programit kryesor.

Shembull

Programi `funkM` përmes së cilit gjendet shuma e anëtarëve maksimalë të rreshtave të veçantë të matricës së dhënë `A(m,n)`. Për gjetje të anëtarit maksimal `x` në rreshtin e `i`-të të matricës shfrytëzohet nënprogrami `maxA`.

a. Pa nënprogram

```
// Programi funkM
#include <iostream>
#include <iomanip>
using namespace std;
```

```
int main()
{
      const m=5,n=4;
      int A[m][n]={{5,2,14,8},
                   {7,9,1,-6},
                   {8,3,15,4},
                   {12,6,5,-1},
                   {6,4,-5,2}};
      int i,j,x;
      char h[]="------------------------------";
      cout << "\nAnëtarët maksimalë të rreshtave\n"
           << "\n      M a t r i c a        x\n"
           << h
           << endl;

      for (i=0;i<m;i++)
      {
            x=A[i][0];
            for (j=0;j<n;j++)
            {
                  if (x<A[i][j])
                        x=A[i][j];
                  cout << setw(5)
                       << A[i][j];
            }

      cout << setw(8)
           << x
           << endl;
      }
      cout << h
           << endl;
return 0;
}
```

Në program, për çdo rresht, i cili përcaktohet përmes variablës `i` të unazës së jashtme, gjendet anëtari maksimal `x`. Për këtë qëllim, para unazës së brendshme me të cilën ndryshohet indeksi i dytë `j` për anëtarët e kolonave të veçanta, përmes shprehjes:

```
x=A[i][0];
```

si vlerë fillestare e anëtarit maksimal në rreshtin aktualë `i` zgjedhet anëtari i parë në rresht. Pastaj, krahasohet vlera që ruhet te variabla `x` me anëtarët e tjerë të rreshtit në fjalë, duke e ndryshuar indeksin e dytë `j` përmes unazës së brendshme. Në procesin e krahasimit, sa herë që gjendet ndonjë anëtar i matricës me vlerë më të madhe se vlera që ruhet te variabla x, vlera e re ruhet te variabla

në fjalë, duke e mbuluar vlerën paraprake. Për këtë qëllim shfrytëzohen komandat:

```
if (x<A[i][j])
      x=A[i][j];
```

Në fund, pasi të krahasohet vlera që ruhet te variabla `x` me të gjitha vlerat e rreshtit aktual, te kjo variabël do ta kemi vlerën maksimale `x` të rreshtit aktual. Njëkohësisht, me procedurën e gjetjes së anëtarit maksimal, në unazën e brendshme është paraparë që të shtypen anëtarët e rreshtit aktual. Pas daljes nga unaza e brendshme, në vazhdim të anëtarëve të rreshtit aktual shtypet edhe vlera e variablës `x`, tek e cila ruhet vlera e anëtarit maksimal. Kështu, rezulati në ekran do të duket si në *Fig.6.24*.

```
Anëtarët maksimalë të rreshtave
      M a t r i c a          x
-------------------------------
    5    2   14    8        14
    7    9    1   -6         9
    8    3   15    4        15
   12    6    5   -1        12
    6    4   -5    2         6
-------------------------------
Press any key to continue
```

Fig.6.24
Rezultati i programit funkM

b. Me nënprogram

```
// Programi funkM
#include <iostream>
#include <iomanip>
const int n=4;
int maxA(int A[][n],int i,int m);
using namespace std;
int main()
{
      const m=5;
      int A[m][n]={{5,2,14,8},
                   {7,9,1,-6},
                   {8,3,15,4},
                   {12,6,5,-1},
                   {6,4,-5,2}};
      int i,j,x;
      char h[]="-------------------------------";
      cout << "\nAnëtarët maksimalë të rreshtave\n"
           << "\n      M a t r i c a        x\n"
           << h
           << endl;
```

```
      for (i=0;i<m;i++)
      {
            for (j=0;j<n;j++)
                  cout << setw(5)
                       << A[i][j];

            x=maxA(A,i,m);

            cout << setw(8)
                 << x
                 << endl;
      }
      cout << h
           << "\n\n";
return 0;
}

// Nënprogrami maxA

int maxA(int A[][n],int i,int m)
{
      int j,x;
      x=A[i][0];
      for (j=0;j<n;j++)
            if (x<A[i][j])
                  x=A[i][j];
      return x;
}
```

Në nënprogramin e dhënë është përcaktuar procedura e gjetjes së anëtarit maksimal `x`, në rreshtin e `i`-të të matricës `A(m,n)`. Për këtë qëllim, në programin kryesor, funksioni `maxA` thirret për çdo vlerë të variablës `i` të unazës së jashtme, duke e vendosur komandën:

```
x=maxA(A,i,m);
```

brenda kësaj unaze. Edhe në këtë rast, rezultati në ekran do të duket plotësisht njëlloj me atë që u dha më sipër, kur nuk shfrytëzohet nënprogrami.

Funksionet inline

Në gjuhën `C++` shfrytëzohet edhe një formë tjetër e deklarimit të funksioneve përmes komandës `inline`, e cila dallon pak nga forma e deklarimit të funksioneve e cila u përmend më sipër. Deklarimi i funksioneve të tilla bëhet menjëherë në pjesën e parë të programit, duke filluar me komandën `inline`.

Dallimi mes funksioneve `inline` dhe atyre të zakonshme qëndron në atë se:

- gjatë thirrjes së funksioneve të zakonshme, procesi llogaritës përcillet në nënprograme me të cilët ato përcaktohen, për t'u kthyer në fund përsëri në programin kryesor,
- sa herë që thirren funksionet `inline`, kopjet e komandave të tyre vendosen në program, duke u bë kështu pjesë përbërëse e programit.

Prej kësaj që u tha më sipër shihet se gjatë shfrytëzimit të *funksioneve inline* programi rritet, sepse i shtohen kopjet e nënprogrameve që thirren, por fitohet në shpejtësinë e ekzekutimit të tij. Nga ana tjetër, gjatë thirrjes së *funksioneve të zakonshme* kompjuterit i duhet një kohë shtesë për kalim në nënprograme dhe kthim pas në programin kryesor. Praktikohet që si funksione `inline` të deklarohen funksionet që kanë më pak komanda, ashtu që thirrja e tyre të mos ndikojë shumë në madhësinë e programit që ekzekutohet.

Shembull Programi përmes së cilit llogaritet perimetri i katërkëndëshit me brinjët `a` dhe `b`, duke e shfrytëzuar funksionin *inline*.

```
// Programi inline1
#include <iostream>
using namespace std;

// Nënprogrami perimetri

inline float perimetri(float a,float b)
{
      return 2*(a+b);
}

// Programi kryesor

int main()
{
      float a,b,p;
      cout << "\nBrinja a: ";
      cin >> a;
      cout << "\nBrinja b: ";
      cin >> b;

      p=perimetri(a,b);

      cout << "\nPerimetri është p="
           << p
           << "\n\n";
      return 0;
```

```
}
```

Siç shihet nga programi i dhënë, funksioni `inline` është deklaruar në fillim të programit, me formë të njëjtë me funksionet e zakonshme. Thirrja e tij në programin kryesor përmes emrit `perimetri` nuk dallon aspak nga thirrja e funksioneve të zakonshme, gjatë së cilës parametrat formalë `a` dhe `b` janë zëvendësuar me parametrat aktualë përkatës, vlerat e të cilëve kompjuterit i jepen përmes tastierës.

Nëse ekzekutohet programi i dhënë, dhe përmes tastierës kompjuterit i jepen vlerat hyrëse 5 dhr 4, rezultati do të duket si në *Fig.6.25*.

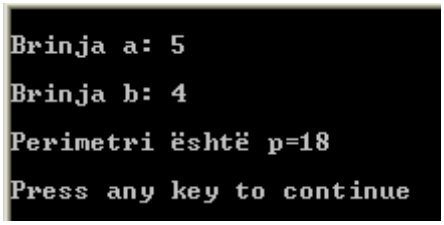


Fig.6.25
Rezultati i programit inline1

Këtu, vlera që llogaritet në nënprogram mund të përcaktohet përmes një variable të veçantë dhe pastaj të shënohet në vazhdim të komandës `return`, p.sh., kështu:

```
// Nënprogrami perimetri

inline float perimetri(float a,float b)
{
      float p;
      p=2*(a+b);
      return p;
}
```

Në një program njëkohësisht mund të shfrytëzohen disa nënprograme `inline`.

Shembull

Programi përmes së cilit llogaritet katrori dhe kubi i numrit `a`, duke shfrytëzuar dy funksione *inline*.

```
// Programi inline2
#include <iostream>
using namespace std;

// Nënprogrami katrori

inline float katrori(float a)
```

```
{
      return a*a;
}

// Nënprogrami kubi

inline float kubi(float a)
{
      return a*a*a;
}

// Programi kryesor

int main()
{
      float a,f,g;
      cout << "\nNumri a: ";
      cin >> a;

      f=katrori(a);

      cout << "\nKatrori i numrit a: "
           << f
           << "\n";

      g=kubi(a);

      cout << "\nKubi i numrit a: "
           << g
           << "\n\n";
      return 0;
}
```

Në fillim të programit, për llogaritjen e katrorit dhe të kubit janë definuar *funksionet inline* përkatëse, `katrori` dhe `kubi`. Kurse, gjatë thirrjes së tyre në programin kryesor, parametri formal `a` është zëvendësuar me numrin `4` si parametër aktual. Pas kësaj, rezultati në ekran do të duket si në *Fig.6.26*.

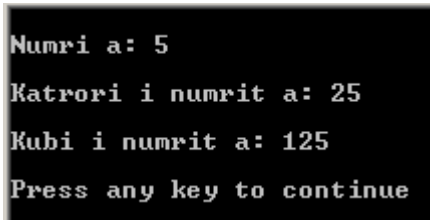


Fig.6.26
Rezultati i programit inline2

Nënprogramet *inline* mund të thirren edhe brenda unazave, ashtu që parametrave formalë t'u ndahen vlera të ndryshme aktuale.

Shembull

Programi tek i cili dy funksionet *inline* të definuara në detyrën paraprake thirren brenda një unaze, për ta fituar tabelën e katrorëve dhe të kubeve të numrave `x` mes vlerave `a` dhe `b`, duke e ndryshuar me hapin `1`.

```
// Programi inline3
#include <iostream>
#include <iomanip>
using namespace std;

// Nënprogrami katrori

inline float katrori(float a)
{
      return a*a;
}

// Nënprogrami kubi

inline float kubi(float a)
{
      return a*a*a;
}

// Programi kryesor

int main()
{
      float a,b,x,f,g;
      char t[]="-------------------------";
      cout << "\nKufiri i poshtëm a: ";
      cin >> a;
      cout << "\nKufiri i sipërm b: ";
      cin >> b;
      cout << "\n\n    x   Katrori    Kubi\n"
           << t
             << endl;
      x=a;
      while (x<=b)
      {

            f=katrori(x);
            g=kubi(x);
```

```
            cout << setw(5)
                 << x
                 << setw(7)
                 << f
                 << setw(10)
                 << g
                 << endl;
            x=x+1;
      }
      cout << t
           << "\n\n";
      return 0;
}
```

Në program, nënprogramet `katrori` dhe `kubi` thirren brenda unazës në të cilën vlerat e variablës së unazës x ndryshohen me hapin `1` mes vlerës fillestare `a` dhe asaj përfundimtare `b`. Rezultati në ekran do të duket si në *Fig.6.27*.

```
Kufiri i poshtëm a: -3

Kufiri i sipërm b: 8

    x   Katrori    Kubi
------------------------
   -3       9      -27
   -2       4       -8
   -1       1       -1
    0       0        0
    1       1        1
    2       4        8
    3       9       27
    4      16       64
    5      25      125
    6      36      216
    7      49      343
    8      64      512
------------------------

Press any key to continue
```

Fig.6.27
Rezultati i programit inline3

Përmes nënprogrameve `inline` mund të përcaktohen edhe funksione më të ndërlikuara.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y = 2x + 3\sum_{i=2}^{n+1}\left\{(2i-1)!+\frac{x}{i}\right\}$$

Vlerat e variablave `x` dhe `n` kompjuterit i jepen si vlera hyrëse përmes tastierës, kurse për llogaritjen e shumës me faktorielin brenda saj shfrytëzohet nënprogrami `shumaF`.

```
// Programi inline4
#include <iostream>
using namespace std;

// Nënprogrami shumaF

inline float shumaF(int n,float x)
{
      int i,k;
      float s,F,y;
      s=0;
      for (i=2;i<=(n+1);i++)
      {
            F=1;
            for (k=1;k<=(2*i-1);k++)
                  F=F*k;
            s=s+(F+x/i);
      }
      y=2*x+3*s;
      return y;
}

// Programi kryesor

int main()
{
      int n;
```

```
      float x,y;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nVlera e variablës x: ";
      cin >> x;

      y=shumaF(n,x);

      cout << "\nVlera e funksionit y="
           << y
           << "\n\n";
return 0;
}
```

Nëse funksioni *inline*, i dhënë më sipër, krahasohet me nënprogramet e zakonshme do të shihet se nuk ka asnjë dallim, sado i ndërlikuar që të jetë ai. Rezultati që fitohet pas ekzekutimit të programit të dhënë, për vlera hyrëse të caktuara, do të duket si në *Fig.6.28*.

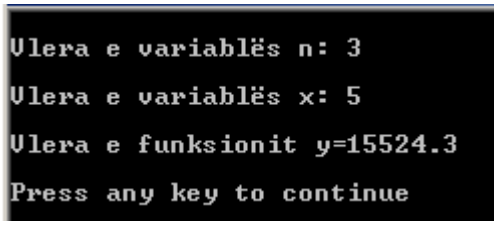


Fig.6.28
Rezultati i programit inline4

Makro funksionet

Duke e shfrytëzuar komandën paraprocesorike `#define`, mund të definohen *makro funksione*, të cilat paraqesin një formë më të thjeshtë të funksioneve. Parametrat që shfrytëzohen në makro funksione nuk shoqërohen edhe me deklaratën përkatëse për tipin e tyre, gjë që është një e metë e funksioneve të deklaruara në këtë mënyrë.

Shembull Programi përmes së cilit tregohet llogaritja e katrorit dhe e kubit të numrit të plotë `a`, duke shfrytëzuar dy *makro funksione*.

```
// Programi makro1
#include <iostream>
using namespace std;

#define katrori(a) (a*a)
#define kubi(a) (a*a*a)
```

```
// Programi kryesor

int main()
{
      float a,f,g;
      cout << "\nNumri a: ";
      cin >> a;

      f=katrori(a);

      cout << "\nKatrori i numrit a: "
           << f
           << "\n";

      g=kubi(a);

      cout << "\nKubi i numrit a: "
           << g
           << "\n\n";
      return 0;
}
```

Për përcaktimin e llogaritjes së katrorit dhe të kubit të numrit a, këtu janë shfrytëzuar dy makro funksione, të vendosura në fillim të programit si komanda paraprocesorike:

```
#define katrori(a) (a*a)
#define kubi(x) (a*a*a)
```

Gjatë shfrytëzimit të tyre në programin kryesor, funksionet në fjalë thirren përmes emrave përkatës, plotësisht njëlloj siç thirren edhe funksionet e zakonshme, duke i zëvendësuar parametrat formalë me ata aktualë.

Nëse ekzekutohet programi i dhënë, për vlerën e variablës a, e cila kompjuterit i jepet përmes tastierës, rezultati që shtypet në ekran do të duket si në *Fig.6.29*.

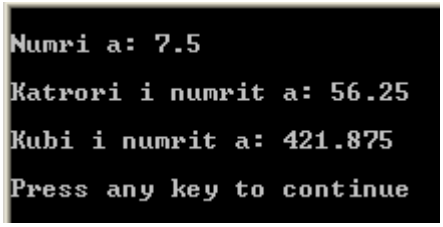


Fig.6.29
Rezultati i programit makro1

Përmes makro funksioneve mund të përcaktohen edhe funksione më komplekse, por definimi i tyre gjithsesi duhet të bëhet me vetëm një shprehje.

Shembull

Programi përmes së cilit llogaritet vlera e funksionit:

$$y=\begin{cases}4x+1 & \text{për } x\le 2\\ x-3 & \text{për } x>2\end{cases}$$

duke e shfrytëzuar makro funksionin `llogaritjaY`.

```
// Programi makro2
#include <iostream>
using namespace std;

#define llogaritjaY(x) ((x<=2) ? (4*x+1):(x-3))

// Programi kryesor

int main()
{
      float x,y;
      cout << "\nVlera hyrëse x: ";
      cin >> x;

      y=llogaritjaY(x);

      cout << "\nVlera e llogaritur y: "
           << y
           << "\n\n";
return 0;
}
```

Në program, gjatë përcaktimit të funksionit y është shfrytëzuar operatori `?:` për llogaritje të kushtëzuar. Siç është shpjeguar edhe në kapitullin e dytë, gjatë shfrytëzimit të operatorit në fjalë, nëse plotësohet kushti `x<=2`, vlera e funksionit llogaritet me shprehjen e parë `4*x+1`, përndryshe për llogaritjen e tij përdoret shprehja e dytë `x-3`.

Nëse programi i dhënë ekzekutohet për një vlerë hyrëse të caktuar, rezultati do të duket si në *Fig.6.30*.

```
Vlera hyrëse x: 1

Vlera e llogaritur y: 5

Press any key to continue
```

Fig.6.30
Rezultati i programit makro2

Funksionet matematikore

Në kuadër të gjuhës C++, ekzistojnë module me shumë funksione të gatshme, të cilët mund të thirren përmes emrave përkatës. Gjatë shfrytëzimit të funksioneve të tilla, prototipet e tyre duhet të deklarohen në program përmes komandave paraprocesorike `#include`.

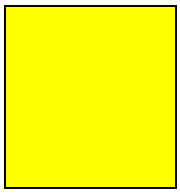
Këtu, fillimisht, do të përmendet moduli `math.h`, në të cilin përfshihen funksione të ndryshme matematikore, ashtu siç shihet në tabelën e dhënë në *Fig.6.31*.

Në matematikë	Në gjuhën C++
sin(x)	sin(x)
cos(x)	cos(x)
tg(x)	tan(x)
arcsin(x)	asin(x)
arccos(x)	acos(x)
arctang(x)	atan(x)
ln(x)	log(x)
log(x)	log10(x)
x^y	pow(x,y)
e^x	exp(x)
$\sqrt{x}$	sqrt(x)
$\lvert x \rvert$	abs(x), fabs(x)
sh(x)	sinh(x)
ch(x)	cosh(x)
th(x)	tanh(x)

Fig.6.31 Tabela e funksioneve matematikore

Parametrat `x` dhe `y` të shënuar brenda kllapave të funksioneve janë vlera të tipit `double`, përveç në rastin e funksionit `abs(x)` përmes së cilit gjendet vlera absolute e variablës `x` të tipit `int`. Gjatë shfrytëzimit të funksioneve trigonometrike vlera e variablës `x` duhet të jepet në radianë.

Shembull Programi përmes së cilit llogaritet vlera numerike e funksionit:



$$y = 3\sin(x+1) + 4e^{x} - 2\log(x+2)$$

ku `x` është anëtari më i madh për nga vlera absolute në vektorin e dhënë `A(m)`.

```
// Programi mat1
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

int main()
{
      const int m=7;
      double y;
      int i,x;
      int A[m]={4,-7,3,-9,5,-3,6};

      cout << "\nVektori i dhënë\n\n"
           << "A={";
      for (i=0;i<m;i++)
            cout << setw(4)
                 << A[i];
      cout << " }\n\n";

// Gjetja e vlerës më të madhe absolute

      x=abs(A[0]);
      for (i=1;i<m;i++)
            if (abs(A[i])>x)
                  x=abs(A[i]);

      cout << "\nVlera më e madhe absolute x="
           << x
           << "\n\n";

      y=3*sin(x+1)+4*exp(x)-2*log(x+2);

      cout << "\nVlera e funksionit y="
           << y
           << "\n\n";
return 0;
}
```

Në program, menjëherë në fillim është vendosur komanda paraprocesorike `#include <math.h>`, për ta njoftuar kompjuterin se do të shfrytëzohen

funksione nga moduli matematikor `math.h`. Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran duket si në *Fig.6.32*.

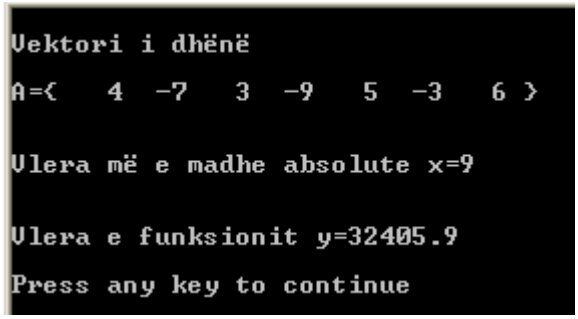


Fig.6.32
Rezultati i programit mat1

Thirrja e funksioneve brenda nënprogrameve

Brenda nënprogramit që shfrytëzohet për definimin e një funksioni të ri nuk mund të definohen funksione të tjera. Por, gjatë definimit të funksioneve mund të thirren funksione të gatshme, të marra nga module të ndryshme me funksione, ose edhe të definuara brenda programit.

Shembull

Programi përmes së cilit llogaritet vlera numerike e shprehjes:

$$z = \frac{x}{3} + 2\sum_{k=1}^{n+1}\{k + 4\cos(3x - 1)\}$$

Për llogaritjen e shumës shfrytëzohet nënprogrami `shumaF`, kurse vlerat e variablave `x` dhe `n` kompjuterit i jepen si vlera hyrëse përmes tastierës.

```
// Programi f1
#include <iostream>
#include <math.h>
double shumaF(int n,float x);
using namespace std;
int main()
{
      float x;
      double z;
      int n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
```

```
      cin >> n;

      z=x/3+2*shumaF(n,x);

      cout << "\nVlera e llogaritur z="
           << z
           << "\n\n";
return 0;
}

// Nënprogrami shuma

double shumaF(int n,float x)
{
      double s;
      int k;
      s=0;
      for (k=1;k<=n+1;k++)
            s=s+(k+4*cos(3*x-1));
      return s;
}
```

Në nënprogram, për llogaritjen e vlerës `cos(3x-1)` është shfrytëzuar funksioni trigonometrik përkatës, i cili merret prej modulit me funksione të ndryshme `math.h`. Pas ekzekutimit të programit të dhënë, për vlera hyrëse të caktuara, rezultati do të duket si në *Fig.6.33*.

```
Vlera e variablës x: 2
Vlera e variablës n: 7
Vlera e llogaritur z=90.821
Press any key to continue
```

Fig.6.33
Rezultati i programit f1

Brenda një nënprogrami mund të thirren edhe funksione të cilat definohen përmes nënprogrameve në kuadër të programit.

Shembull

Programi nga shembulli paraprak, i cili është modifikuar ashtu që për llogaritje dhe për shtypje të vlerës së variablës `z` shfrytëzohet nënprogrami `llogaritjaZ`.

```
// Programi f2
#include <iostream>
#include <math.h>
```

```
double shumaF(int n,float x);
void llogaritjaZ(int n,float x);

using namespace std;
int main()
{
      float x;
      int n;
      cout << "\nVlera e variablës x: ";
      cin >> x;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      llogaritjaZ(n,x);

return 0;
}

// Nënprogrami shumaF

double shumaF(int n,float x)
{
      double s;
      int k;
      s=0;
      for (k=1;k<=n;k++)
            s=s+(k+4*cos(3*x));
      return s;
}

// Nënprogrami llogaritjaZ

void llogaritjaZ(int n,float x)
{
      double z;

      z=x/3+2*shumaF(n,x);

      cout << "\nVlera e llogaritur z="
           << z
           << "\n\n";
      return;
}
```

Nënprogrami i parë `shumaF` ka mbetur i pandryshuar si edhe në shembullin paraprak të programit `f1`. Por, këtu programit i është shtuar edhe nënprogrami i dytë `llogaritjaZ`, tek i cili llogaritet dhe shtypet vlera e variablës z. Gjatë llogaritjes së shumës, brenda nënprogramit të dytë është

shfrytëzuar funksioni `shumaF`, i cili përcaktohet përmes nënprogramit të parë. Meqë rezultati është paraparë të shtypet brenda nënprogramit të dytë, prej tij nuk merret asnjë vlerë dalëse. Prandaj, para emrit të funksionit `llogaritjaZ` në titullin përkatës është shënuar fjala void.

Në programin kryesor është thirrur vetëm funksioni `llogaritjaZ`, me parametrat aktualë `n` dhe `x` brenda kllapave të tij, kurse funksioni `shumaF` është thirrur brenda tij.

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse, që janë marrë në shembullin paraprak, rezultati do të duket si në *Fig.6.33*.

Siç është dhënë edhe më parë, sipas nevojës, nënprogramet mund të thirren brenda një programi edhe disa herë. Kështu, p.sh., nëse te programi `f2`, nënprogrami `llogaritjaZ` thirret `3` herë për parametra aktualë të ndryshëm:

```
llogaritjaZ(n,x);
llogaritjaZ(5,x+2);
llogaritjaZ(2*n,x+0.5);
```

për vlerat hyrëse `2` dhe `7` rezultati do të duket si në *Fig.6.34*.

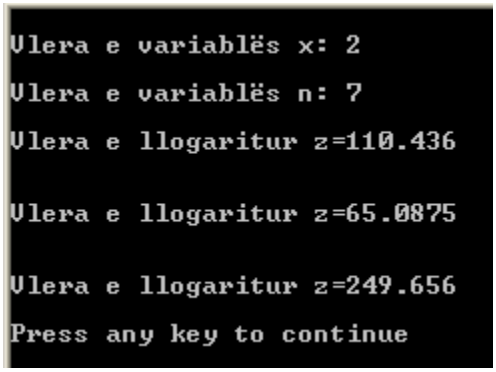


Fig.6.34
Rezultati i versionit të programit f1 tek i cili thirret 3 herë nënprogrami llogaritjaZ

Gjatë 3 thirrjeve në fjalë, parametrat formalë:

```
n           x
```

zëvendësohen me parametrat aktualë:

```
n           x
5           x+2
2*n         x+0.5
```

Funksionet për punë me stringje

Nga grumbulli i funksioneve për punë me stringje, të cilat përfshihen në modulin `string`, përmes shembujve këtu do të tregohet përdorimi i vetëm disa prej tyre. Në fillim të çdo programi vijues është vendosur komanda paraprocesorike `#include <string>`, për ta njoftuar kompjuterin se do të shfrytëzohen funksione të modulit në fjalë.

Gjatësia e stringut

Për përcaktimin e gjatësisë të stringut shfrytëzohet funksioni `strlen(x)`, ku `x` është variabla përmes së cilës ruhet stringu në memorien e kompjuterit.

Shembull Programi përmes së cilit përcaktohet gjatësia e tekstit të lexuar në variablën `A`, për të cilën në memorien e kompjuterit janë rezervuar `20` simbole.

```
// Programi string1
#include <iostream>
#include <string>
using namespace std;

int main()
{
      const m=20;
      char A[m];
      int b;
      cout << "\nShkruaj një fjali: ";

      cin.getline(A,m);

      cout << "\nFjalia që u lexua: "
           << A
           << "\n";

      b=strlen(A);

      cout << "\nGjatësia e fjalisë së lexuar: "
           << b
           << "\n\n";
      return 0;
}
```

Në program, për leximin e fjalisë është shfrytëzuar komanda:

```
cin.getline(A,m);
```

përdorimi i së cilës është shpjeguar në kapitullin e tretë.

Nëse ekzekutohet programi i dhënë më sipër, pasi përmes tastierës të shkruhet fjalia `Koha e bukur`, rezultati do të duket si në *Fig.6.35*.

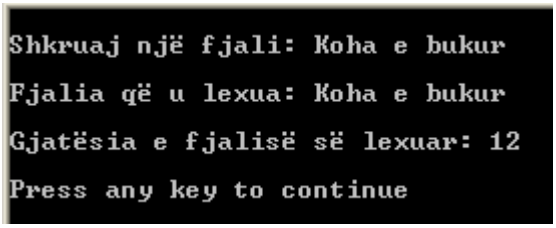


Fig.6.35
Rezultati i programit string1

Kopjimi i stringut

Komplet stringu mund të kopjohet në një string tjetër duke e shfrytëzuar funksionin `strcpy`. Por, ekziston mundësia e kopjimit të vetëm një pjese të stringut përmes funksionit `strncpy`.

Kopjimi i komplet stringut

Stringu `x` mund të kopjohet në stringun `y`, duke e shfrytëzuar funksionin `strcpy(y,x)`.

Shembull

Programi përmes së cilit tregohet kopjimi te variabla `B`, i fjalisë së lexuar, e cila ruhet te variabla `A`.

```
// Programi string2
#include <iostream>
#include <string>
using namespace std;
int main()
{
      const m=20;
      char A[m],B[m];
      cout << "\nShkruaj një fjali: ";

      cin.getline(A,m);

      cout << "\nFjalia që u lexua: "
           << A
           << "\n";
```

```
      strcpy(B,A);

      cout << "\nTeksti i kopjuar: "
           << B
           << "\n\n";
      return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepet teksti `Libri dhe lapsi`, rezultati në ekran do të duket si në *Fig.6.36*.

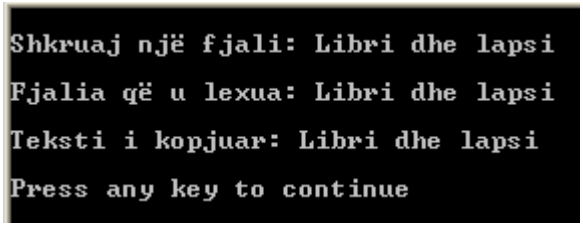


Fig.6.36
Rezultati i programit string2

Teksti që ruhet në një variabël mund të kopjohet në variablën tjetër, pa e shfrytëzuar komandën `strcpy`, njëlloj siç kopjohen vlerat e një vektori në vektorin tjetër.

Shembull Programi përmes së cilit tregohet kopjimi në rrugë të zakonshme te variabla `B`, i fjalisë të lexuar, e cila ruhet te variabla `A`.

```
// Programi string3
#include <iostream>
#include <string>
using namespace std;
int main()
{
      const m=20;
      char A[m],B[m];
      int i,n;
      cout << "\nShkruaj një fjali: ";

      cin.getline(A,m);

      cout << "\nFjalia që u lexua: "
           << A
           << "\n";

      n=strlen(A);
```

```
        for (i=0;i<n;i++)
            B[i]=A[i];
        B[n]='\0';

        cout << "\nTeksti i kopjuar: ";

        for (i=0;i<n;i++)
            cout << B[i];
        cout << "\n\n";
        return 0;
}
```

Në program, pasi lexohet fjalia te variabla A, përmes komandës:

```
n=strlen(A);
```

gjendet gjatësia e saj. Pastaj, duke e shfrytëzuar pjesën e programit:

```
for (i=0;i<n;i++)
    B[i]=A[i];
```

simbolet e veçanta të vektorit A përcillen tek anëtarët përkatës të vektorit B. Në fund, që stringu të kompletohet, përmes komandës:

```
B[n]='\0';
```

atij i shtohet edhe *karakteri zero*, me të cilin përcaktohet plotësisht gjatësia e tekstit të kopjuar.

Nëse ekzekutohet programi `string3` dhe përmes tastierës shkruhet fjalia `Lapsi dhe libri`, rezultati do të jetë i njëjtë me atë që u dha në *Fig.6.36*.

Kopjimi i një pjese të stringut

Përmes funksionit `strncpy(y,x,n)`, te stringu `y` mund të kopjohen vetëm n-simbolet e para të stringut `x`.

Shembull

Programi përmes së cilit tregohet kopjimi te variabla B, i n-simboleve të para të fjalisë së lexuar, e cila ruhet te variabla A.

```
// Programi string4
#include <iostream>
#include <string>
using namespace std;
```

```
int main()
{
      const m=20;
      int n;
      char A[m],B[m];
      cout << "\nShkruaj një fjali: ";

      cin.getline(A,m);

      cout << "\nFjalia që u lexua: "
           << A
           << "\n";
      cout << "\nNumri i simboleve që kopjohen: ";
      cin >> n;

      strncpy(B,A,n);
      B[n]='\0';

      cout << "\nTeksti i kopjuar: "
           << B
           << "\n\n";
      return 0;
}
```

Këtu, pas kopjimit të `n`-simboleve të para të fjalisë, duke e shfrytëzuar funksionin:

```
strncpy(B,A,n);
```

përmes komandës:

```
B[n]='\0';
```

në fund të pjesës së kopjuar të fjalisë vendoset *karakteri zero*.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës, si string hyrës, kompjuterit i jepet fjalia `Libri dhe lapsi`, kurse për numrin `n` të simboleve që kopjohen në tastierë shkruhet numri `7`, rezultati do të duket si në Fig.*6.37*.

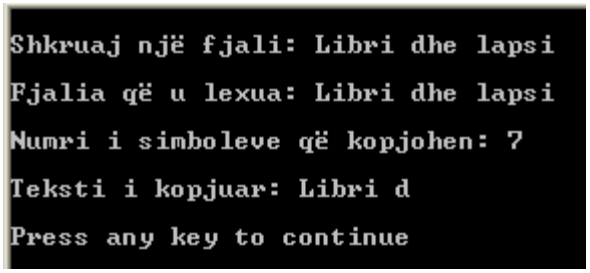


Fig.6.37
Rezultati i programit string4

Një pjesë e caktuar e stringut mund të kopjohet edhe në rrugë të zakonshme, ngjashëm me atë që u shpjegua më sipër.

Shembull Programi përmes së cilit tregohet kopjimi në rrugë të zakonshme te variabla B, i n-simboleve të para të fjalisë së lexuar, e cila ruhet te variabla A.

```
// Programi string5
#include <iostream>
#include <string>
using namespace std;
int main()
{
      const m=20;
      int i,n;
      char A[m],B[m];
      cout << "\nShkruaj një fjali: ";

      cin.getline(A,m);

      cout << "\nFjalia që u lexua: "
           << A
           << "\n";

      cout << "\nNumri i simboleve që kopjohen: ";
      cin >> n;

      for (i=0;i<n;i++)
          B[i]=A[i];
      B[n]='\0';

      cout << "\nTeksti i kopjuar: ";
      for (i=0;i<n;i++)
          cout << B[i];

      cout << "\n\n";
      return 0;
}
```

Programi i dhënë është i ngjashëm me programin `string3`, i cili u dha më sipër. Por, këtu numrin n të simboleve që kopjohen kompjuterit ia japim si vlerë hyrëse përmes tastierës.

Nëse ezekutohet programi `string5`, për të dhënat hyrëse, të cilat kompjuterit iu dhanë gjatë ekzekutimit të programit `string4`, rezultati në ekran do të duket plotësisht njëlloj me atë që shihet në *Fig.6.36*.

Bashkimi i dy stringjeve

Stringut `x` mund t'i shtohet stringu `y`, duke e shfrytëzuar funksionin `strcat(x,y)`.

Shembull Programi përmes së cilit tregohet shtimi i stringut `B` në vazhdim të stringut `A`.

```
// Programi string6
#include <iostream>
#include <string>
using namespace std;
int main()
{
      const m=10,n=25;
      char A[m+n],B[n];
      cout << "\nTeksti i parë A: ";

      cin.getline(A,m);

      cout << "\nTeksti i dytë B: ";

      cin.getline(B,n);

      strcat(A,B);

      cout << "\nTeksti i bashkuar: "
           << A
           << "\n\n";
      return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepen dy tekste, rezultati do të duket si në *Fig.6.38*.

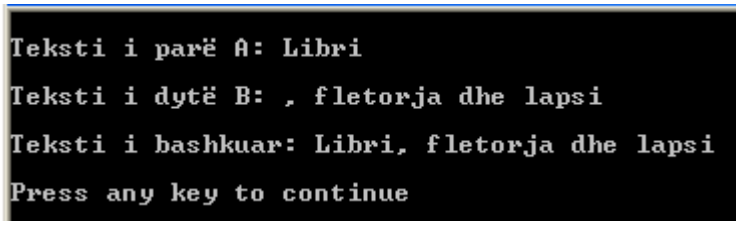


Fig.6.38 Rezultati i programit string6

Si në shembujt paraprakë, edhe këtu mund të shfrytëzohet rruga e zakonshme e shtimit të një stringu në vazhdim të stringut tjetër, pa e shfrytëzuar funksionin `strcat`.

Shembull Programi përmes së cilit tregohet shtimi në rrugë të zakonshme i stringut B në vazhdim të stringut A, pa e shfrytëzuar funksionin `strcat`.

```
// Programi string7
#include <iostream>
#include <string>
using namespace std;
int main()
{
      const m=10,n=25;
      char A[m+n],B[n];
      int a,b,i;
      cout << "\nTeksti i parë A: ";

      cin.getline(A,m);

      cout << "\nTeksti i dytë B: ";

      cin.getline(B,n);

      a=strlen(A);
      b=strlen(B);

      for (i=a;i<=(a+b);i++)
            A[i]=B[i-a];

      cout << "\nTeksti i bashkuar: "
           << A
           << "\n\n";
      return 0;
}
```

Këtu, gjatë shtimit të anëtarëve të stringut B në vazhdim të stringut A, karakteri zero në fund të stringut të ri është shtuar duke e marrë karakterin përkatës i cili gjendet në fund të stringut B.

Shtimi i pjesës së stringut

Për shtimin e n-simboleve të stringut y në vazhdim të stringut x, shfrytëzohet funksioni `strncat(x,y,n)`.

Shembull Programi përmes së cilit tregohet shtimi i k-simboleve të

stringut B në vazhdim të stringut A.

```
// Programi string8
#include <iostream>
#include <string>
using namespace std;
int main()
{
      const m=10,n=25;
      char A[m+n],B[n];
      int k;
      cout << "\nTeksti i parë A: ";

      cin.getline(A,m);

      cout << "\nTeksti i dytë B: ";

      cin.getline(B,n);

      cout << "\nNumri i simboleve që shtohen: ";
      cin >> k;

      strncat(A,B,k);

      cout << "\nTeksti i bashkuar: "
           << A
           << "\n\n";
      return 0;
}
```

Nëse, pas ekzekutimit të programit të dhënë, kompjuterit përmes tastierës i jepen dy tekstet e shfrytëzuara në shembujt paraprakë, dhe numri i simboleve që shtohen merret 7, rezultati do të duket si në *Fig.6.39*.

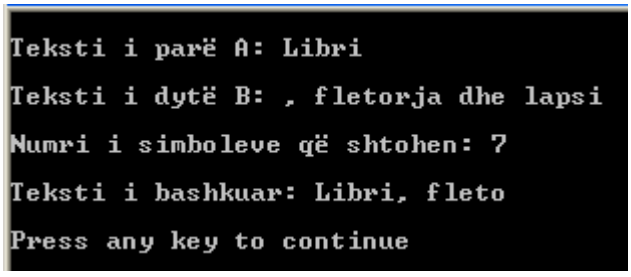


Fig.6.39
Rezultati i programit string8

Nga figura e dhënë shihet se prej tekstit të dytë, në vazhdim të tekstit të parë janë shtuar 7 simbolet e para, përkatësisht janë shtuar simbolet:

```
, fleto
```

Dukshmëria e variablave

Varësisht nga ajo se në cilën pjesë të programit deklarohen variablat, ato mund të jenë *variabla lokale* dhe *variabla globale*.

Variablat lokale

Variablat të cilat përcaktohen brenda një funksioni, përfshirë edhe programin kryesor si funksion, paraqesin variabla lokale. Këto variabla mund të shfrytëzohen vetëm brenda funksionit ku janë definuar, përkatësisht vetëm brenda hapësirës e cila përcaktohet me kllapat e mëdha në fillim dhe në fund të tij.

Shembull Programi përmes së cilit gjendet mbetja nga pjesëtimi i dy numrave të plotë.

```
// Programi lokale
#include <iostream>
using namespace std;
int mbetja(int x,int y);
int main()
{
      int a,b,m;
      cout << "\nVlera e variablës a: ";
      cin >> a;
      cout << "\nVlera e variablës b: ";
      cin >> b;

      m=mbetja(a,b);

      cout << "\nMbetja nga pjesëtimi m="
           << m
           << "\n\n";
      return 0;
}

// Nënprogrami mbetja
int mbetja(int x,int y)
{
```

```
        int z;
        z=x%y;
        return z;
}
```

Në programin e dhënë ka dy grupe variablash lokale. Variablat `x`, `y` dhe `z` janë variabla të cilat duken vetëm brenda dy kllapave të nënprogramit `mbetja`, kurse variablat `a`, `b` dhe `m` duken vetëm brenda programit kryesor. Në pjesën e dukshmërisë së variablave, ato mund të shfrytëzohen për llogaritje, ose vlerat e tyre edhe mund të shtypen. Kështu, nëse ekzekutohet programi i dhënë, për vlerat hyrëse `a=10` dhe `b=4`, rezultati do të duket si në *Fig.6.40*.

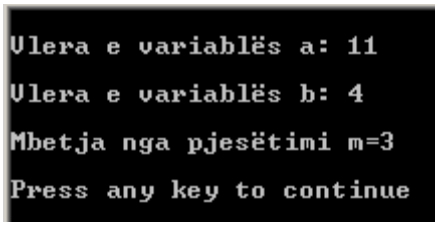


Fig.6.40
Rezultati i programit lokale

Siç shihet nga figura e dhënë, meqë pjesëtimi i numrit `11` me `4` e jep si rezultat numrin `2`, mbetja e pjesëtimit në fjalë është `m=3`.

Në pjesët e tjera të programit, jashtë funksioneve përkatëse, variablat lokale nuk mund të shfrytëzohen. Kështu, p.sh., nëse në programin kryesor të dhënë më sipër duam ta shfrytëzojmë variablën `x`, kompjuteri do të na njoftoi se variabla `x` është identifikator i padeklaruar.

Variablat globale

Variabla të cilat deklarohen para funksionit `main()`, ose jashtë kllapave të funksioneve, paraqesin variabla globale. Variablat globale shihen nga të gjitha nënprogramet, përfshirë edhe programin kryesor, pavarësisht nga ajo se a shfrytëzohen ose jo prej tyre.

Shembull

Programi `globale1`, që është i ngjashëm me programin `lokale`, por tek i cili variablat `a` dhe `b` janë deklaruar para funksionit `main()`.

```
// Programi globale1
#include <iostream>
using namespace std;
int mbetja(int x,int y);
```

```
int a,b;
int main()
{
      int m;
      cout << "\nVlera e variablës a: ";
      cin >> a;
      cout << "\nVlera e variablës b: ";
      cin >> b;

      m=mbetja(a,b);

      cout << "\nMbetja nga pjesëtimi m="
           << m
           << "\n\n";
      return 0;
}
// Nënprogrami mbetja

int mbetja(int x,int y)
{
      int z;
      z=x%y;
      return z;
}
```

Siç shihet nga programi i dhënë, variablat `a` dhe `b` janë deklaruar para programit kryesor dhe paraqesin variabla globale. Prandaj, ato pa asnjë problem shfrytëzohen brenda programit kryesor.

Nëse ekzekutohet programi i dhënë, për vlerat hyrëse `a=11` dhe `b=4`, rezultati në ekran përsëri do të duket si ai që u dha në *Fig.6.40*.

Këtu, variablat `a` dhe `b` pa problem mund të shfrytëzohen edhe brenda nënprogramit `mbetja`. Kështu, p.sh., nëse urdhërohet shtypja e vlerave të këtyre dy variablave brenda nënprogramit, përkatësisht nëse shfrytëzohet versioni i modifikuar i nënprogramit në fjalë:

```
// Nënprogrami mbetja
int mbetja(int x,int y)
{
      int z;
      cout << "\nVlerat e variablave në nënprogram:\n"
           << "\n  a="
           << a
           << "    b="
           << b
           << "\n";
      z=x%y;
      return z;
}
```

rezultati do të duket si në *Fig.6.41*.

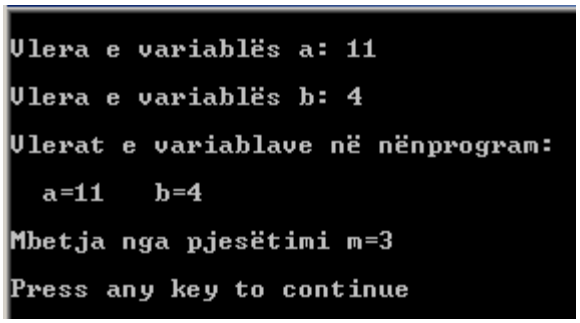


Fig.6.41
Rezultati i programit globale1 pas modifikimit të nënprogramit mbetja

Variablat globale mund të shfrytëzohen edhe gjatë punës me vektorë, me matrica dhe me fusha shumëdimensionale.

Shembull

Programi përmes së cilit formohet vektori `B` nga anëtarët pozitivë të vektorit të dhënë `A(m)`. Në program, për formimin e vektorit shfrytëzohet nënprogrami `pozitivB`.

```
// Programi globale2
#include <iostream>
#include <iomanip>
using namespace std;
void pozitivB(int X[],int Y[],int n);
int k;
int main()
{
      const int n=10;
      int i,A[n]={7,-5,4,-8,2,9,16,-5,13,15},B[n];
      cout << "\nVektori i dhënë\n\n"
           << "A=[";
      for (i=0;i<n;i++)
            cout << setw(4)
                 << A[i];
      cout << " ]\n";

      pozitivB(A,B,n);

      cout << "\nVektori i formuar\n\n"
           << "B=[";
      for (i=0;i<=k;i++)
            cout << setw(4)
                 << B[i];
      cout << " ]\n\n";
```

```
  return 0;
}
// Nënprogrami pozitivB
void pozitivB(int X[],int Y[],int n)
{
      int i;
      k=-1;
      for (i=0;i<n;i++)
            if (X[i]>=0)
            {
                  k=k+1;
                  Y[k]=X[i];
            }
      return;
}
```

Këtu, variabla `k` është deklaruar si variabël globale, sepse komanda për deklarimin e tipit të saj:

```
int k;
```

është vendosur jashtë kllapave të funksionit `main` dhe funksionit `pozitivB`. Por, pas kësaj, variabla `k` shfrytëzohet lirisht në të gjitha pjesët e programit, përkatësisht në programin kryesor dhe në nënprogram.

Nëse ekzekutohet programi `globale2`, rezultati në ekran do të duket si në *Fig.6.42*.

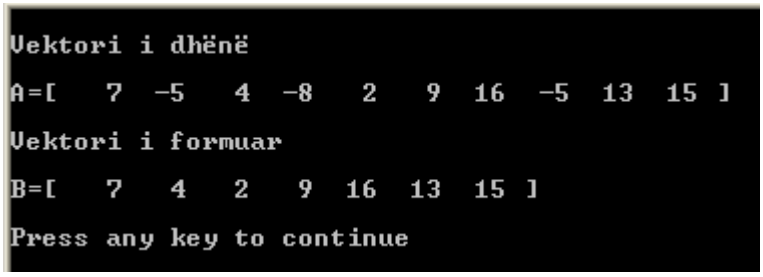


Fig.6.42 Rezultati i programit globale2

Siç shihet nga rezultati, brenda vektorit `B(k)` janë përfshirë vetëm anëtarët pozitivë të vektorit `A(n)`.

Rekursioni

Procesi i thirrjes së një funksioni nga vetvetja, njihet si *thirrje rekursive*, ose thjesht *rekursion*. Në gjuhën programuese `C++` një thirrje e tillë është e lejueshme.

Shembull

Programi për llogaritjen e vlerës së faktorielit:

$$F = n!$$

Në program, llogaritja e faktorielit është përcaktuar përmes një nënprogrami, brenda të cilit shfrytëzohet rekursioni.

```
// Programi rekursion1
#include <iostream>
double fakt(int n);
using namespace std;
int main()
{
      double F;
      int n;
      cout << "\nVlera e variablës n: ";
      cin >> n;

      F=fakt(n);

      cout << "\nFaktorieli F="
           << F
           << "\n\n";
return 0;
}

// Nënprogrami fakt

double fakt(int n)
{
      double F;
      if(n<=1)
            F=1;
      else
            F=n*fakt(n-1);
      return F;
}
```

Në nënprogram, për përcaktimin e llogaritjes së faktorielit shfrytëzohet funksioni rekursiv përkatës:

$$n! = \begin{cases} 1 & \text{për } n \le 1 \\ n \cdot (n-1)! & \text{për } n > 1 \end{cases}$$

Gjatë llogaritjes së faktorielit, pasi kompjuteri në anën e djathtë të shprehjes për llogaritjen e vlerës `F` të faktorielit e has funksionin `fakt(n-1)`,

fillon me thirrjet rekursive, duke e zvogëluar parametrin `n` për `1` gjatë çdo thirrje vijuese. Nëse, p.sh., parametri `n` në thirrjen fillestare merret `5`, zinxhiri i thirrjeve rekursive të funksionit `fakt` do të duket si në vijim.

Fillimisht, kompjuteri tenton që ta llogarisë vlerën:

```
5*fakt(4)
```

Por, meqë mungon vlera `fakt(4)`, në tentimin vijues mundohet që atë ta llogarisë përmes shprehjes:

```
4*fakt(3)
```

Në këtë mënyrë, vazhdon zinxhiri i tentimeve për llogaritje:

```
3*fakt(2)
  2*fakt(1)
```

gjersa nuk e gjen vlerën `1` për `fakt(1)`, kur edhe përfundon zinxhiri i tentimeve. Pamja komplete e këtij zinxhiri duket kështu:

```
5*fakt(4)
  4*fakt(3)
    3*fakt(2)
      2*fakt(1)
        1
```

Pasi të arrihet në fund të zinxhirit, duke shkuar prej fundit kah fillimi, kompjuteri i llogarit vlerat e anëtarëve të zinxhirit në fjalë:

```
        1
      2*1
    3*2*1
  4*3*2*1
5*4*3*2*1
```

Nëse ekzekutohet programi `rekursioni1`, rezultati do të duket si në *Fig.6.43*.

Fig.6.43
Rezultati i programit rekursion1

Procedura e llogaritjes, e cila zbatohet gjatë shfrytëzimit të funksionit `fakt`, në program mund të qartësohet, nëse në brendinë e nënprogramit vendoset komanda për shtypje të vlerave të variablave përkatëse.

Shembull Versioni i programit `rekursion1`, tek i cili brenda nënprogramit është urdhëruar shtypja e vlerave të variablave `n` dhe `F`.

```
// Programi rekursion1a
#include <iostream>
#include <iomanip>
double fakt(int n);
using namespace std;
int main()
{
      double F;
      int n;
      char t[]="------------------";
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nVlerat brenda funksionit\n"
           << "\n    n          F\n"
           << t
           << endl;

      F=fakt(n);

      cout << t
           << "\nFaktorieli F="
           << F
           << "\n\n";
return 0;
}

// Nënprogrami fakt

double fakt(int n)
{
      double F;
      if(n<=1)
            F=1;
      else
            F=n*fakt(n-1);
      cout << setw(5)
           << n
           << setw(11)
           << F
```

```
            << endl;
      return F;
}
```

Nëse ekzekutohet programi i dhënë, rezultati do të duket si në *Fig.6.44*.

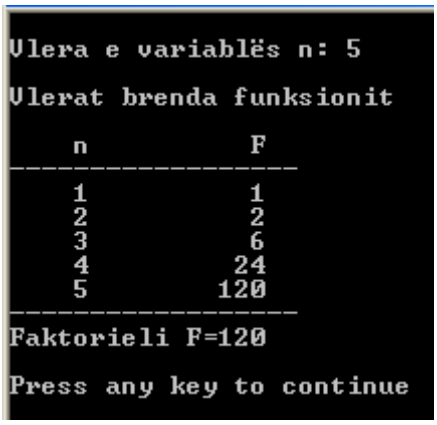


Fig.6.44
Rezultati i programit rekursion1a

Nga tabela e shtypur shihet qartë se si kompjuteri e llogarit vlerën e faktorielit, duke u nisur me faktorielin për vlerën `n=1`.

Shembull

Programi për llogaritjen e shumës së numrave natyrorë mes `0` dhe `n`, duke e shfrytëzuar nënprogramin `shumaN`, tek i cili shfrytëzohet thirrja rekursive e tij. Vlera e variablës `n` kompjuterit i jepet si vlerë hyrëse përmes tastierës.

```
// Programi rekursion2
#include <iostream>
double shumaN(int n);
using namespace std;
int main()
{
      double s;
      int n;
      cout << "\nVlera e variablës n: ";
      cin >> n;
      s=shumaN(n);
      cout << "\nShuma s="
           << s
           << "\n\n";
return 0;
}

// Nënprogrami shumaN
```

```
double shumaN(int n)
{
      double s;
      if (n<=0)
          s=0;
      else
          s=n+shumaN(n-1);
      return s;
}
```

Në nënprogram, llogaritja e shumës është përcaktuar duke e shfrytëzuar funksionin rekursiv përkatës:

$$s = \begin{cases} 0 & për\ n \le 0 \\ n + \sum_{i=0}^{n-1} i & për\ n > 0 \end{cases}$$

Nëse ekzekutohet programi i dhënë, për vlerën hyrëse `n=5`, rezultati do të duket si në *Fig.6.45*.

```
Vlera e variablës n: 5
Shuma s=15
Press any key to continue
```

Fig.6.45
Rezultati i programit rekursion2

Këtu, zinxhiri i tentimeve të veçanta për llogaritjen e shumës është:

```
5+shumaN(4)
  4+shumaN(3)
    3+shumaN(2)
      2+shumaN(1)
        1+shumaN(0)
          0
```

Pastaj, duke shkuar prej fundit, kompjuteri e llogarit vlerën e shumës në bazë të zinxhirit:

```
          0
        1+0
      2+1+0
```

```
          3+2+1+0
        4+3+2+1+0
      5+4+3+2+1+0
```

Për ta ndjekur procedurën të cilën e zbaton kompjuteri gjatë llogaritjes së shumës, në brendinë e nënprogramit mund të vendoset komanda për shtypje të vlerave të variablave përkatëse.

Shembull Versioni i programit `rekursion2`, tek i cili brenda nënprogramit është urdhëruar shtypja e vlerave të variablave `n` dhe `s`.

```
// Programi rekursion2a
#include <iostream>
#include <iomanip>
double shumaN(int n);
using namespace std;
int main()
{
      double s;
      int n;
      char t[]="----------------";
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nVlerat brenda funksionit\n"
           << "\n    n       s\n"
           << t
           << endl;

      s=shumaN(n);

      cout << t
           << "\n   Shuma s="
           << s
           << "\n\n";
return 0;
}

// Nënprogrami shumaN

double shumaN(int n)
{
      double s;
      if (n<=0)
            s=0;
      else
          s=n+shumaN(n-1);
```

```
        cout << setw(5)
             << n
             << setw(8)
             << s
             << endl;
        return s;
}
```

Nëse ekzekutohet ky version i programit, rezultati do të duket si në *Fig.6.46*.

Fig.6.46
Rezultati i programit rekursion2a

Me një ndryshim të vogël te nënprogrami `shumaN` që u dha më sipër, funksioni përkatës mund të shfrytëzohet për llogaritjen e shumës së numrave natyrorë tek ose çift.

Shembull

Programi `rekursion2b`, përmes së cilit llogaritet shuma e numrave natyrorë tek ose çift mes vlerave `0` dhe `n`, varësisht nga vlera hyrëse për variablën `n`, e cila kompjuterit i jepet përmes tastierës.

```
// Programi rekursion2b
#include <iostream>
#include <iomanip>
double shumaK(int n);
using namespace std;
```

```
int main()
{
      double s;
      int n;
      char t[]="----------------";
      cout << "\nVlera e variablës n: ";
      cin >> n;
      cout << "\nVlerat brenda funksionit\n"
           << "\n    n       s\n"
           << t
           << endl;

      s=shumaK(n);

      cout << t
           << "\n   Shuma s="
           << s
           << "\n\n";
return 0;
}

// Nënprogrami shumaK

double shumaK(int n)
{
      double s;
      if (n<=0)
            s=0;
      else
          s=n+shumaK(n-2);
      cout << setw(5)
           << n
           << setw(8)
           << s
           << endl;
      return s;
}
```

Në këtë version të programit, llogaritja e shumës në nënprogramin `shumaK` është përcaktuar me funksionin rekursiv:

$$s = \begin{cases} 0 & \text{për } n \le 0 \\ n + \sum_{i=0}^{n-2} i & \text{për } n > 0 \end{cases}$$

Nëse ky funksion krahasohet me funksionin që u shfrytëzua te nënprogrami `shumaN`, i cili u dha më parë, vërehet vetëm një dallim i vogël te kufiri i sipërm në simbolin e shumës, ku në vend të vlerës `n-1` paraqitet kufiri `n-2`.

Këtu, brenda nënprogramit `shumaK` është paraparë që të shtypen shumat parciale në procesin e llogaritjes rekursive të saj. Rezultati që fitohet pas ekzekutimit të programit `rekursion2b`, për vlerën hyrëse `n=10`, është dhënë në *Fig.6.47*.

```
Vlera e variablës n: 10

Vlerat brenda funksionit

    n        s
----------------
    0        0
    2        2
    4        6
    6       12
    8       20
   10       30
----------------
   Shuma s=30

Press any key to continue
```

Fig.6.47
Rezultati i programit rekursion2b, në rastin kur vlera e variablës n është numër çift

Këtu, meqë vlera hyrëse për variablën `n` është numër çift, përmes nënprogramit `shumaK` llogaritet shuma e numrave natyrorë çift. Por, p.sh., nëse përmes tastierës kompjuterit i jepet numri tek `9`, me nënprogramin në fjalë llogaritet shuma e numrave natyrorë tek. Rezultati në këtë rast do të duket si në *Fig.6.48*, ku në zinxhirin e llogaritjes rekursive, gjatë shkuarjes prej fundit kah fillimi, te vlera fillestare 0 e shumës arrihet për vlerë negative të variablës `n`.

```
Vlera e variablës n: 9

Vlerat brenda funksionit

    n        s
----------------
   -1        0
    1        1
    3        4
    5        9
    7       16
    9       25
----------------
   Shuma s=25

Press any key to continue
```

Fig.6.48
Rezultati i programit rekursion2b, në rastin kur vlera e variablës n është numër tek

Llogaritja rekursive mund të shfrytëzohet edhe gjatë shumë llogaritjeve të ndryshme.

Shembull Programi `rekursion3`, përmes së cilit llogaritet shuma e anëtarëve të vektorit `A(n)`.

```
// Programi rekursion3
#include <iostream>
#include <iomanip>
double shumaV(int A[],int n);
using namespace std;
int main()
{
      const int n=8;
      double s;
      int A[n]={5,-1,3,8,4,2,9,6};
      char t[]="----------------";

      cout << "\nLlogaritja rekursive e shumës\n"
           << "\n    n       s\n"
           << t
           << endl;

      s=shumaV(A,n);

      cout << t
           << "\nShuma e anëtarëve të vektorit s="
           << s
           << "\n\n";
return 0;
}

// Nënprogrami shumaV

double shumaV(int A[],int n)
{
      double s;
```

```
        if (n<=0)
            s=0;
        else
            s=A[n-1]+shumaV(A,n-1);

        cout << setw(5)
             << n
             << setw(8)
             << s
             << endl;
        return s;
}
```

Rezultati i programit `rekursion3` në ekran do të duket ashtu siç është dhënë në *Fig.6.49*.

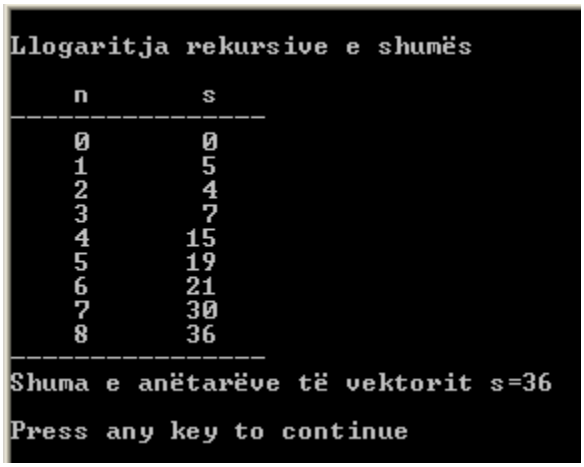


Fig.6.49
Rezultati i programit rekursion3

Procedura që shfrytëzohet gjatë llogaritjes rekursive të shumës edhe këtu është e ngjashme me atë që u shpjegua më sipër. Kjo shihet edhe nga vlerat parciale të shumave të cilat janë shtypur, meqë komanda për shtypje të tyre është vendosur brenda nënprogramit.

Literatura

1. Chris H. Pappas, William H. Murray
The Complete Reference, Visual C++.Net
McGraw-Hill/Osborne, New York 2002

2. Ulka Kirch-Prinz, Peter Prinz
Programming in C++, A Complete Guide
Jones and Bartlett Publishers, Sudbury, USA 2002

3. Julian Templeman, Andy Olsen
Microsoft Visual C++.NET, Step by Step
Microsoft Corporation, Redmond, Washington 2002

4. Victor Shtern
Core C++, A Software Engineering Approach
Prentice Hall PTR, New Jersey 2000

5. Robert Lafore
Object-Oriented Programming in C++
SAMS, Indianopolis, Indiana 1999

6. Bjarne Stroustrup
C++ Programming Language
Addison-Wesley Publishing Company, Massachusetts 1997

7. H.M. Deitel, P. J. Deitel
How to Program C
Prentice Hall, Englewood Cliffs, New Jersey 1994

8. Jesse Libery
Teach Yourself C++ in 21 Days
Sams Publishing, Indianapolis, Indiana

9. S. B. Lippman, J. Lajoie
C++ Primer
Addison-Wesley Publishing Company, Massachusetts

10. Kris Jamsa, Lars Klander
C/C++ Programmer's Bible
Gulf Publishing Company, Houston, Texas

11. Rob McGregor
Practical C++
QUE, Indianapolis, Indiana 1999

12. H.M. Deitel, P. J. Deitel
How to Program C++
Prentice Hall, Upper Saddle River, New Jersey 2003

13. H.M. Deitel, P. J. Deitel
How to Program C
Prentice Hall, Upper Saddle River, New Jersey 2004

14. Jim Keogh, John Shapley Gray
C++ Programer's Notebook
Prentice Hall PTR, Upper Saddle River, New Jersey 2002

15. Agni H. Dika
Algoritmet, njohuri themelore, me programe në C++
Fakulteti Elektroteknik, Prishtinë 2002, 2004

16. James P. Cohoon, Jack W. Davidson
C++ Program Design
Irwin/McGraw-Hill, USA 1997

17. Agni Dika
Bazat e programimit në C++
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike, Prishtinë 2004

18. D. S. Malik
C++ Programming: From Problem Analysis to Program Design
Course Technology, Massachusetts 2002

19. Frank L. Friedman, Elliot B. Koffman
Problem Solving, Abstarction, and Design Using C++
Pearson Addison Wesley, USA 2004

20. Stanley B. Lippman, Josée Lajoie, Barbara E. Moo
C++ Primer, Fourth Edition
Addison-Wesley, USA 2005

21. Herbert Schildt
C++ from the Ground Up
McGraw-Hill/Osborne, Berkeley, California 2003

Shtesa A

Fjalët të cilat shfrytëzohen nga gjuha C++ nuk mund të përdoren si variabla ose emra funksionesh. Lista e këtyre fjalëve nuk është fikse sepse te kompajler të ndryshëm kjo listë mund të ndryshojë.

Në tabelën vijuese është dhënë një listë e fjalëve të rezervuara nga gjuha C++, e cila llogaritet si standarde.

asm	else	new	this
auto	enum	operator	throw
bool	explicit	private	true
break	export	protected	try
case	extern	public	typedef
catch	false	register	typeid
char	float	reinterpret_cast	typename
class	for	return	union
const	friend	short	unsigned
const_cast	goto	signed	using
continue	if	sizeof	virtual
default	inline	static	void
delete	int	static_cast	volatile
do	long	struct	wchar_t
double	mutable	switch	while
dynamic_cast	namespace	template	

Këtu, fjalët e shënuara në fushat e nxiera janë fjalë të rezervuara në gjuhën C++, por jo edhe në gjuhën C. Kurse fjalët në fushat tjera, njëkohësisht janë të rezervuara në gjuhët C dhe C++.

Shtesa B

Në gjuhën C++ gjatë shtypjes së rezultateve shfrytëzohen kombinime të ndryshme të simboleve, ose siç quhen ndryshe - sekuenca dalëse. Disa nga këto sekuenca janë dhënë në tabelën vijuese.

Sekuenca	Karakteri	Vlera decimale
'\0'	zero	0
'\a'	i ziles	7
'\b'	për kthim pas	8
'\t'	për tabelim horizontal	9
'\n'	për rresht të ri	10
'\v'	për tabelim vertikal	11
'\f'	për lëvizje të letrës	12
'\r'	për kthim në fillim të rreshtit	13
'\"'	për thonjëz të dyfishtë	34
'\''	për thonjëz të njëfishtë	39
'\\'	për vijë të pjerrët mbrapshtë	92

Te sekuencat dalëse, në vend të thonjëzës së njëfishtë ('), plotësisht njëlloj mund të shfrytëzohet edhe thonjëza e dyfishtë (").

Universiteti i Europës Juglindore
Fakulteti i Shkencave dhe i Teknologjive të Komunikimit

Agni Dika
Bazat e Programimit në C++

Lektor
Dr. Ilaz Metaj

Kopertina
AfiDesign

Shtypi
Arbëria Design
Tetovë

Katalogimi në publikim - CIP
Biblioteka Popullore dhe Universitare "Sv. Kliment Ohridski", Shkup

CIP - Katalogizimi në publikim
Biblioteka popullore dhe universitare "Sv. Kliment Ohridski:,
Shkup
004.432
DIKA, Agni
Bazat e programimit në C++/Agni Dika
400 faqe, 24cm

ISBN 9989-866-23-6

a) C++ (gjuhë programuese)
COBISS.MK-ID 62677770