Universiteti i Prishtinës
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Agni H. Dika

Algoritmet
njohuri themelore
me programe në C++

Prishtinë 2007

Parathënie

Libri të cilin e keni në dorë së pari u dedikohet studentëve të Fakultetit të Inxhinierisë Elektrike dhe Kompjuterike në Prishtinë. Por, ai mund të përdoret edhe nga të interesuar tjerë për programim me kompjuter.

Njësitë mësimore në libër janë organizuar ashtu që materiali i përfshirë brenda tyre të jetë sa më i afërt për të gjithë ata të cilët fillojnë të punojnë me kompjuter. Për këtë qëllim, gjatë shpjegimit të algoritmeve janë shfrytëzuar shembuj të ndryshëm, duke filluar nga ato elementare.

Në libër, çdo algoritmi i shoqërohet edhe programi përkatës i shkruar në gjuhën programuese C++, ashtu që përmes ekzekutimit në kompjuter, të jetë edhe më i qartë funksionimi i tyre. Lexuesit të cilët nuk kanë njohuri mbi gjuhën programuese C++, pa asnjë pengesë mund t'i kapërcejnë programet që paraqiten në libër.

Autori

Paraqitja e algoritmeve

Grumbulli i veprimeve me një radhë të fiksuar, të cilët ndërmerren gjatë zgjidhjes së një problemi të caktuar, quhet **algoritëm**.

Në jetën e përditshme, për zgjidhjen e problemeve të ndryshme, njeriu krijon algoritme përkatëse, duke shfrytëzuar dijen e grumbulluar. Por, me kohë, algoritmet që përsëriten fiksohen në ndërdije dhe sipas nevojës shfrytëzohen si të gatshme. Kështu, p.sh., ardhja para rrugëkryqit, çdo këmbësori i imponon përdorimin e algoritmit, i cili mund të përshkruhet përmes tekstit të dhënë në *Fig.1.1*.

```
   Nëse në rrugëkryq është vendosur semafori
dhe ai punon, rruga mund të kalohet në
vendëkalim pasi të paraqitet ngjyra e
gjelbër. Nëse në rrugëkryq nuk ka semafor,
ose ai nuk punon, rruga mund të kalohet në
vendkalim kur nuk ka automjete, duke shikuar
majtas dhe djathtas.
```

Fig.1.1 Algoritmi logjik për kalimin e rrugëkryqit

Varësisht nga operacionet që përdoren gjatë përpilimit, algoritmet mund të grupohen në **algoritme logjike** dhe **algoritme numerike**. Derisa algoritmet logjike mbështeten në operacione dhe konkluzione logjike, ashtu siç shihet në shembullin e algoritmit të dhënë më sipër, në algoritmet numerike shfrytëzohen operacionet dhe shprehjet aritmetikore.

Paraqitja analitike

Në formë më të lirë, paraqitja analitike e algoritmeve duket ashtu siç është dhënë në shembullin e algoritmit për kalimin e rrugëkryqit, në *Fig.1.1*. Kjo formë e paraqitjes së algoritmeve, nëse përdoret gjatë zgjidhjes së problemeve të komplikuara, mund të jetë e paqartë dhe e papërcaktuar plotësisht.

Në praktikë përdoret forma analitike e paraqitjes së algoritmeve e shprehur përmes një numri hapash, të cilët kryhen sipas një radhe të fiksuar plotësisht. Kështu, shembulli i algoritmit të dhënë në *Fig.1.1*, i paraqitur në `10` hapa, do të duket si në *Fig.1.2*.

```
1.  Fillimi
2.  A ka semafor?
     Nëse JO, hapi i 6.
3.  A punon semafori?
     Nëse JO, hapi i 6.
4.  A është paraqitur ngjyra e gjelbër?
     Nëse PO, hapi i 9.
5.  Duhet pritur. Hapi i 4.
6.  Shiko majtas e djathtas
7.  A ka automjete?
     Nëse JO, hapi i 9.
8.  Duhet pritur. Hapi i 6.
9.  Kalo rrugën në vendëkalim
10. Fundi.
```

Fig.1.2 Forma analitike e algoritmit për kalimin e rrugëkryqit

Hapat e veçantë të këtij algoritmi kryhen me radhë prej fillimi, derisa nuk urdhërohet kapërcimi në një hap të caktuar. Kështu, p.sh., nëse në rrugëkryq ka semafor dhe paraqitet ngjyra e gjelbër, vargu i hapave nëpër të cilët do të kalohet gjatë ekzekutimit të algoritmit të dhënë është: **1**, **2**, **3**, **4**, **9** dhe **10**. Por, në kushtet e kulturës së komunikacionit në hapësirat tona, kur shpesh ndodh që automjetet e kalojnë rrugëkryqin kur në semafor është e ndezur drita e kuqe, më e sigurt për këmbësorin është nëse hapi i katërt i algoritmit shkruhet kështu:

```
4.  A është paraqitur ngjyra e gjelbër?
     Nëse PO, hapi i 6.
```

ashtu që para se të kalohet rrugëkryqi, pavarësisht se për këmbësor është paraqitur ngjyra e gjelbër, duhet të shikohet mos ndoshta kalon ndonjë automjet. Kështu, për rastin e përmendur më sipër, kur në rrugëkryq ka semafor dhe është paraqitur ngjyra e gjelbër, vargu i hapave nëpër të cilët kalohet është: **1**, **2**, **3**, **4**, **6**, **7**, **9** dhe **10**.

Tek algaritmet numerike hapat e veçantë janë më të qartë, sepse konsistojnë në operacione dhe në shprehje matematikore.

Shembull Algoritmi numerik për llogaritjen e vlerës së funksionit:

$$y = \begin{cases} x^2 & \text{për } x < 0.9 \\ 2x & \text{për } x = 0.9 \\ x - 3 & \text{për } x > 0.9 \end{cases}$$

nëse dihet vlera e variablës `x`.

```
1.  Fillimi.
2.  Merre vlerën e variablës x
3.  A është x<0.9, ose x=0.9, ose x>0.9?
     Nëse x<0.9, hapi i 4.
     Nëse x=0.9, hapi i 5.
     Nëse x>0.9, hapi i 6.
4.  y=x². Hapi i 7.
5.  y=2x. Hapi i 7.
6.  y=x-3. Hapi i 7.
7.  Shtype vlerën e variablës y
8.  Fundi.
```

Fig.1.3 Algoritmi numerik

Algoritmi i dhënë është shkruar duke pasur parasysh zgjidhjen e këtij problemi me kompjuter, gjë që vlen edhe për të gjithë algoritmet të cilat janë dhënë në pjesën vijuese të librit. Gjatë ekzekutimit të këtij algoritmi, nëse për variablën `x` merret vlera `4.7`, do të kalohet nëpër hapat: **1**, **2**, **3**, **6**, **7** dhe **8**.

Tek algoritmet e përbëra paraqitja e detajizuar e hapave të veçantë të algoritmit e komplikon shumë dhe e bën të paqartë strukturën logjike të algoritmit. Prandaj, si zgjidhje imponohet nevoja e paraqitjes së algoritmit në disa hapa të përgjithësuar, përkatësisht të përpilimit të *algoritmit të përgjithësuar*. Pastaj, për çdo hap të veçantë, mund të përpilohet edhe *algoritmi i detajizuar*.

Paraqitja grafike

Gjatë paraqitjes analitike të algoritmeve, nëse kemi të bëjmë edhe me algoritme relativisht të komplikuara, vështirë se mund të ndiqet rrjedhja e procesit llogaritës. Në praktikë, algoritmet paraqiten përmes *skemave grafike*, për vizatimin e të cilave përdoren disa *figura gjeometrike*, përkatësisht *blloqe të formave të ndryshme*.

Forma gjeometrike e blloqeve që shfrytëzohen gjatë përpilimit të skemave grafike e tregon edhe natyrën e operacioneve që kryhen brenda tyre. Disa nga

blloqet elementare që përdoren gjatë vizatimit të skemave grafike janë dhënë në *Fig.1.4*.

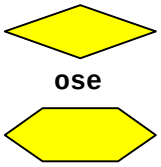
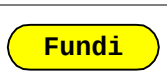
Blloku	Përdorimi
	Tregon fillimin e algoritmit
	Lexohen vlerat e variablave të shënuara në bllok
	Shtypen vlerat e variablave të shënuara në bllok
	Kryhen veprimet ose llogaritjet, duke shfrytëzuar shprehjet e shënuara në bllok
	Përcaktohet degëzimi i veprimeve të mëtejme, duke pasur parasysh kushtet e shënuara në bllok
	Tregon fundin e algoritmit

Fig.1.4 Blloqet elementare

Me *leximin* e vlerave të variablave të shënuara në bllok nënkuptohet marrja e vlerave përkatëse përmes njësisë hyrëse dhe vendosja e tyre në kujtesën e kompjuterit. Kurse përmes *shtypjes* vlerat e variablave merren nga kujtesa e kompjuterit dhe shtypen në njësinë dalëse të tij.

Skemat grafike të vizatuara duke shfrytëzuar blloqe të formave të ndryshme, shpesh quhen edhe *bllok-diagrame*, siç do të quhen edhe në pjesën vijuese të librit.

Shembull Algoritmi për llogaritjen e sipërfaqes s dhe perimetrit p të katërkëndëshit kënddrejtë, me brinjët a dhe b.

a. Forma analitike

```
1.  Fillimi.
2.  Merri vlerat e brinjëve: a, b
3.  s=a·b
4.  p=2·(a+b)
5.  Shtypi vlerat e llogaritura: s, p
6.  Fundi.
```

Fig.1.5

b. Forma grafike

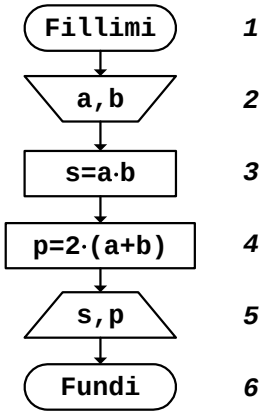


Fig.1.6

Shembull Bllok-diagrami i algoritmit për kalimin e rrugëkryqit, i cili u dha në formë analitike në *Fig.1.2*.

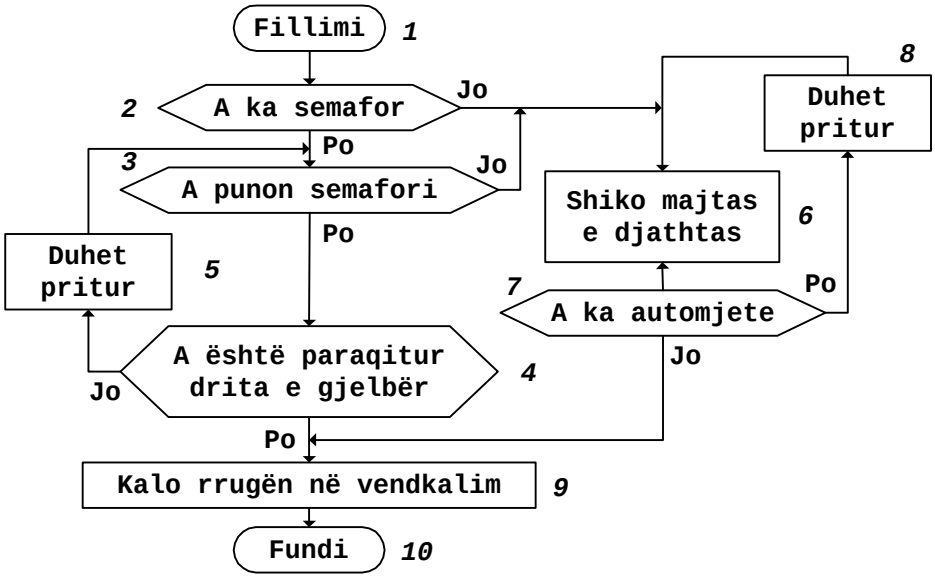


Fig.1.7 Bllok-diagrami i algoritmit për kalimin e rrugëkryqit

Nga shembujt e dhënë shihet se paraqitja e algoritmeve përmes bllok-diagrameve jep një dukshmëri shumë më të madhe të veprimeve që kryhen brenda blloqeve, si dhe në krejt algoritmin.

Testimi i algoritmeve

Me qëllim të kontrollimit të saktësisë së funksionimit të algoritmeve që përpilohen, duhet të bëhet testimi i tyre, duke marrë vetëm vlerat me të cilat përfshihen të gjitha rastet e mundshme të shfrytëzimit të tyre. Në këtë mënyrë, me punë minimale vërtetohet sjellja reale e algoritmeve para përdorimit praktik të tyre.

Shembull Testimi i algoritmit numerik të dhënë në *Fig.1.3*, duke e përpiluar fillimisht bllok-diagramin përkatës.

a. Bllok-diagrami

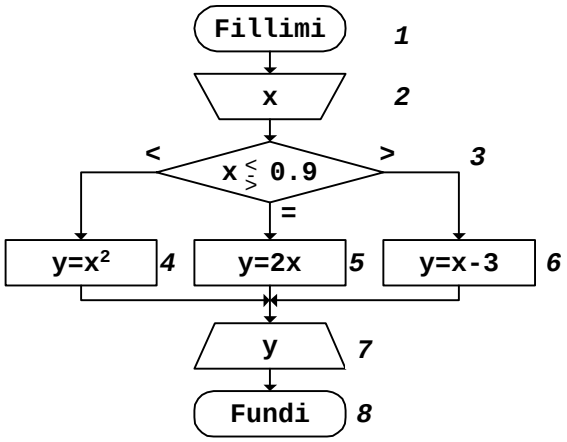


Fig.1.8

Me qëllim që testimi i bllok-diagramit të rrjedhë me një procedurë standarde, mund të shfrytëzohet një tabelë për testim, p.sh., si ajo që është dhënë në *Fig.1.9*.

b. Testimi - për `x=4.7`

Hapi	Blloku	Urdhri	Vlerat numerike merren	Rezultati
1	*1*	Fillimi	-	Fillimi i algoritmit
2	*2*	Lexo: x	Prej njësisë hyrëse	x=4.7
3	*3*	Pyet:x≤0.9	x → 2	>
4	*6*	y=x-3	x → 2	y=4.7-3=1.7
5	*7*	Shtyp: y	y → 3	Shtypet numri 1.7
6	*8*	Fundi	-	Fundi i algoritmit

Fig.1.9

Në tabelë, me shkurtesat `x→2` dhe `y→3` duhet nënkuptuar se vlera numerike e variablës `x` merret nga hapi i **2**, kurse ajo e variablës `y` - nga hapi i **3**.

Gjithnjë, mes *Fillimit* dhe *Fundit* të algoritmit, gjegjësisht bllok-diagramit, ekziston një *rrugë e mbyllur*, e cila, varësisht nga vlerat hyrëse, kalon nëpër pjesë të ndryshme të bllok-diagramit, ose një numër të ndryshëm herësh në pjesë të caktuara të tij.

c. Rruga - për x=4.7

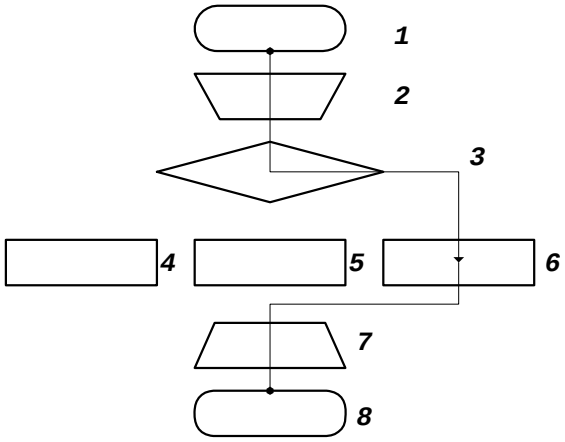


Fig.1.10

Testimi i algoritmit do të jetë komplet vetëm nëse, duke kaluar nëpër të gjitha rrugët e mundshme, vërtetohet funksionimi i saktë i tij.

Algoritmet nëpër blloqet e bllok-diagramit përkatës të të cilëve mund të kalohet vetëm njëherë, p.sh. si ai që është dhënë në *Fig.1.6*, quhen *algoritme lineare*. Kurse, algoritmet te bllok-diagramet e të cilëve paraqiten më shumë degë, p.sh. ashtu siç është ai i dhënë në *Fig.1.8*, quhen *algoritme të degëzuara*.

Përcjellja në kompjuter

Përkthimi i algoritmeve në një formë të kuptueshme për kompjuterin bëhet *duke i shkruar programet përkatëse në një gjuhë programuese*. Meqë aktualisht njëra ndër gjuhët programuese më të popullarizuara është gjuha C, përkatësisht versioni i saj C++, në pjesën vijuese të librit, përveç paraqitjeve grafike të algoritmeve, do të jepen edhe programet përkatëse në këtë gjuhë.

Shembull Programi i cili është shkruar në bazë të bllok-diagramit të dhënë në *Fig.1.8*.

```
// Programi Prg1_8
#include <iostream>
using namespace std;
int main()
{
      double x,y;
      cout << "Vlera e variablës x=";
      cin >> x;
      if (x < 0.9)
         y=x*x;
      else
         if (x == 0.9)
            y=2*x;
         else
            y=x-3;
      cout << "Rezultati y="
           << y
           << "\n";
      return 0;
}
```

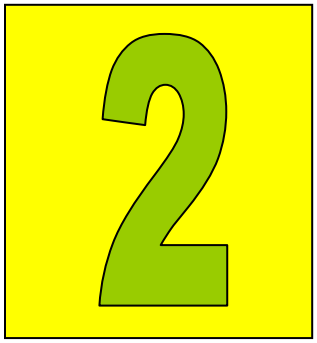
Gjatë ekzekutimit të programit, nëse pas mesazhit:

```
Vlera e variablës x=
```

përmes tastierës kompjuterit i jepet vlera `4.7`, në ekran do të shtypet vlera e variablës y, kështu:

```
Rezultati y=1.7
```

gjë që i përgjigjet vlerës së fituar në tabelën e *Fig.1.9*.



Llogaritja e shumës

Gjatë zgjidhjeve të problemeve të ndryshme, shpesh herë nevojitet të mblidhen anëtarët e një serie numrash. Në matematikë kjo mbledhje realizohet duke shtuar një nga një anëtarët e serisë, gjë që mund të shfrytëzohet edhe gjatë zgjidhjes së këtij problemi me kompjuter. Për shtimin automatik të anëtarëve të serisë, në algoritëm duhet të shfrytëzohet një unazë e mbyllur, para së cilës vlera fillestare e shumës merret zero.

Shumat e zakonshme

Si raste më të thjeshta të llogaritjes së shumave merren mbledhjet e numrave natyrorë, katrorëve ose kubeve të këtyre numrave, numrave natyrorë tek ose çift etj.

Shembull Shuma e numrave natyrorë mes `3` dhe `n`, nëse është dhënë vlera e variablës `n`.

$$s = 3 + 4 + \ldots + n = \sum_{i=3}^{n} i$$

a. Bllok-diagrami

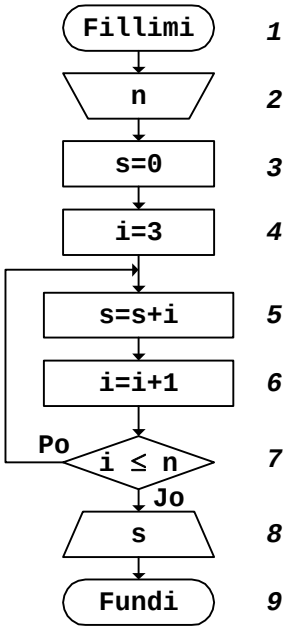


Fig.2.1

b. Rruga - për `n=5`

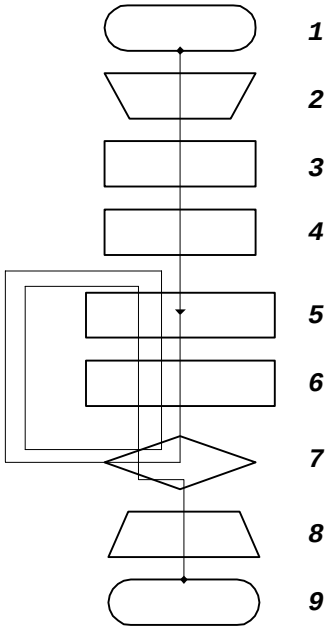


Fig.2.2

Në bllok-diagram është shfrytëzuar shprehja `i=i+1`, gjë që është e palogjikshme në matematikë. Por, këtu me shprehjen e barazimit kompjuterit i urdhërohet:

```
Llogarite vlerën numerike të shprehjes në anën e
djathtë dhe jepja variablës në anën e majtë të
barazimit!
```

Nga kjo shihet se me shprehjen `i=i+1` së pari variabla `i` rritet për `1` dhe pastaj rezultati i fituar ruhet përsëri te kjo variabël, përkatësisht me shprehjen e dhënë kompjuteri e nënkupton rritjen për `1` të vlerës së variablës `i`.

Algoritmet, si ai që është dhënë përmes bllok-diagramit në *Fig.2.1*, te të cilët pjesë të caktuara të tyre përsëriten brenda një unaze të mbyllur, quhen *algoritme ciklike*. Përfundimi i cikleve të përsëritjes së unazës, përcaktohet me kushtin për dalje nga unaza, i cili në shembullin konkret, lidhet me raportin e vlerave të variablave `i` dhe `n`. Kështu, në momentin kur plotësohet kushti `i>n`, përsëritja e ekzekutimit të unazës ndërpritet, përkatësisht dilet nga unaza.

c. Testimi - për `n=5`

Hapi	Blloku	Urdhëri	Vlerat numerike merren	Rezultati
1	***1***	Fillimi	**-**	Fillimi i algoritmit
2	***2***	Lexo: n	prej njësisë hyrëse	n=5
3	***3***	s=0	**-**	s=0
4	***4***	i=3	**-**	i=3
5	***5***	s=s+i	s → **3**, i → **4**	s=0+3=3
6	***6***	i=i+1	i → **4**	i=3+1=4
7	***7***	Pyet:i≤n	i → **6**, n → **2**	Po
8	***5***	s=s+i	s → **5**, i → **6**	s=3+4=7
9	***6***	i=i+1	i → **6**	i=4+1=5
10	***7***	Pyet:i≤n	i → **9**, n → **2**	Po
11	***5***	s=s+i	s → **8**, i → **9**	s=7+5=12
12	***6***	i=i+1	i → **9**	i=5+1=6
13	***7***	Pyet:i≤n	i → **12**, n → **2**	Jo
14	***8***	Shtyp:s	s → **11**	Shtypet numri 12
15	***9***	Fundi	**-**	Fundi i algoritmit

Fig.2.3

Vlera numerike e një variable, e cila nevojitet në një hap të caktuar, merret në hapin ku ajo variabël takohet së pari, nëse kthehemi pas në rrugën e kaluar. Kështu, p.sh., në tabelën e *Fig.2.3*, vlerat e variablave `s` dhe `i` në hapin e **11** janë marrë nga hapat **8** e **9**, sepse, nëse prej hapit të **11** kthehemi pas, në kolonën e fundit të tabelës, variabla `i` së pari takohet në hapin e **9**, kurse variabla `s` - në hapin e **8**. Shigjeta → e cila përdoret për të treguar se në cilin hap merren vlerat e nevojshme numerike, duhet të lexohet *prej hapit*.

Gjatë testimit në tabelën e mësipërme, shtimi i anëtarëve të serisë së numrave natyrorë, në hapat e veçantë rrjedh ashtu siç është treguar në *Fig.2.4*.

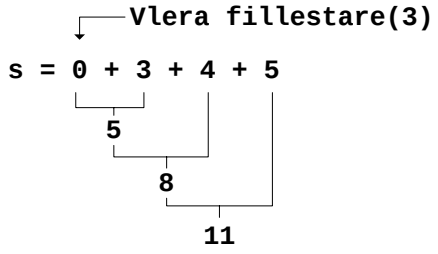


Fig.2.4

d. Programi

```
// Programi Prg2_1
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double s;
  cout << "Vlera e variablës n=";
  cin >> n;
  s=0;
  i=3;
  do
  {
    s=s+i;
    i=i+1;
  }
  while (i<=n);
  cout << "Shuma e numrave natyrorë s="
       << s
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për `n=5`, rezultati që shtypet në ekran është:

```
Shuma e numrave natyrorë s=12
```

gjë që përputhet me rezultatin i cili është fituar gjatë testimit përmes tabelës në *Fig.2.3*.

Anëtarët e vargut mund të jenë numra me një ligjshmëri të caktuar, përkatësisht nuk do të thotë se mes vete duhet të dallohen për vlerën `1`. Për gjetjen e shumës së anëtarëve të vargjeve të tilla shfrytëzohen algoritme të

ngjashme me ato që u dhanë më sipër, duke pasur kujdes vetëm në ligjshmërinë e gjenerimit të anëtarëve të vargut.

Shembull Shuma e kubeve të numrave natyrorë çiftë mes 2 dhe n, nëse është dhënë vlera e variablës n.

$$s = 2^3 + 4^3 + \ldots = \sum_{\substack{i=2 \\ (\text{çift})}}^{n} i^3$$

a. Bllok-diagrami

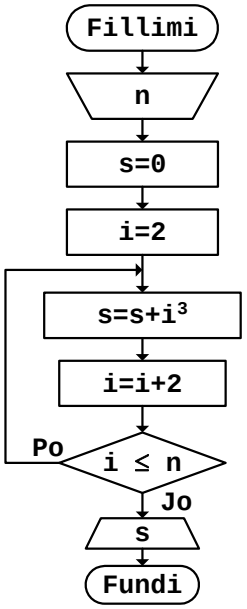


Fig.2.5

b. Rruga - për n=9

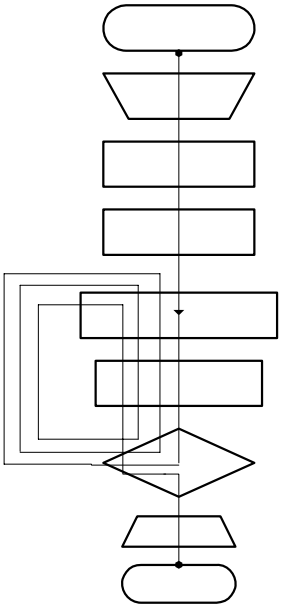


Fig.2.6

Rezultati që fitohet gjatë rrugës së kaluar është:

vlera fillestare

$$s = 0 + 2^3 + 4^3 + 6^3 + 8^3 = 800$$

c. Programi

```
// Programi Prg2_5
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double s;
  cout << "Vlera e variablës n=";
  cin >> n;
  s=0;
  i=2;
  do
  {
    s=s+i*i*i;
    i=i+2;
  }
  while (i<=n);
  cout << "Shuma s="
       << s
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë për `n=9`, si rezultat në ekran do të shtypet:

```
Shuma s=800
```

ashtu siç u tregua më sipër.

Detyra

Të llogaritet shuma:

a. e numrave natyrorë çiftë mes `2` dhe `n`;
b. e numrave natyrorë tekë mes `1` dhe `n`;
c. e katrorëve të numrave natyrorë mes `4` dhe `n`;
d. e rrënjëve katrore të numrave natyrorë tekë mes `3` dhe `n`,

nëse dihet vlera e variablës `n`.

Shprehjet e shumave përkatëse do të duken:

a.

$$s = 2 + 4 + 6 + \ldots = \sum_{\substack{i=2 \\ (\text{çift})}}^{n} i$$

b.

$$s = 1 + 3 + 5 + \ldots = \sum_{\substack{i=1 \\ (\text{tek})}}^{n} i$$

c.

$$s = 4^2 + 5^2 + 6^2 + \ldots + n^2 = \sum_{i=4}^{n} i^2$$

d.

$$s = \sqrt{3} + \sqrt{5} + \sqrt{7} + \ldots = \sum_{\substack{i=3 \\ (\text{tek})}}^{n} \sqrt{i}$$

Anëtarët e serive mund të formohen edhe duke shfrytëzuar ligjshmëri më komplekse të raporteve të numrave natyrorë.

Shembull Shuma e prodhimit të numrave natyrorë tekë e çiftë - të njëpasnjëshëm, mes vlerave 1 dhe n, nëse dihet vlera e variablës n.

$$s = 1 \cdot 2 + 3 \cdot 4 + 5 \cdot 6 + \ldots = \sum_{\substack{i=1 \\ (\text{tek})}}^{n} i \cdot (i + 1)$$

a. Bllok-diagrami

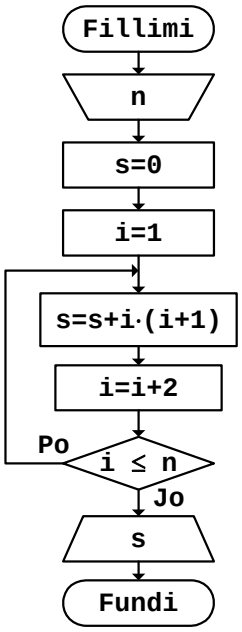


Fig.2.7

b. Rruga - për `n=8`

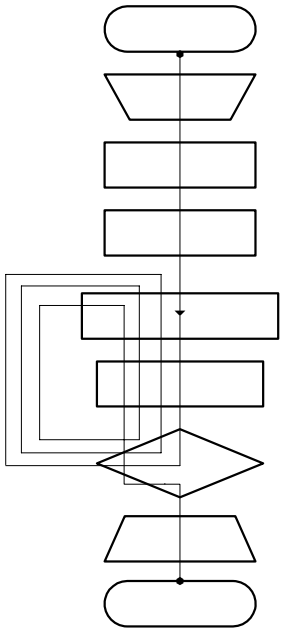


Fig.2.8

Shuma që fitohet gjatë rrugës së kaluar, për `n=8` është:

$$s = 1 \cdot 2 + 3 \cdot 4 + 5 \cdot 6 + 7 \cdot 8 = 100$$

Detyra

Të llogaritet shuma:

a. e pjesëtimit të numrave natyrorë tekë me katrorët e numrave natyrorë çiftë - të njëpasnjëshëm, mes vlerave `1` dhe `n`;
b. e katrorëve të numrave natyrorë tekë dhe e kubeve të numrave natyrorë çiftë - të njëpasnjëshëm, mes vlerave `1` dhe `n`,

nëse dihet vlera e variablës `n`.

Shprehjet e shumave të kërkuara duken:

a.

$$s = \frac{1}{2^2} + \frac{3}{4^2} + \frac{5}{6^2} + \ldots$$

b.

$$s = 1^2 + 2^3 + 3^2 + 4^3 + \ldots$$

Shumat e çfarëdoshme

Anëtarët e serive numerike mund të formohen si shprehje të çfarëdoshme matematikore. Procedurat e gjetjes së shumave të tyre nuk do të ndryshojnë aspak nga ato që u dhanë më sipër.

Shembull

Llogaritja e vlerës së shumës:

$$s = \sum_{\substack{i=2 \\ (i \neq 4)}}^{n+1} \left\{ 2i + \frac{x}{3} \right\}^2$$

nëse dihen vlerat e variablave `n` dhe `x`.

a. Bllok-diagrami

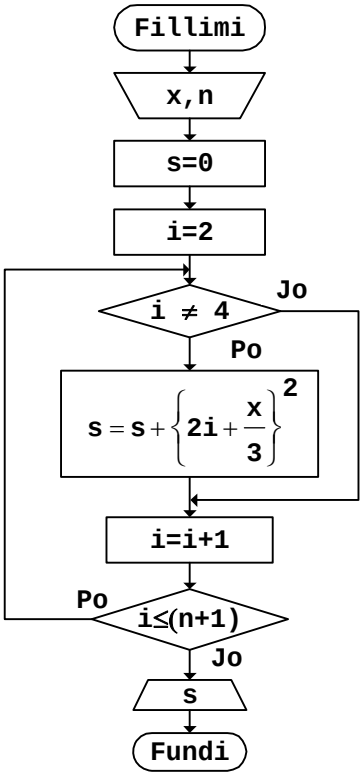


Fig.2.9

b. Rruga - për `x=1` dhe `n=5`

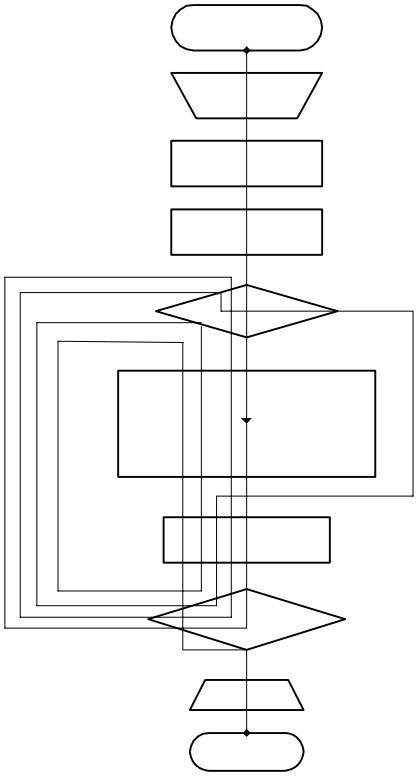


Fig.2.10

Vlera e shumës, që fitohet gjatë rrugës së kaluar në *Fig.2.10*, është:

$$s = \left\{2\cdot 2+\frac{1}{3}\right\}^2 + \left\{2\cdot 3+\frac{1}{3}\right\}^2 + \left\{2\cdot 5+\frac{1}{3}\right\}^2 + \left\{2\cdot 6+\frac{1}{3}\right\}^2 = 317.778$$

c. Programi

```
// Programi Prg2_9
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int n,i;
  double x,s;
```

```
  cout << "Vlerat hyrëse x dhe n: ";
  cin >> x
      >> n;
  s=0;
  i=2;
  do
  {
    if (i!=4)
       s=s+pow(2*i+x/3,2);
    i=i+1;
  }
  while (i<=n+1);
  cout << "Shuma s="
       << s
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat hyrëse `x=1` dhe `n=5`, si rezultat do të shtypet:

```
Shuma s=317.778
```

ashtu siç u fitua edhe gjatë llogaritjes me dorë.

Nga krejt kjo që u dha më sipër, si përfundim mund të nxirret se në rastin e përgjithshëm, gjatë llogaritjes së shumës, shprehja për llogaritjen e shumës brenda unazës së mbyllur shkruhet:

s=s+(shprehja nën simbolin e shumës)

Detyra

Të llogariten shumat:

a.

$$y = \sum_{k=1}^{n+1} \{x + 2k - 1\}^{3/2}$$

b.

$$z = \sum_{\substack{i=2 \\ (\text{çift})}}^{2n} \left\{ i + \frac{i}{3} \right\}^2$$

c.

$$g = \sum_{\substack{j=1 \\ (j\neq 3,4,5)}}^{n+2} \left\{\frac{1}{2j+3}\right\}^{j/3}$$

nëse dihen vlerat e variablave n dhe x.

Gjatë vizatimit të bllok-diagrameve përkatës, duhet pasur parasyshë shprehjet e zbërthyera të shumave të dhëna, të cilat duken:

a.

$$y = \{x + 2 \cdot 1 - 1\}^{3/2} + \{x + 2 \cdot 2 - 1\}^{3/2} + \ldots + \{x + 2 \cdot (n+1) - 1\}^{3/2}$$

b.

$$z = \left\{2 + \frac{2}{3}\right\}^2 + \left\{4 + \frac{4}{3}\right\}^2 + \ldots + \left\{2n + \frac{2n}{3}\right\}^2$$

c.

$$g = \left\{\frac{1}{2 \cdot 1 + 3}\right\}^{1/3} + \left\{\frac{1}{2 \cdot 2 + 3}\right\}^{2/3} + \left\{\frac{1}{2 \cdot 6 + 3}\right\}^{6/3} + \ldots$$

Shumat mund të paraqiten edhe brenda shprehjeve të funksioneve të ndyshme. Gjatë kësaj, së pari llogariten vlerat e shumave dhe pastaj, duke shfrytëzuar këto vlera, llogariten edhe vlerat e funksioneve.

Shembull Llogaritja e vlerës së funksionit:

$$y = \frac{x}{3} + 2\sum_{i=1}^{m+n-1}\left\{x + \frac{2}{i}\right\}^{i/3}$$

nëse dihen vlerat e variablave m, n dhe x.

a. Bllok-diagrami

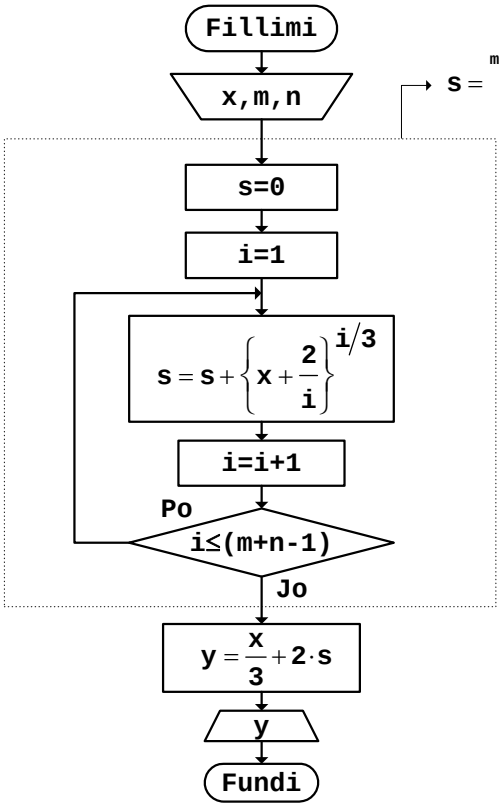


Fig.2.11

b. Rruga - për `x=1`, `m=3`, `n=1`

$$s = \sum_{i=1}^{m+n+1} \left\{ x + \frac{2}{i} \right\}^{i/3}$$

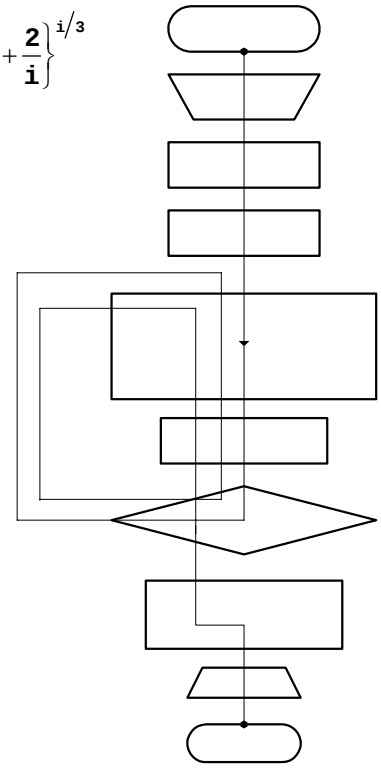


Fig.2.12

Për vlerat hyrëse që janë marrë si shembull gjatë vizatimit të rrugës në *Fig.2.12*, vlera e llogaritur e funksionit është:

$$y = \frac{1}{3} + 2 \cdot \left[\left\{ 1 + \frac{2}{1} \right\}^{1/3} + \left\{ 1 + \frac{2}{2} \right\}^{2/3} + \left\{ 1 + \frac{2}{3} \right\}^{3/3} \right]$$

c. Programi

```
// Programi Prg2_11
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
```

```
  int m,n,i;
  double x,s,y;
  cout << "Vlerat hyrëse x,m dhe n: ";
  cin >> x
      >> m
      >> n;
  s=0;
  i=1;
  do
  {
    s=s+pow(x+2./i,i/3.);
    i=i+1;
  }
  while (i<=m+n-1);
  y=x/3+2*s;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Nëse pas ekzekutimit të programit, si vlera hyrëse përmes tastierës kompjuterit i jepen vlerat `x=1`, `m=3` dhe `n=1`, rezultati që shtypet në ekran është:

```
Vlera e funksionit y=9.72597
```

Shuma mund të paraqitet edhe te funksionet të cilat përcaktohen përmes më shumë shprehjeve.

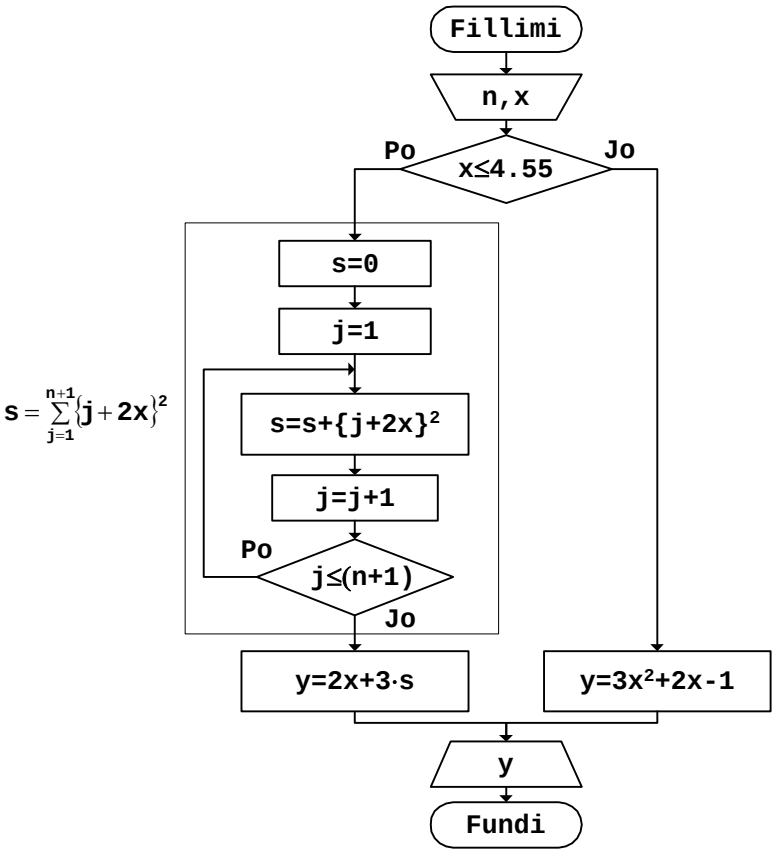
Shembull

Llogaritja e vlerës së funksionit:

$$y = \begin{cases} 2x + 3\sum_{j=1}^{n+1}\{j + 2x\}^2 & \text{për } x \le 4.55 \\ 3x^2 + 2x - 1 & \text{për } x > 4.55 \end{cases}$$

nëse dihen vlerat e variablave `n` dhe `x`.

a. Bllok-diagrami



$$s = \sum_{j=1}^{n+1} \{j + 2x\}^2$$

Fig.2.13

Nga bllok-diagrami i dhënë në *Fig.2.13* shihet se këtu përveç *strukturës algoritmike të degëzuar*, në njërën degë të algoritmit paraqitet edhe *strukturë algoritmike ciklike*. Për të qenë algoritmi më i saktë, vlera e variablës n do të duhej të lexohet në degën e majtë të bllokut për degëzim.

b. Rruga - për `n=3` dhe `x=3.5`

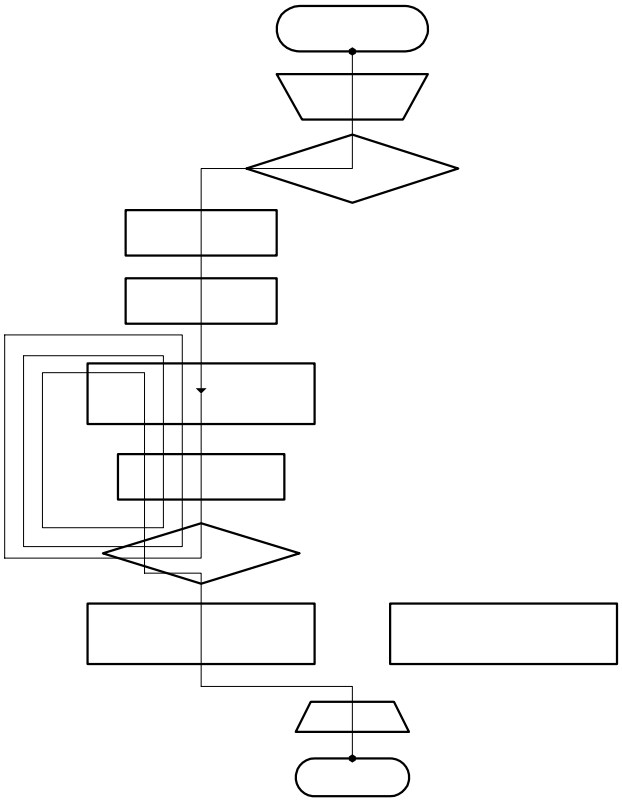


Fig.2.14

c. Programi

```
// Programi Prg2_13
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int n,j;
  double x,y,s;
  cout << "Vlerat hyrëse n dhe x: ";
  cin >> n
      >> x;
  if (x<=4.55)
    {
     s=0;
```

```
      j=1;
      do
      {
            s=s+pow(j+2*x,2);
            j=j+1;
      }
      while (j<=n+1);
      y=2*x+3*s;
    }
  else
    y=3*pow(x,2)+2*x-1;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Nëse programi i dhënë ekzekutohet për vlerat hyrëse `n=3` dhe `x=3.5`, si rezultat në ekran do të shtypet:

```
Vlera e funksionit y=1105
```

Detyra Të llogariten vlerat e funksioneve:

a.

$$y = ax^2 - bx + 4\sum_{i=3}^{n+1}\left\{i^2 + 2a\right\}^{x/b}$$

b.

$$z = \frac{x}{a} + \frac{3x}{\sum\limits_{\substack{i=2\\(\text{çift})}}^{n+3}\{m + 2i\}^a}$$

c.

$$g = \begin{cases} 2x^2 + 3x - 4 & \text{për } x+2 > a+1 \\ x^2 - 2\sum\limits_{k=2}^{n+m}\left\{3k + k^2\right\} & \text{për } x+2 \le a+1 \end{cases}$$

nëse dihen vlerat e variablave `a`, `b`, `m`, `n` dhe `x`.

Brenda shprehjes së një funksioni, njëkohësisht mund të paraqiten disa shuma, të cilat duhet të llogariten para se kompjuterit t'i urdhërohet llogaritja e vlerës së funksionit.

Shembull Llogaritja e vlerës së funksionit:

$$y = \frac{x}{2} + 3\sum_{i=1}^{m}(2i + b) - 4\sum_{i=2}^{m+2}\left(\frac{i}{a} - b\right)$$

nëse dihen m, a dhe b.

a. Bllok-diagrami

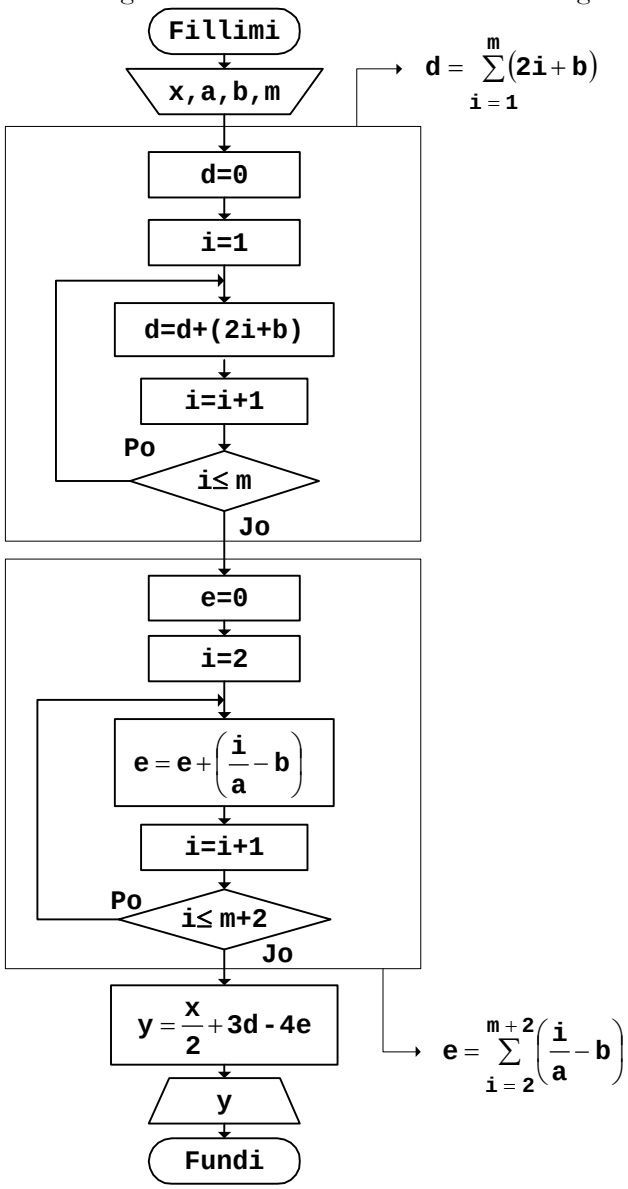


Fig.2.15

b. Rruga - për m=2, a=2, b=3, x=4

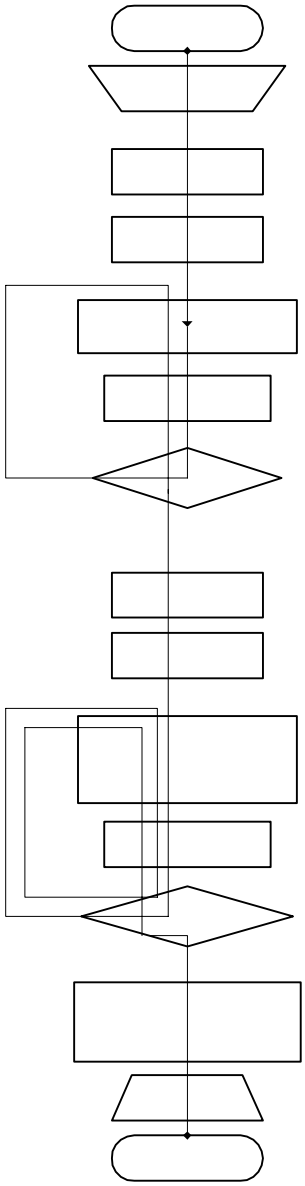


Fig.2.16

Vlera e funksionit, e cila llogaritet gjatë rrugës së kaluar më sipër, është:

$$y = \frac{4}{2} + 3\{(2 \cdot 1 + 3) + (2 \cdot 2 + 3)\} - 4\left\{\left(\frac{2}{2} - 3\right) + \left(\frac{3}{2} - 3\right) + \left(\frac{4}{2} - 3\right)\right\}$$

c. Programi

```
// Programi Prg2_15
#include <iostream>
using namespace std;
int main()
{
  int m,i;
  double a,b,x,y,d,e;
  cout << "Vlerat hyrëse x,a,b dhe m: ";
  cin >> x
      >> a
      >> b
      >> m;
  d=0;
  i=1;
  do
  {
    d=d+(2*i+b);
    i=i+1;
  }
  while (i<=m);
        e=0;
        i=2;
        do
          {
                e=e+(i/a-b);
                i=i+1;
          }
        while (i<=m+2);
  y=x/3+3*d-4*e;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat e variablave të cilat janë shfrytëzuar gjatë testimit të bllok-diagramit, rezultati që shtypet në ekran është:

```
Vlera e funksionit y=55.3333
```

Funksioni mund të ketë më shum vlera, nëse në shprehjen e shumës, e cila merr pjesë në funksion, figurojnë parametra që ndryshojnë.

Shembull Llogaritja e vlerës së funksionit:

$$g = 2m - 3\sum_{i=2}^{m+1}(ki + b)$$

$$\text{për } k = 1,2,\ldots,m$$

nëse janë dhënë vlerat e variablave `m` dhe `b`.

a. Bllok-diagrami

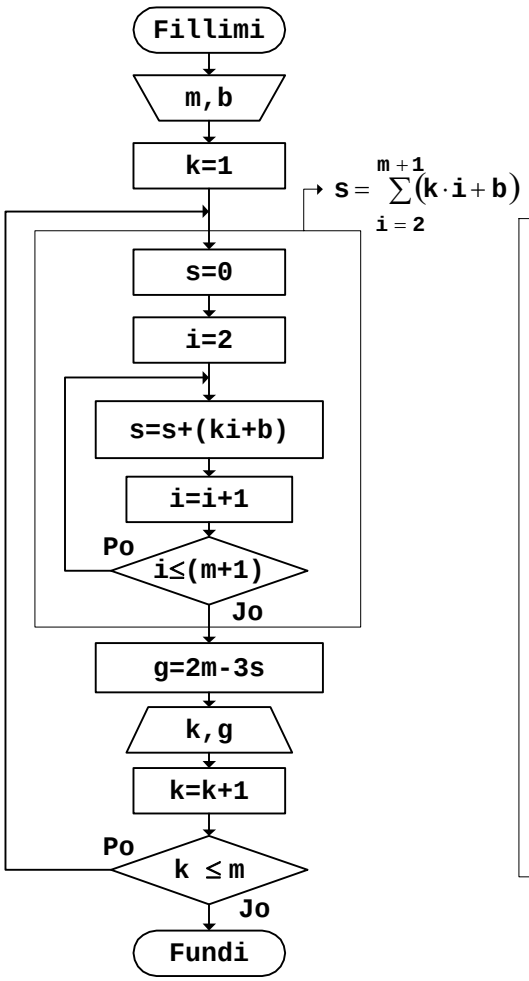


Fig.2.17

b. Rruga - për `m=2` dhe `b=3.5`

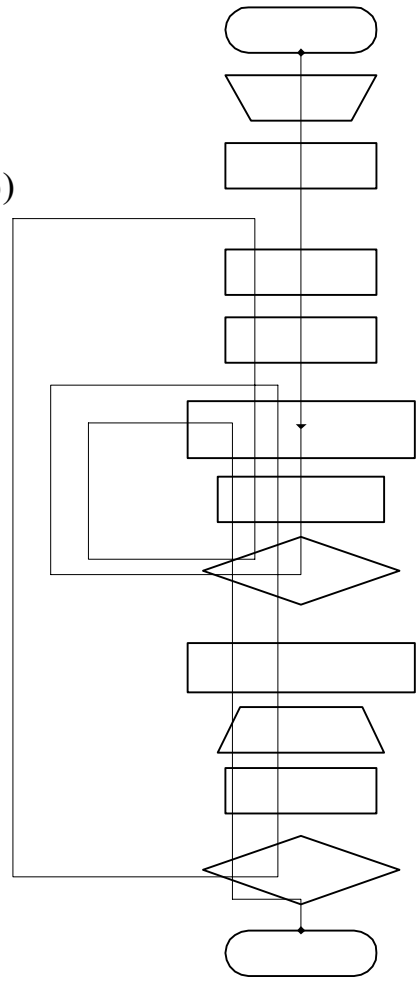


Fig.2.18

Vlerat që llogariten gjatë rrugës së kaluar janë:

$k = 1 \quad g = 2 \cdot 2 - 3 \cdot \{(1 \cdot 2 + 3.5) + (1 \cdot 3 + 3.5)\}$

$k = 2 \quad g = 2 \cdot 2 - 3 \cdot \{(2 \cdot 2 + 3.5) + (2 \cdot 3 + 3.5)\}$

c. Programi

```
// Programi Prg2_17
#include <iostream>
using namespace std;
int main()
{
  int m,k,i;
  double b,s,g;
  cout << "Vlerat hyrëse b dhe m: ";
  cin >> b
      >> m;
  for (k=1;k<=m;k++)
  {
    s=0;
    for (i=2;i<=m+1;i++)
        s=s+(k*i+b);
    g=2*m-3*s;
    cout << "k="
         << k
         << "   g="
         << g
         << "\n";
  }
  return 0;
}
```

Për vlerat hyrëse `m=2` dhe `b=3.5`, pas ekzekutimit të programit, si rezultat në ekran shtypen dy vlera, kështu:

```
k=1   g=-32
k=2   g=-47
```

sepse komanda `cout`, e cila gjendet brenda unazës së variablës `k`, ekzekutohet vetëm dy herë, gjë që shihet edhe gjatë testimit të bllok-diagramit në *Fig.2.18*.

3

Llogaritja e prodhimit

Prodhimi i anëtarëve të një serie numerike gjendet ngjajshëm si edhe shuma e tyre. Por, për dallim nga shuma, gjatë llogaritjes së prodhimit vlera fillestare merret 1, kurse brenda unazës së mbyllur në rast të përgjithshëm shkruhet:

p=p·(shprehja nën simbolin e prodhimit)

Prodhimet e zakonshme

Prodhimet elementare të cilat takohen në praktikë janë ato të llogaritjes së prodhimit të numrave natyrorë, katrorëve ose kubeve të tyre, prodhimit të numrave natyrorë tekë ose çiftë etj.

Shembull Prodhimi i numrave natyrorë mes 2 dhe n, nëse është dhënë vlera e variablës n.

$$p = 2 \cdot 3 \cdot 4 \cdot 5 \cdot \ldots \cdot n = \prod_{i=2}^{n} i$$

a. Bllok-diagrami

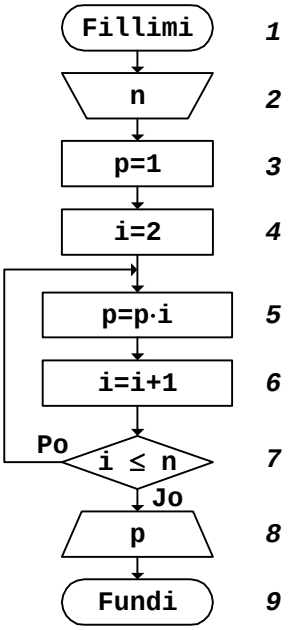


Fig.3.1

b. Rruga - për n=3

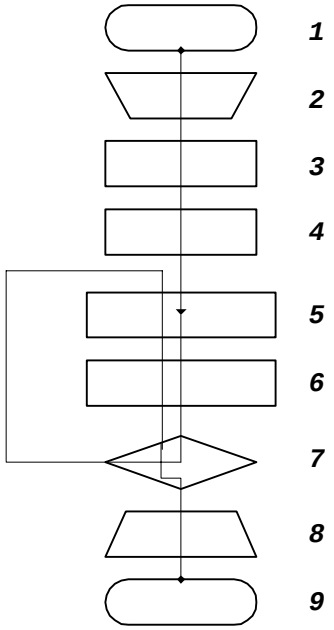


Fig.3.2

c. Testimi - për n=3

Hapi	Blloku	Urdhëri	Vlerat numerike merren	Rezultati
1	*1*	Fillimi	-	Fillimi i algoritmit
2	*2*	Lexo: n	prej njësisë hyrëse	n=3
3	*3*	p=1	-	p=1
4	*4*	i=2	-	i=2
5	*5*	p=p·i	p → **3**, i → **4**	p=1·2=2
6	*6*	i=i+1	i → **4**	i=2+1=3
7	*7*	Pyet:i≤n	i → **6**, n → **2**	Po
8	*5*	p=p·i	p → **5**, i → **6**	p=2·3=6
9	*6*	i=i+1	i → **6**	i=3+1=4
10	*7*	Pyet:i≤n	i → **9**, n → **2**	Jo
11	*8*	Shtyp:p	p → **8**	Shtypet numri 6
12	*9*	Fundi	-	Fundi i algoritmit

Fig.3.3

Llogaritja e vlerës së prodhimit, në hapat e veçantë të testimit, rrjedh kështu:

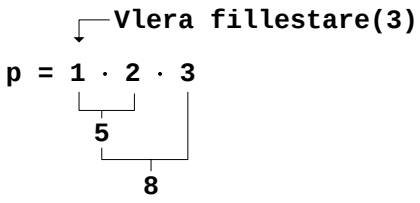


Fig.3.4

d. Programi

```
// Programi Prg3_1
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double p;
  cout << "Vlera e variablës n=";
  cin >> n;
  p=1;
  i=2;
  do
  {
    p=p*i;
    i=i+1;
  }
  while (i<=n);
  cout << "Prodhimi p="
       << p
       << "\n";
  return 0;
}
```

Nëse gjatë ekzekutimit të programit, përmes tastierës kompjuterit i jepet vlera n=3, rezultati që shtypet në ekran është:

```
Prodhimi p=6
```

gjë që fitohet edhe gjatë testimit të algoritmit, ashtu siç është treguar në *Fig.3.4*.

Detyra

Të llogaritet prodhimi:

a. i numrave natyrorë çiftë mes `4` dhe `n`;
b. i katrorëve të numrave natyrorë tekë mes `3` dhe `n`;
c. i shumës së numrave natyrorë tekë e çiftë - të njëpasnjëshëm mes `1` dhe `n`;
d. i katrorëve të numrave natyrorë tekë dhe i kubeve të numrave natyrorë çiftë - të njëpasnjëshëm mes `1` dhe `n`,

nëse dihet vlera e variablës `n`.

Shprehjet e prodhimeve të kërkuara duken kështu:

a.

$$p = 4 \cdot 6 \cdot 8 \cdot \ldots$$

b.

$$p = 3^2 \cdot 5^2 \cdot 7^2 \cdot \ldots$$

c.

$$p = (1+2) \cdot (3+4) \cdot (5+6) \cdot \ldots$$

d.

$$p = 1^2 \cdot 2^3 \cdot 3^2 \cdot 4^3 \cdot \ldots$$

Prodhimet e çfarëdoshme

Procedura që zbatohet gjatë llogaritjes së prodhimit të anëtarëve të serive të çfarëdoshme, është e ngjashme me ate që përdoret për llogaritjen e shumës së serive përkatëse.

Shembull

Llogaritja e vlerës numerike të shprehjes:

$$g = 3x + 4 \prod_{\substack{i=2 \\ (\text{çift})}}^{m+1} \left(i + \frac{2}{x} \right)$$

nëse janë dhënë vlerat e variablave `m` dhe `x`.

a. Bllok-diagrami

b. Rruga - për `m=8` dhe `x=1.5`

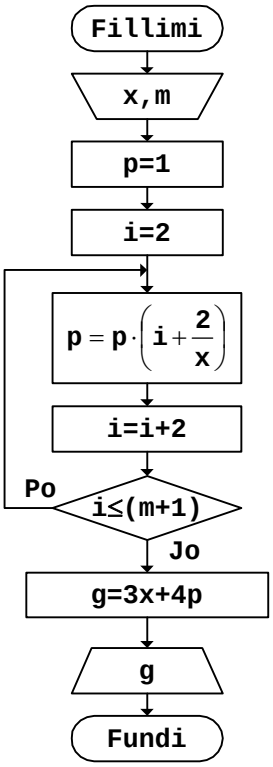


Fig.3.5

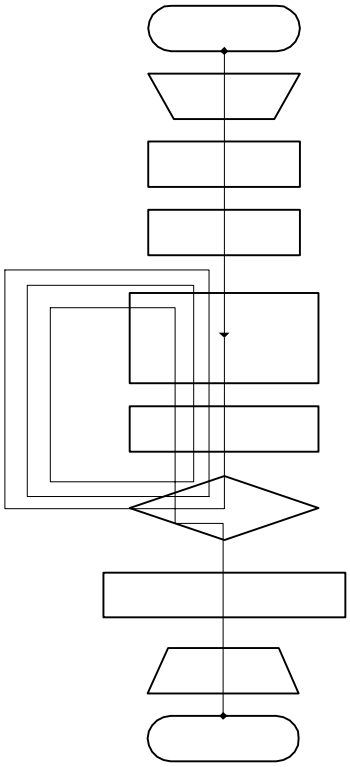


Fig.3.6

Vlera që llogaritet gjatë rrugës së kaluar është:

$$g = 3 \cdot 1.5 + 4 \cdot \left\{ \left(2 + \frac{2}{1.5}\right) \cdot \left(4 + \frac{2}{1.5}\right) \cdot \left(6 + \frac{2}{1.5}\right) \cdot \left(8 + \frac{2}{1.5}\right) \right\}$$

c. Programi

```
// Programi Prg3_5
#include <iostream>
using namespace std;
int main()
{
  int m,i;
  double x,p,g;
  cout << "Vlerat hyrëse x dhe m: ";
  cin >> x
      >> m;
```

```
  p=1;
  i=2;
  do
  {
    p=p*(i+2/x);
    i=i+2;
  }
  while (i<=m+1);
  g=3*x+4*p;
  cout << "Vlera e funksionit g="
       << g
       << "\n";
  return 0;
}
```

Për vlerat hyrëse `x=1.5` dhe `m=8`, rezultati që shtypet në ekran është:

```
Vlera e funksionit g=4871.66
```

Prodhimi, njëlloj si edhe shuma, mund të paraqitet brenda shprehjeve te funksioneve të cilat përcaktohen me më shumë shprehje.

Shembull Llogaritja e vlerës numerike të funksionit:

$$d = \begin{cases} 2x + 3n - 1 & \text{për } x > n + 2 \\ x - 2\prod_{k=1}^{n+2}\left(k^2 + \frac{x}{3}\right) & \text{për } x < n + 2 \\ e^{2x} + 3 & \text{për } x = n + 2 \end{cases}$$

nëse janë dhënë vlerat e variablave n dhe x.

a. Bllok-diagrami

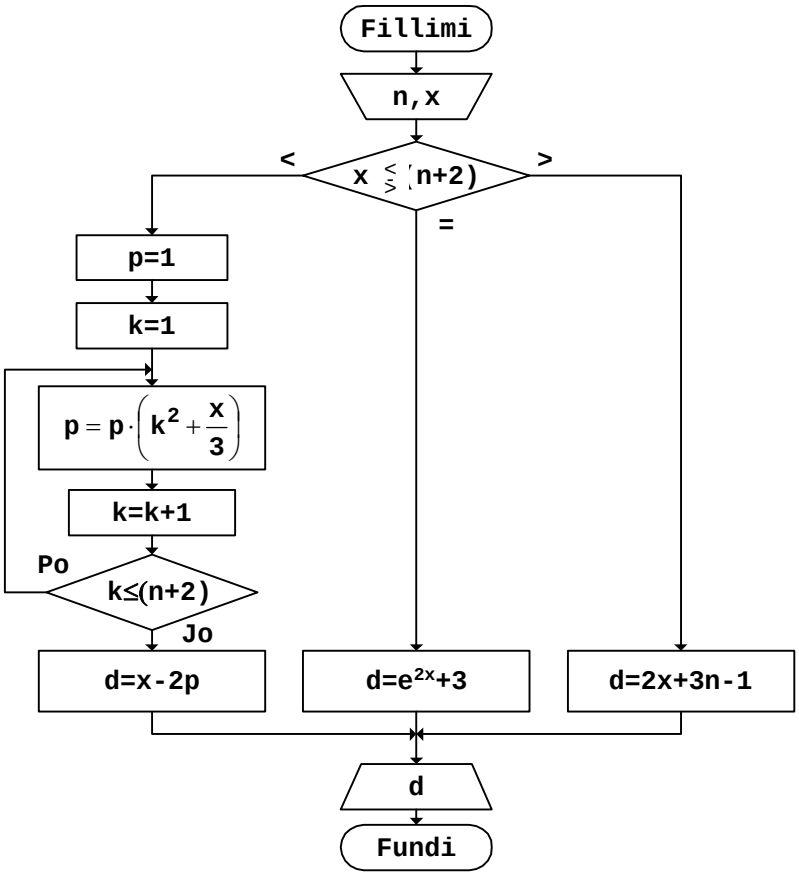


Fig.3.7

Versioni më i saktë i bllok-diagramit mund të vizatohet duke e vendosur bllokun për leximin e variablës n në degën e majtë të degëzimit që është vendosur në fillim të tij, sepse variabla në fjalë shfrytëzohet vetëm në këtë pjesë.

b. Rruga - për n=2 dhe x=3

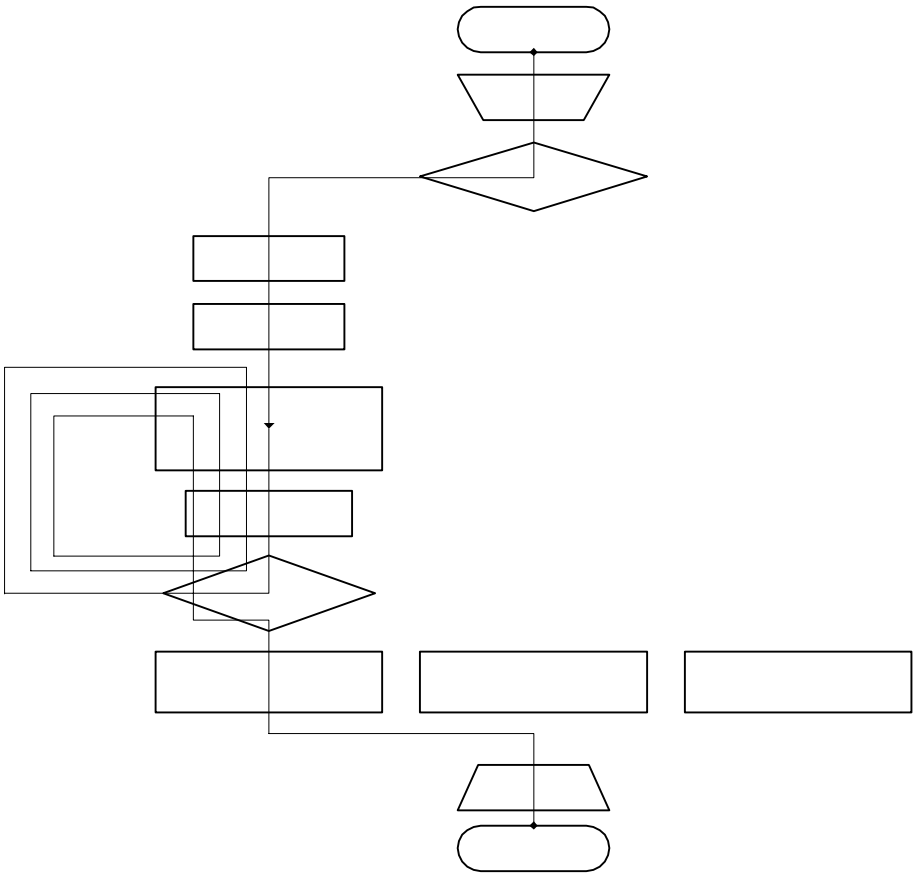


Fig.3.8

c. Programi

```
// Programi Prg3_7
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int n,k;
  double x,p,d;
  cout << "Vlerat hyrëse n dhe x: ";
  cin >> n
      >> x;
  if (x<(n+2))
    {
```

```
      p=1;
      for (k=1;k<=n+2;k++)
          p=p*(k*k+x/3);
      d=x-2*p;
     }
  else
     if (x==n+2)
        d=exp(2*x)+3;
     else
        d=2*x+3*n-1;
  cout << "Rezultati d="
       << d
       << "\n";
  return 0;
}
```

Nëse programi i dhënë ekzekutohet për vlerat hyrëse `n=2` dhe `x=3`, në ekran do të shtypet:

```
Rezultati d=-3397
```

Brenda shprehjes së një funksioni mund të paraqiten edhe më shumë prodhime të anëtarëve të serive, të cilat duhet të llogariten para se prej kompjuterit të kërkohet llogaritja e vlerës së vetë funksionit.

Shembull Llogaritja e vlerës së funksionit:

$$z = 3a + 2\prod_{i=1}^{m+1}(2i + a) - \prod_{\substack{i=2 \\ (\text{çift})}}^{m+n}(i + b)$$

nëse janë dhënë vlerat e variablave `m`, `n`, `a` dhe `b`.

a. Bllok-diagrami

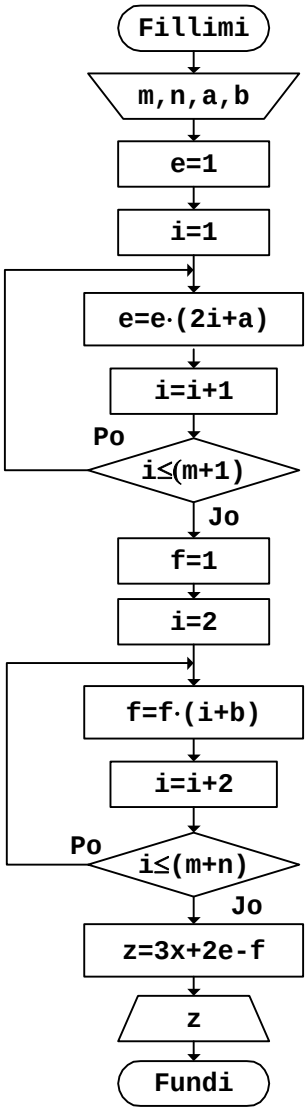


Fig.3.9

b. Rruga - për `m=2`, `n=7`, `a=3` dhe `b=1`

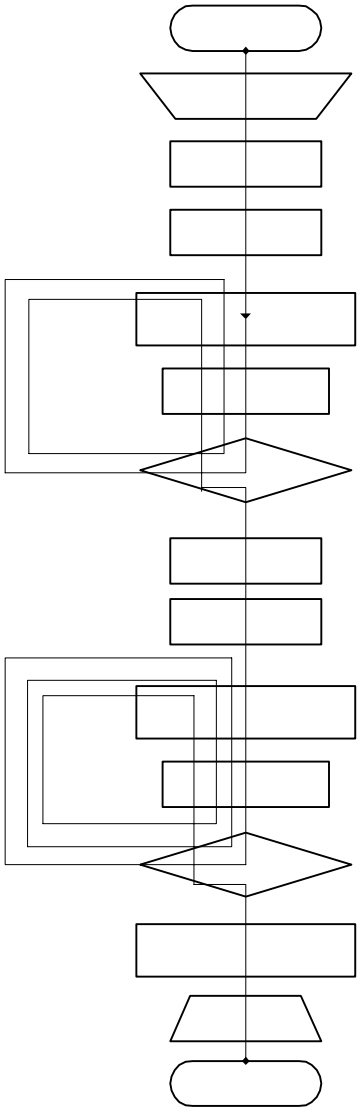


Fig. 3.10

Vlera e funksionit, e cila llogaritet gjatë rrugës së kaluar në *Fig.3.10*, është:

$$z = 3\cdot 3 + 2\cdot \underbrace{\{(2\cdot 1+3)\cdot(2\cdot 2+3)\cdot(2\cdot 3+3)\}}_{e} - \underbrace{\{(2+1)\cdot(4+1)\cdot(6+1)\cdot(8+1)\}}_{f}$$

c. Programi

```
// Programi Prg3_9
#include <iostream>
using namespace std;
int main()
{
  int m,n,i;
  double a,b,e,f,z;
  cout << "Vlerat hyrëse m,n,a dhe b: ";
  cin >> m
      >> n
      >> a
      >> b;
  e=1;
  for (i=1;i<=m+1;i++)
      e=e*(2*i+a);
  f=1;
  i=2;
  do
  {
    f=f*(i+b);
    i=i+2;
  }
  while (i<=m+n);
  z=3*a+2*e-f;
  cout << "Vlera e funksionit z="
       << z
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat e variablave hyrëse, të cilat janë shfrytëzuar gjatë vizatimit të rrugës së kaluar, si rezultat në ekran shtypet:

```
Vlera e funksionit z=-306
```

Detyra Të llogariten vlerat e funksioneve:

a.

$$y = \frac{x}{2} + 2\prod_{\substack{k=1\\(k\neq 3)}}^{m+n}\{k + 3x\}^{a/2}$$

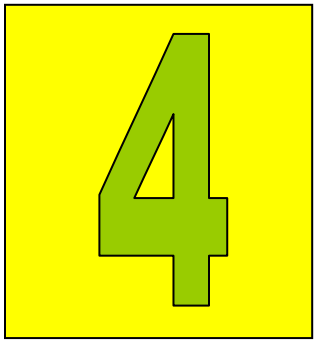
b.

$$g = 3\prod_{j=2}^{m}\{x + 2j\}^2 - \frac{1}{\prod_{k=2}^{n}(k + 1)}$$

c.

$$t = \begin{cases} \prod_{\substack{i=2\\(\text{çift})}}^{n}(i+3) & \text{për } x^2+1 \le 7 \\ \frac{x}{2} - 4\prod_{i=1}^{m+2}(2i+1) & \text{për } x^2+1 > 7 \end{cases}$$

nëse janë dhënë vlerat e variablave m, n dhe x.



Shuma dhe prodhimi

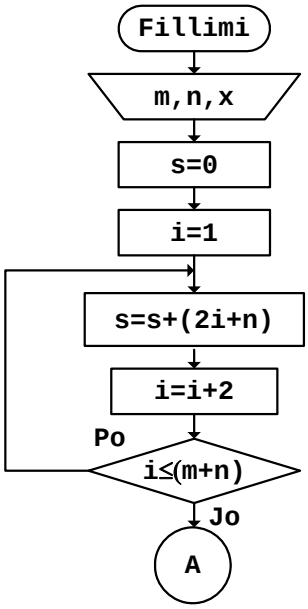
Brenda shprehjeve të funksioneve mund të paraqiten njëkohësisht shuma dhe prodhime të anëtarëve të serive. Në këto raste, llogaritja rrjedh plotësisht njëlloj si edhe kur në shprehje paraqiten disa shuma ose disa prodhime.

Shembull Llogaritja e vlerës numerike të shprehjes:

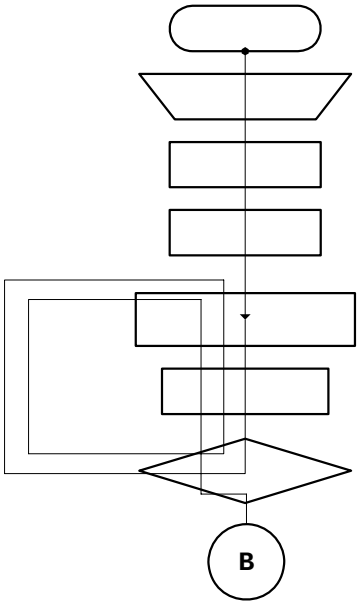
$$y = \frac{x}{2} + 3\sum_{\substack{i=1\\(tek)}}^{m+n}(2i + n) - 2\prod_{k=2}^{n}(k + x)$$

nëse janë dhënë vlerat e variablave `m`, `n` dhe `x`.

a. Bllok-diagrami



b. Rruga - për `m=2`, `n=3` dhe `x=1`



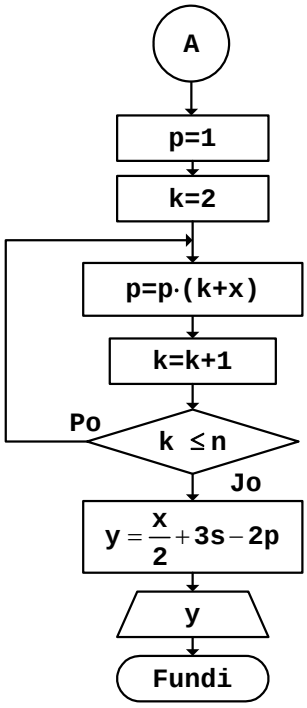


Fig.4.1

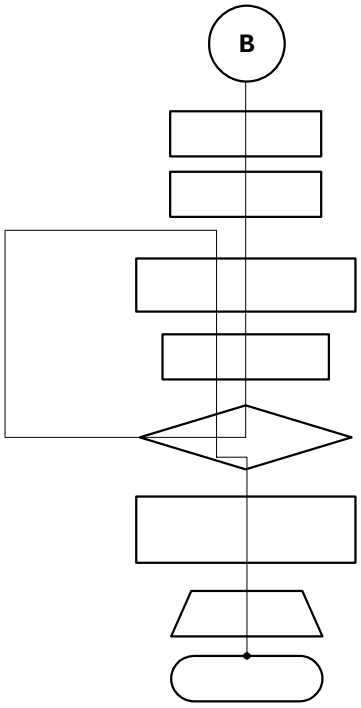


Fig.4.2

Vlera që llogaritet gjatë rrugës së kaluar në *Fig.4.2* është:

$$y = \frac{1}{2} + 3 \cdot \{(2 \cdot 1 + 3) + (2 \cdot 3 + 3) + (2 \cdot 5 + 3)\} - 2 \cdot \{(2 + 1) \cdot (3 + 1)\}$$

c. Programi

```
// Programi Prg4_1
#include <iostream>
using namespace std;
int main()
{
  int m,n,i,k;
  double s,p,x,y;
  cout << "Vlerat hyrëse m,n dhe x: ";
  cin >> m
      >> n
      >> x;
  s=0;
```

```
  i=1;
  do
  {
    s=s+(2*i+n);
    i=i+2;
  }
  while (i<=m+n);

  p=1;
  for (k=2;k<=n;k++)
    p=p*(k+x);

  y=x/2+3*s-2*p;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë, për vlerat e shfrytëzuara gjatë vizatimit të rrugës së kaluar në *Fig.4.2*, rezultati që shtypet në ekran është:

```
Vlera e funksionit y=57.5
```

Shuma dhe prodhimi mund të paraqiten njëkohësisht edhe në rastet kur funksionet përcaktohen me më shumë shprehje.

Shembull

Llogaritja e vlerës numerike të funksionit:

$$g=\begin{cases} 2x-3\sum\limits_{i=2}^{n+1}(i+x) & \text{për } x+1<4.5 \\ 4x^2+3x+1 & \text{për } x+1=4.5 \\ \frac{x}{2}+4\prod\limits_{\substack{k=1 \\ (k\neq 3)}}^{m}\left(k+\frac{x}{3}\right) & \text{për } x+1>4.5 \end{cases}$$

nëse janë dhënë vlerat e variablave `m`, `n` dhe `x`.

a. Bllok-diagrami

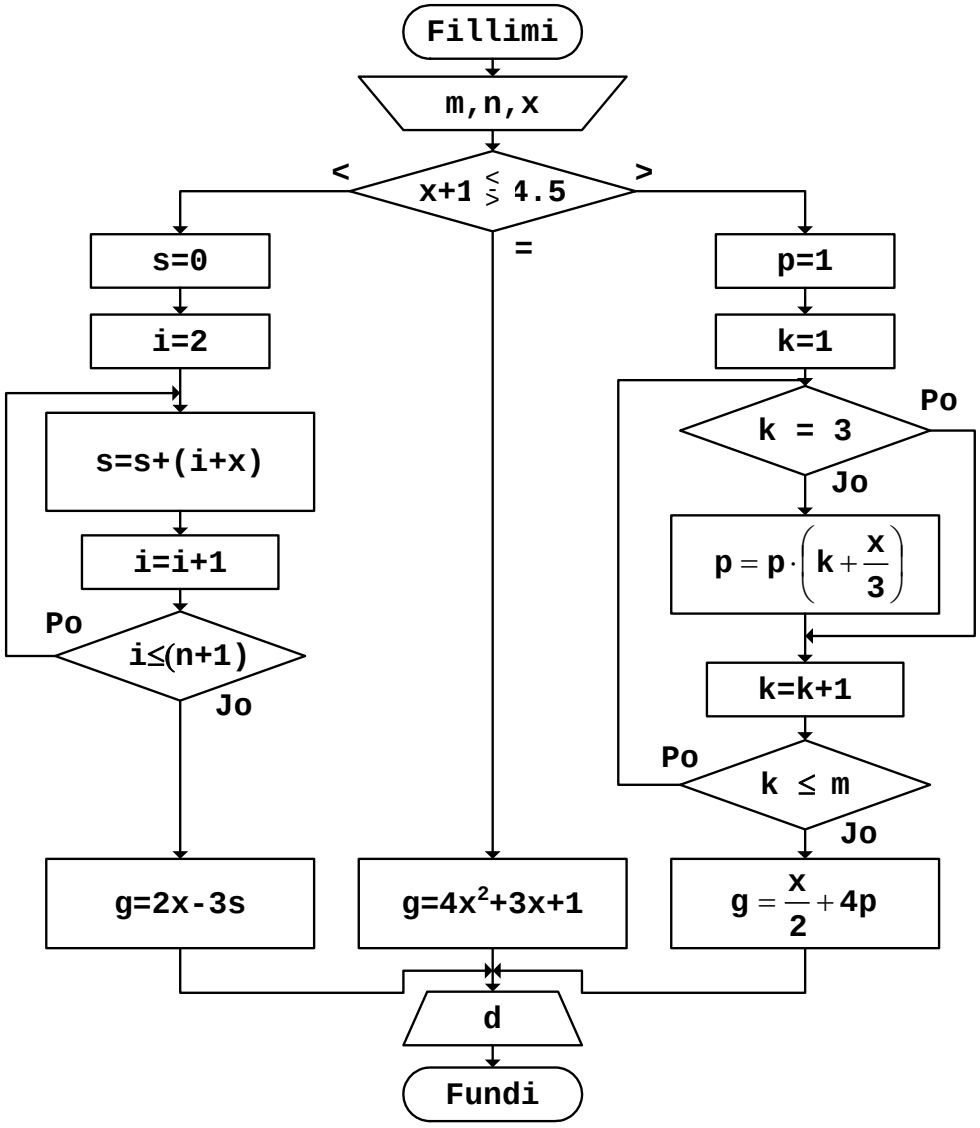


Fig.4.3

Këtu, versioni më i saktë i bllok-diagramit do të vizatohej nëse blloqet për leximin e variablave `m` dhe `n` vendosen në degën e majtë dhe të djathtë (përkatësisht), sepse vetëm në këto degë shfrytëzohen.

b. Rruga - për `m=4`, `n=4` dhe `x=2`

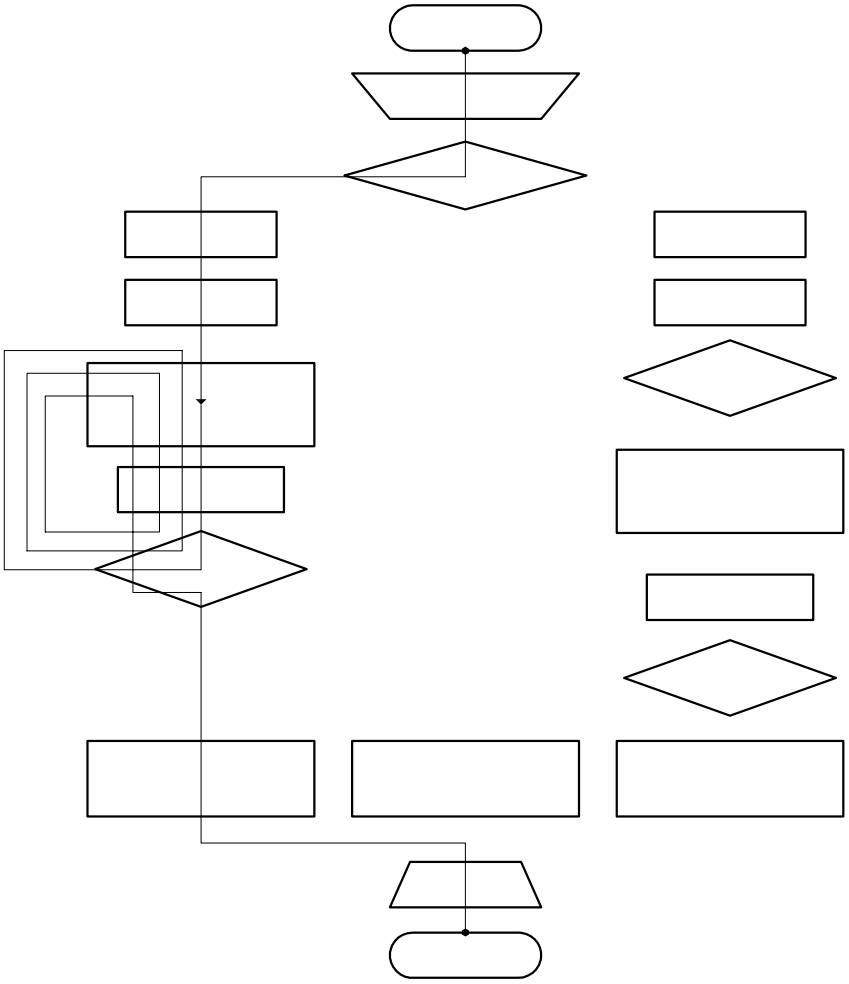


Fig.4.4

Gjatë rrugës së kaluar në *Fig.4.4*, meqë plotësohet kushti `x+1<4.5`, vlera e funksionit llogaritet përmes shprehjes së parë ku figuron shuma, kështu:

$$g = 2 \cdot 2 - 3 \cdot \underbrace{\{(2+2)+(3+2)+(4+2)+(5+2)\}}_{s}$$

c. Programi

```
// Programi Prg4_3
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int m,n,i,k;
  double s,p,x,g;
  cout << "Vlerat hyrëse m,n dhe x: ";
  cin >> m
      >> n
      >> x;
  if (x+1<4.5)
  {
    s=0;
    for (i=2;i<=n+1;i++)
        s=s+(i+x);
    g=2*x-3*s;
  }
  else
    if (x+1==4.5)
       g=4*pow(x,2)+3*x+1;
    else
    {
     p=1;
     for (k=1;k<=m;k++)
         if (k!=3)
            p=p*(k+x/3);
     g=x/2+4*p;
    }
  cout << "Vlera e llogaritur g="
       << g
       << "\n";
  return 0;
}
```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës i jepen vlerat e shfrytëzuara gjatë vizatimit të rrugës së kaluar, në ekran do të shtypet rezultati:

```
Vlera e llogaritur g=-62
```

Detyra

Të llogariten vlerat e funksioneve:

a.

$$y = x + 2\sum_{\substack{i=1\\(tek)}}^{n+1}(i+3)^2 - \frac{x+1}{\prod_{i=2}^{m}(i+1)}$$

b.

$$z = \begin{cases} \frac{x}{2} + 3\prod_{i=1}^{n}(2i+1) & \text{për } 2x > m \\ \frac{x}{3} - 2\sum_{\substack{k=1\\(k\neq 3,4)}}^{n}(k+x) & \text{për } 2x \leq m \end{cases}$$

c.

$$v = e^{2x} + \frac{1}{\sum_{i=1}^{m}(i+1)} - \frac{2\prod_{k=2}^{m}(k+1)}{3\sum_{k=1}^{n}(k+3)^2}$$

nëse janë dhënë vlerat e variablave m, n dhe x.

Llogaritja e faktorielit

Në praktikë shpesh herë përdoren edhe vlera të faktorielëve të ndryshëm. Llogaritja e faktorielit është përafërsisht e njëjtë me llogaritjen e prodhimit, sepse vlera fillestare e faktorielit merret `1`. Por, këtu duhet pasur kujdes në një dallim esencial të shprehjes e cila shkruhet brenda unazës në të cilën llogaritet faktorieli, sepse ajo gjithnjë e ka formën:

$$F = F \cdot i$$

ku `i` është një numërator.

Faktorieli i zakonshëm

Procedura për llogaritjen e faktorielit të zakonshëm, kur ai gjindet jashtë shprehjeve të shumave, prodhimeve ose edhe shprehjeve tjera, është e njëjtë me llogaritjen e prodhimit të numrave natyrorë.

Shembull Llogaritja e vlerës numerike të faktorielit:

`F=n!`

nëse është dhënë vlera e variablës `n`.

$$F = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n = \prod_{i=1}^{n} i$$

Brenda unazës përmes së cilës realizohet shumëzimi i numrave natyrorë mes `1` dhe `n`, shprehja për llogaritjen e faktorielit shkruhet ashtu siç u tha më sipër.

$$F = F \cdot i$$

ku përmes variablës `i` gjenerohen të gjithë numrat natyror mes `1` dhe `n`.

a. Bllok-diagrami *b. Rruga* - për `n=3`

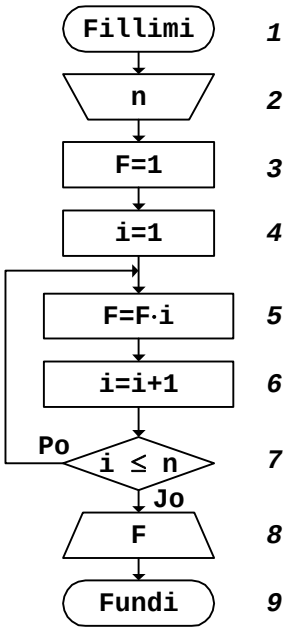


Fig. 5.1

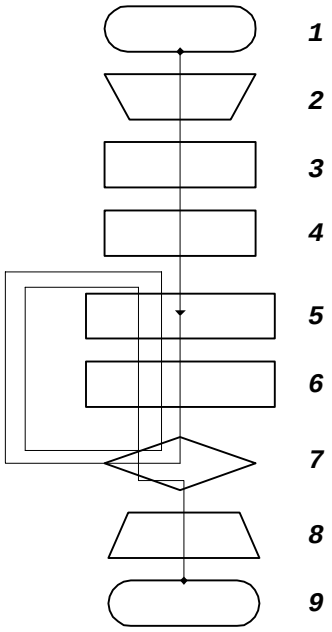


Fig.5.2

c. Testimi - për n=3

Hapi	Blloku	Urdhëri	Vlerat numerike merren	Rezultati
1	***1***	Fillimi	**-**	Fillimi i algoritmit
2	***2***	Lexo: n	prej njësisë hyrëse	n=3
3	***3***	F=1	**-**	F=1
4	***4***	i=1	**-**	i=1
5	***5***	F=F·i	F → **3**, i → **4**	F=1·1=1
6	***6***	i=i+1	i → **4**	i=1+1=2
7	***7***	Pyet:i≤n	i → **6**, n → **2**	Po
8	***5***	F=F·i	F → **5**, i → **6**	F=1·2=2
9	***6***	i=i+1	i → **6**	i=2+1=3
10	***7***	Pyet:i≤n	i → **9**, n → **2**	Po
11	***5***	F=F·i	F → **8**, i → **9**	F=2·3=6
12	***6***	i=i+1	i → **9**	i=3+1=4
13	***7***	Pyet:i≤n	i → **12**, n → **2**	Jo

14	***8***	Shtyp:F	F → **11**	Shtypet vlera 6
15	***9***	Fundi	-	Fundi i algoritmit

Fig.5.3

Vlera e faktorielit që fitohet gjatë procedurës së testimit është:

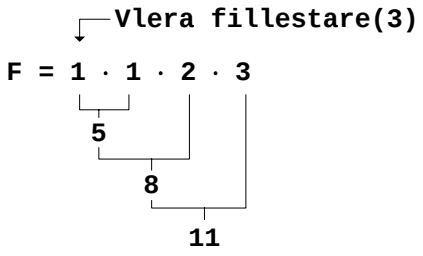


Fig.5.4

d. Programi

```
// Programi Prg5_1
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double F;
  cout << "Vlera hyrëse n: ";
  cin >> n;
  F=1;
  for (i=1;i<=n;i++)
      F=F*i;
  cout << "Faktorieli F="
       << F
       << "\n";
  return 0;
}
```

Nëse programi i dhënë ekzekutohet për vlerën hyrëse `n=3`, rezultati që shtypet në ekran është:

```
Faktorieli F=6
```

gjë që përputhet me vlerën e fituar gjatë testimit.

Njëlloj gjendet edhe vlera e faktorielit për shprehjet e ndryshme, rezultati i të cilave është një numër natyror.

Shembull Llogaritja e vlerës numerike të faktorielit:

$$F=(2m+n)!$$

nëse janë dhënë vlerat e variablave `m` dhe `n`.

$$F = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (2m + n) = \prod_{i=1}^{2m+n} i$$

a. Bllok-diagrami

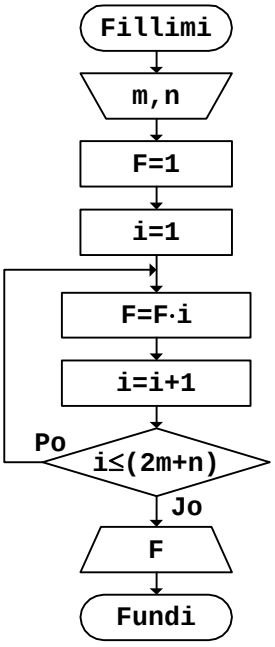


Fig.5.5

b. Rruga - për `m=1` dhe `n=2`

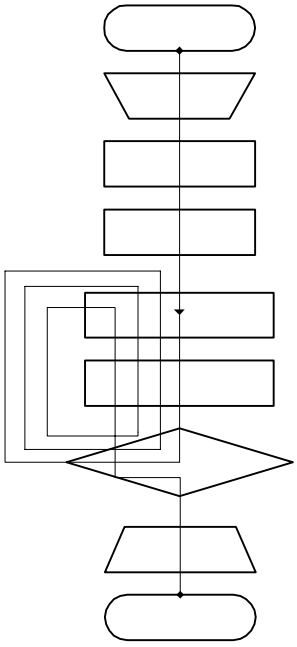


Fig.5.6

c. Programi

```
// Programi Prg5_5
#include <iostream>
using namespace std;
```

```
int main()

{
  int m,n,i;
  double F;
  cout << "Vlerat hyrëse m dhe n: ";
  cin >> m
      >> n;
  F=1;
  for (i=1;i<=2*m+n;i++)
      F=F*i;
  cout << "Faktorieli F="
       << F
       << "\n";
  return 0;
}
```

Nëse gjatë ekzekutimit të programit, si vlera hyrëse përmes tastierës jepen vlerat `m=1` dhe `n=2`, në ekran do të shtypet:

```
Faktorieli F=24
```

ku vlera `24` i përgjigjet `4!`.

Procesi i llogaritjes do të ketë një cikël më pak, nëse gjatë llogaritjes së faktorielit të një numri më të madhë se `1` nisemi prej vlerës fillestare `F=1` dhe vlera e variablës merret `i=2`. Ngjashëm, kur llogaritet faktorieli i një numri më të madh se `2`, procesi i llogaritjes do të zvogëlohet për dy cikle, nëse merret `F=2` dhe `i=3`.

Nëse faktorieli paraqitet brenda shprehjeve të funksioneve, veprohet njëlloj si edhe te shumat e prodhimet, përkatësisht së pari llogaritet vlera e faktorielit dhe pastaj vlera e funksionit.

Shembull Llogaritja e vlerës numerike të funksionit:

$$y = \begin{cases} (m+1)! + 3x - 4 & \text{për } x + m > 5 \\ 0 & \text{për } x + m = 5 \\ (3n+2)! - x^2 & \text{për } x + m < 5 \end{cases}$$

nëse janë dhënë vlerat e variablave `m`, `n` dhe x.

a. Bllok-diagrami

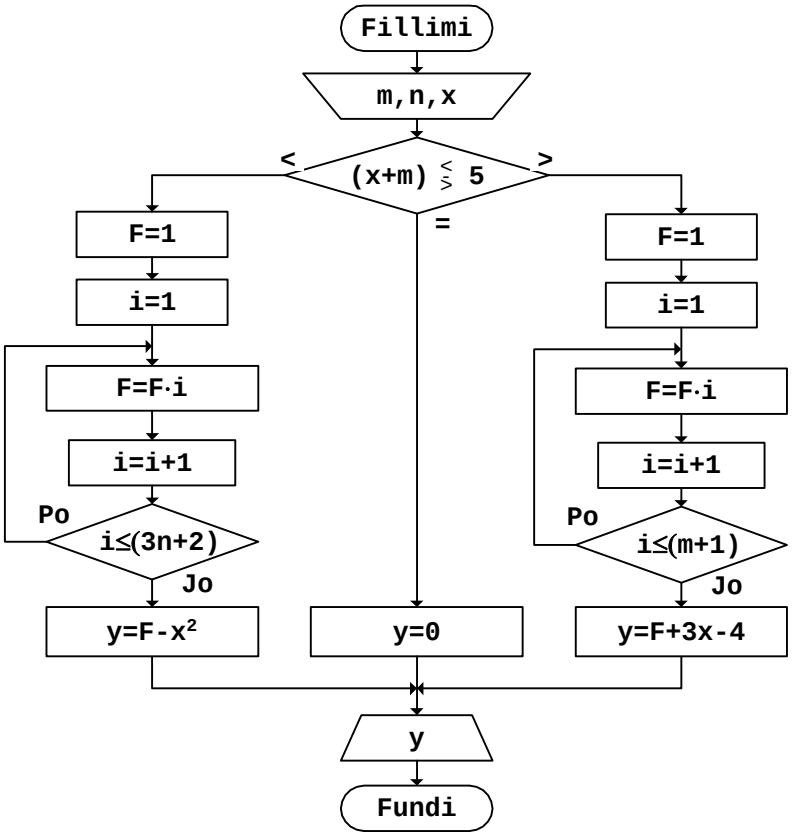


Fig. 5.7

b. Rruga - për `m=2`, `n=1` dhe `x=5`

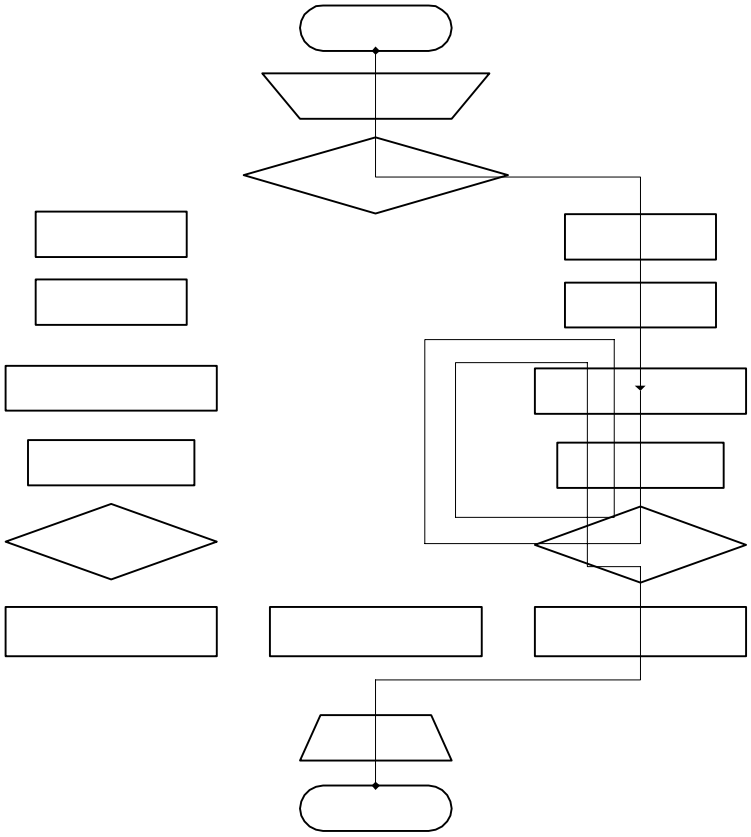


Fig.5.8

Vlera që llogaritet gjatë rrugës së kaluar është:

$$y = \underbrace{1 \cdot 2 \cdot 3}_{F} + 3 \cdot 5 - 4$$

c. Programi

```
// Programi Prg5_7
#include <iostream>
#include <cmath>
using namespace std;
int main()
```

```
{
  int m,n,i;
  double F,x,y;
  cout << "Vlerat hyrëse m,n dhe x: ";
  cin >> m
      >> n
      >> x;
  if (x+m<5)
  {
    F=1;
    for (i=1;i<=3*n+2;i++)
        F=F*i;
    y=F-pow(x,2);
  }
  else
    if (x+m==5)
       y=0;
    else
    {
     F=1;
     for (i=1;i<=m+1;i++)
         F=F*i;
     y=F+3*x-4;
    }
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat hyrëse që janë shfrytëzuar gjatë vizatimit të rrugës në *Fig.5.8*, si rezultat në ekran do të shtypet:

```
Vlera e funksionit y=17
```

gjë që fitohet ashtu siç u tregua edhe më sipër.

Që algoritmi për llogaritjen e faktorielit të jetë plotësisht i saktë, në te mund të parashihet edhe rasti kur duhet të llogaritet vlera `0!=1`.

Detyra

Të llogariten vlerat e funksioneve:

a.

$$y = \frac{x}{2} - (3n + m + 1)!$$

b.

$$z = \begin{cases} 3x + 2 & \text{për } x < m \\ (m+1)! + \dfrac{x}{(m+n)!} & \text{për } x \geq m \end{cases}$$

c.

$$g = \begin{cases} e^{2x} - 3x^2 + (4n)! & \text{për } x + 3 < n \\ (n+1)! - 2x & \text{për } x + 3 = n \\ (n!)^2 + \dfrac{1}{(2n)!} & \text{për } x + 3 > n \end{cases}$$

nëse janë dhënë vlerat e variablave m, n dhe x.

Faktorieli çift dhe tek

Në matematikë përdoret edhe faktorieli i cili fitohet vetëm me shumëzimin e numrave natyrorë çiftë, ose të numrave natyrorë tekë.

Shembull Llogaritja e vlerës numerike të faktorielit çift:

$$F = (2n)!!$$

nëse është dhënë vlera e variablës n.

$$F = 2 \cdot 4 \cdot 6 \cdot \ldots \cdot 2n = \prod_{\substack{i=2 \\ (\text{çift})}}^{2n} i$$

a. Bllok-diagrami

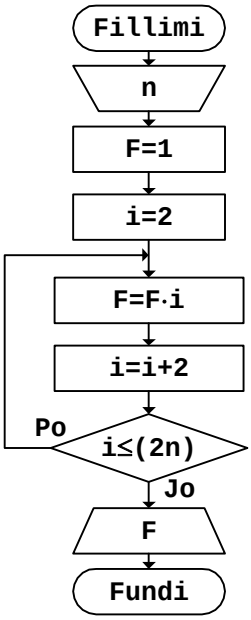


Fig.5.9

c. Rruga - për n=3

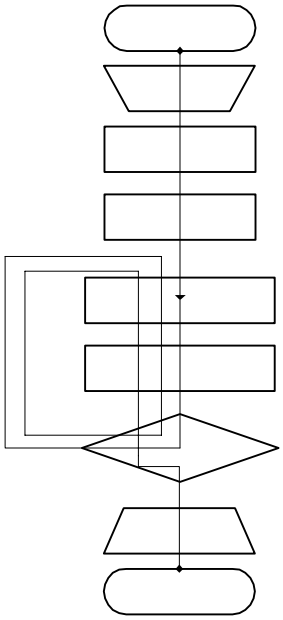


Fig.5.10

c. Programi

```
// Programi Prg5_9
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double F;
  cout << "Vlera hyrëse n: ";
  cin >> n;
  F=1;i=2;
  do
  {
    F=F*i;
    i=i+2;
  }
  while (i<=2*n);
  cout << "Faktorieli çift F="
       << F
       << "\n";
  return 0;
```

```
}
```

Për vlerën hyrëse `n=3`, rezultati që shtypet në ekran është:

```
Faktorieli çift F=48
```

Plotësisht njëlloj llogaritet edhe vlera e faktorielit tek. Kështu, p.sh., nëse kërkohet vlera e faktorielit:

```
F=(2n-1)!!
```

në bazë të përkufizimit përkatës matematikor kjo vlerë llogaritet:

$$F = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1) = \prod_{\substack{i=1 \\ (tek)}}^{2n-1} i$$

gjë që nuk është aspak rëndë të realizohet përmes bllok-diagramit.

Faktorieli brenda shumës

Faktorielët mund të paraqiten brenda shprehjeve të shumave, ose edhe brenda funksioneve të ndryshme.

Shembull Llogaritja e vlerës numerike të funksionit:

$$y = 3x - 2\sum_{i=1}^{n}\left[(n+2)! + \frac{i}{3}\right]$$

nëse janë dhënë vlerat e variablave x dhe n.

$$y = 3x - 2\left\{\left[(n+2)! + \frac{1}{3}\right] + \left[(n+2)! + \frac{2}{3}\right] + \ldots + \left[(n+2)! + \frac{n}{3}\right]\right\}$$

Meqë, siç shihet edhe më sipër, vlera e faktorielit është konstante brenda të gjithë anëtarëve të serisë që mblidhen, atë duhet llogaritur së pari, për ta shfrytëzuar pastaj gjatë procesit të gjetjes së shumës.

a. Bllok-diagrami

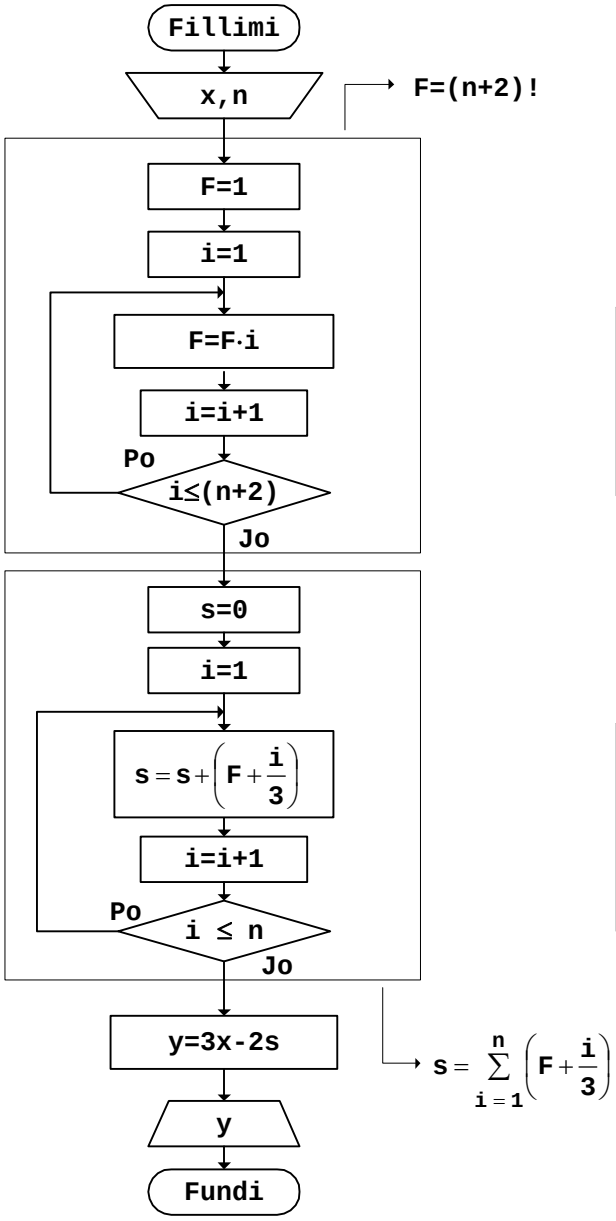


Fig.5.11

b. Rruga - për n=2 dhe x=3

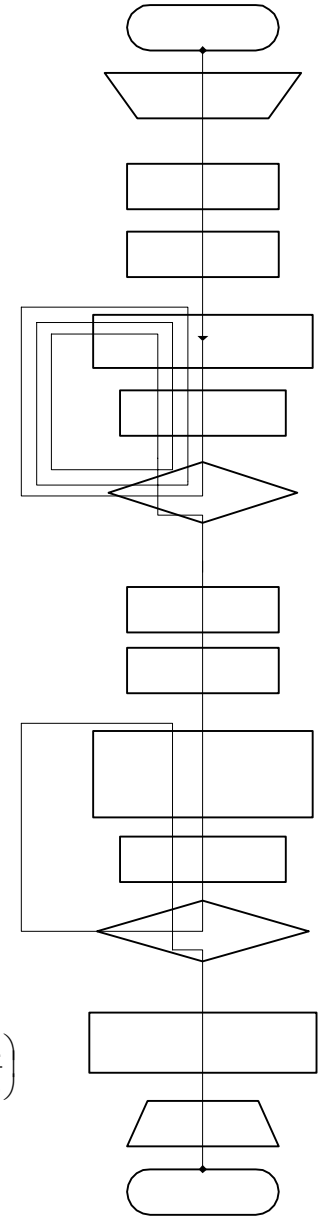


Fig.5.12

c. Programi

```
// Programi Prg5_11
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double x,y,F,s;
  cout << "Vlerat hyrëse x dhe n: ";
  cin >> x
      >> n;
  F=1;
  for (i=1;i<=n+2;i++)
      F=F*i;
  s=0;
  for (i=1;i<=n;i++)
      s=s+(F+i/3.);
  y=3*x-2*s;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Nëse programi i dhënë ekzekutohet për vlerat e shfrytëzuara gjatë vizatimit të rrugës në *Fig.5.12*, rezultati që shtypet në ekran do të duket:

```
Vlera e funksionit y=-89
```

Faktorieli brenda shumës mund të mos ketë vlerë konstante, përkatësisht të mos varet nga vlera e variablës e cila i përcakton anëtarët e serisë që mblidhen.

Shembull Llogaritja e vlerës numerike të funksionit:

$$y = 3x - 2\sum_{i=1}^{n}\left\lfloor (i+2)! + \frac{i}{3}\right\rfloor$$

nëse janë dhënë vlerat e variablave x dhe n.

$$y = 3x - 2\left\{\left\lfloor 3! + \frac{1}{3}\right\rfloor + \left\lfloor 4! + \frac{2}{3}\right\rfloor + \ldots + \left\lfloor (n+2)! + \frac{n}{3}\right\rfloor\right\}$$

Nga shprehja e dhënë më sipër shihet se vlera e faktorielit nuk është konstante, por ajo ndryshon për çdo `i`, gjë që e imponon nevojën e llogaritjes së kësaj vlere brenda unazës për llogaritjen e shumës. Meqë në anëtarin e parë i cili merr pjesë në mbledhje figuron `3!`, vlera fillestare e faktorielit duhet të merret `2!=2` dhe pastaj të shumëzohet me `(i+2)`.

a. Bllok-diagrami

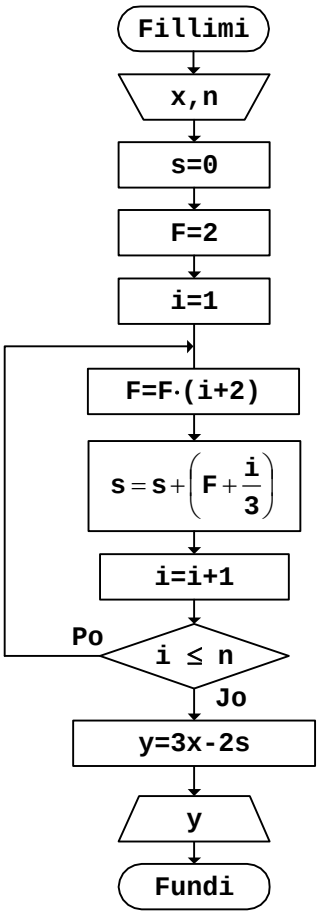


Fig.5.13

b. Rruga - për `n=2` dhe `x=3`

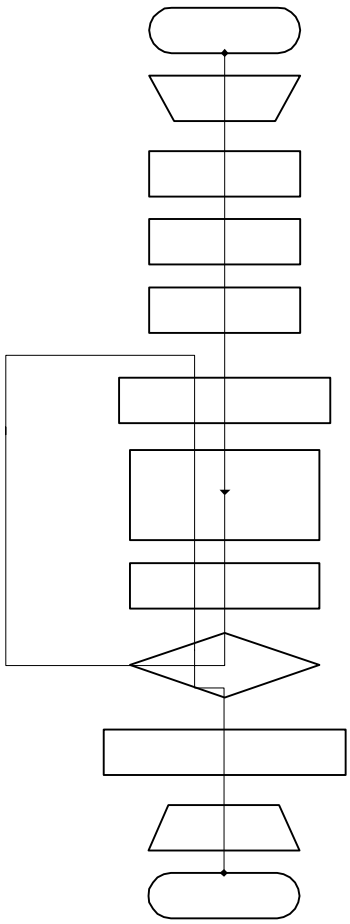


Fig.5.14

c. Programi

```
// Programi Prg5_13
#include <iostream>
using namespace std;
int main()
{
  int n,i;
  double x,y,F,s;
  cout << "Vlerat hyrëse x dhe n: ";
  cin >> x
      >> n;
  s=0;
  F=2;
  for (i=1;i<=n;i++)
  {
    F=F*(i+2);
    s=s+(F+i/3.);
  }
  y=3*x-2*s;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë, p.sh. për vlerat hyrëse të shfrytëzuara gjatë vizatimit të rrugës së kaluar, rezultati që shtypet në ekran është:

```
Vlera e funksionit y=-53
```

Gjetja e ligjshmërisë për ta fituar vlerën aktuale të faktorielit, duke shfrytëzuar vlerën paraprake të tij, në rast të përgjithshëm e komplikon punën e shkruarjes së programit për llogaritjen e vlerës së faktorielit. Më thjesht është nëse brenda unazës së shumës çdo vlerë e faktorielit llogaritet prej fillimit, pa menduar aspak se a mund të thjeshtohet llogaritja, nëse shfrytëzohet vlera e faktorielit e cila është llogaritur paraprakisht.

Shembull Llogaritja e vlerës numerike të funksionit:

$$y = \frac{x}{2} - 3\sum_{i=1}^{n}[(2i+1)! - x]$$

nëse janë dhënë vlerat e variablave x dhe n.

$$y = \frac{x}{2} - 3 \cdot \{[(2 \cdot 1 + 1)! - x] + [(2 \cdot 2 + 1)! - x] + \ldots + [(2 \cdot n + 1)! - x]\}$$

a. Bllok-diagrami

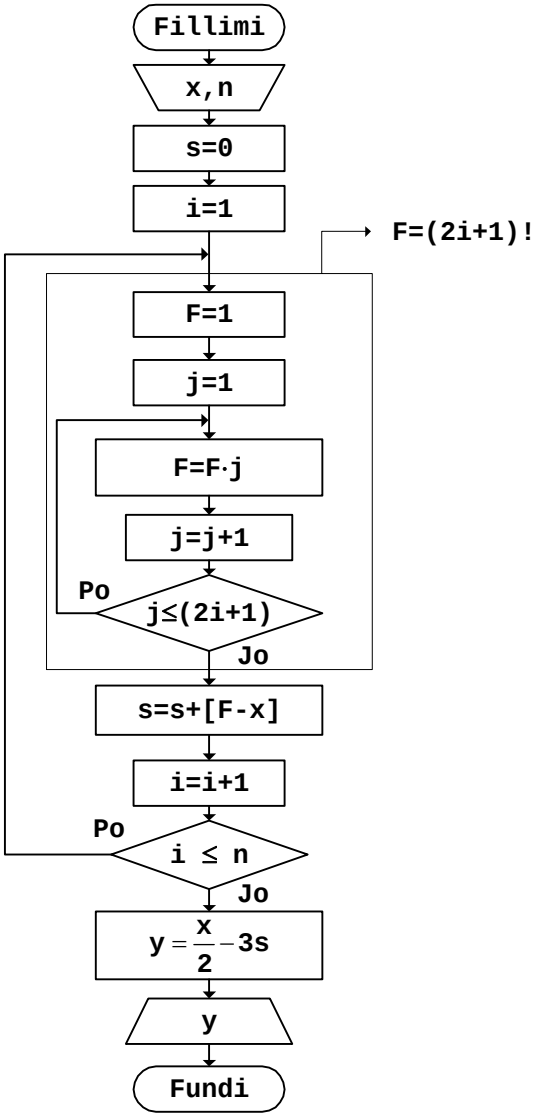


Fig.5.15

Nëse vizatohet rruga e kaluar në këtë rast, numri i vijave te pjesa e bllok-diagramit për llogaritjen e faktorielit do të jetë i madh, sepse llogaritja përsëritet prej fillimit për çdo vlerë të variablës `i`.

b. Programi

```
// Programi Prg5_15
#include <iostream>
using namespace std;
int main()
{
  int n,i,j;
  double x,y,s,F;
  cout << "Vlerat hyrëse x dhe n: ";
  cin >> x
      >> n;
  s=0;
  for (i=1;i<=n;i++)
  {
    F=1;
    for (j=1;j<=2*i+1;j++)
        F=F*j;
    s=s+(F-x);
  }
  y=x/2-3*s;
  cout << "Vlera e funksionit y="
       << y
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat hyrëse `x=3` dhe `n=2`, rezultati që shtypet në ekran duket:

```
Vlera e funksionit y=-358.5
```

Plotësisht njëlloj rrjedh llogaritja e faktorielëve, kur ata figurojnë brenda anëtarëve të serive për të cilët gjenden prodhimet.

Gjatë llogaritjes së faktorielit çift ose tek, kur ata figurojnë nën shenjën e shumës ose të prodhimit, duhet pasur kujdes të veçantë.

Shembull Llogaritja e vlerës numerike të funksionit:

$$y = 3x + 4\sum_{i=1}^{n+1}[(2i+1)!! + i]$$

nëse janë dhënë vlerat e variablave x dhe n.

$$y = 3x + 4\{[3!! + 1] + [5!! + 2] + \ldots + [2(n+1)!! + (n+1)]\}$$

a. Bllok-diagrami

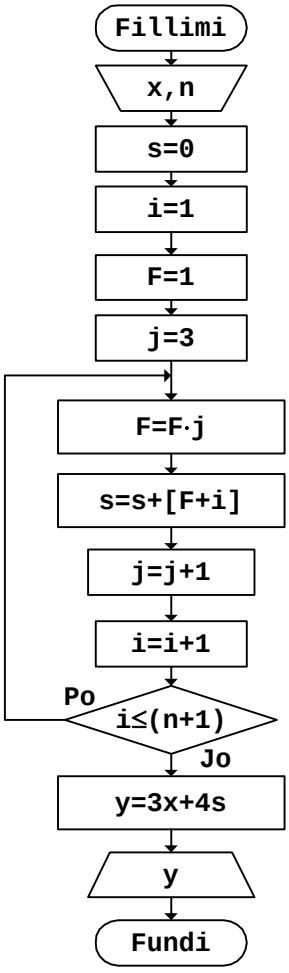


Fig.5.16

b. Rruga - për x=2 dhe n=2

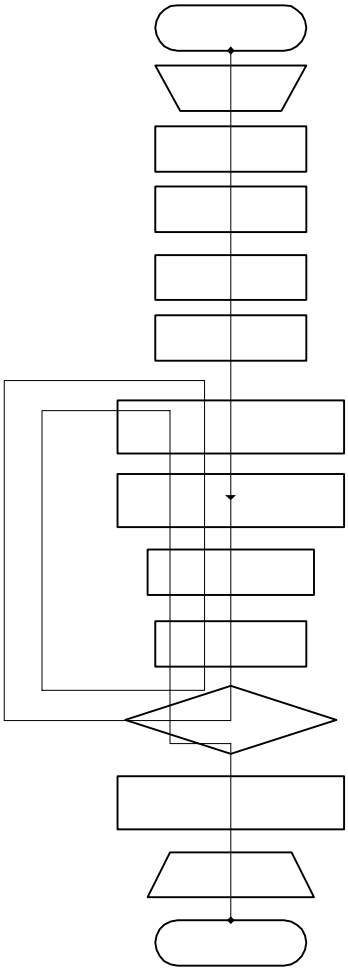


Fig.5.17

c. Programi

```
// Programi Prg5_16
#include <iostream>
using namespace std;
int main()

{
```

```
  int n,i,j;
  double x,y,s,F;
  cout << "Vlerat hyrëse x dhe n: ";
  cin >> x
      >> n;
  s=0;
  i=1;
  F=1;
  j=3;
  do
  {
    F=F*j;
    s=s+(F+i);
    j=j+2;
    i=i+1;
  }
  while (i<=n+1);
  y=3*x+4*s;
  cout << "Vlera e llogaritur y="
       << y
       << "\n";
  return 0;
}
```

Rezultati që do të shtypet në ekran, nëse ekzekutohet programi i dhënë për vlerat e variablave hyrëse, të cilat janë marrë gjatë vizatimit të rrugës së kaluar në *Fig.5.17*, është:

```
Vlera e llogaritur y=522
```

Në një shprehje të funksionit mund të ndodhë të nevojitet llogaritja alternative e faktorielit çift dhe tek.

Shembull Llogaritja e vlerës numerike të funksionit:

$$g = 2a + 3\sum_{i=1}^{n}\left\lfloor i!! + \frac{x}{2}\right\rfloor$$

nëse janë dhënë vlerat e variablave a, x dhe n.

$$g = 2a + 3 \cdot \left\{ \left[1!! + \frac{x}{2} \right] + \left[2!! + \frac{x}{2} \right] + \ldots + \left[n!! + \frac{x}{2} \right] \right\}$$

a. Bllok-diagrami

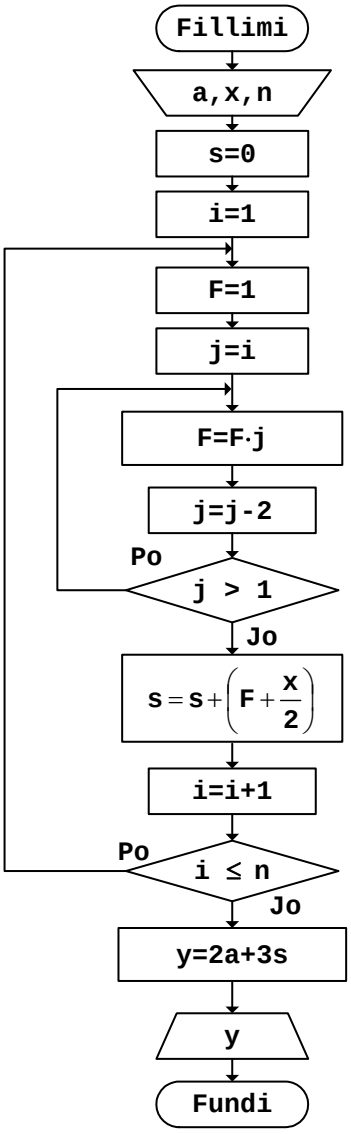


Fig.5.18

b. Rruga - për a=5, x=2 dhe n=4

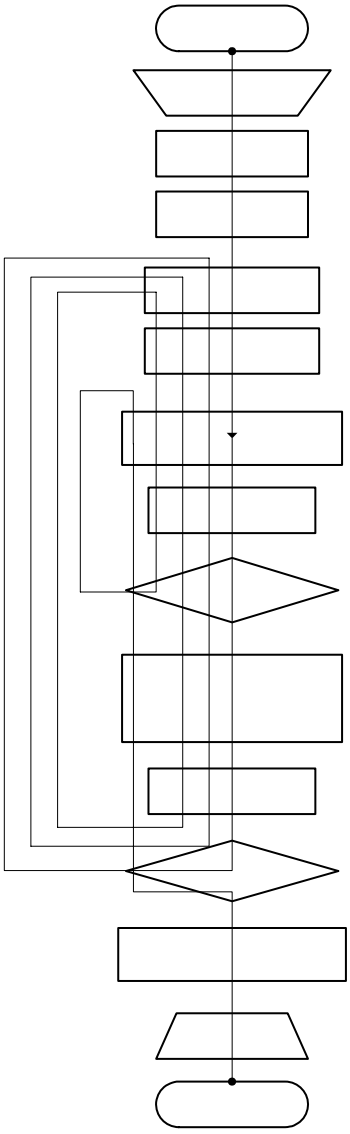


Fig.5.19

c. Programi

```
// Programi Prg5_18
#include <iostream>
using namespace std;
int main()
{
  int n,i,j;
  double a,x,y,s,F;
  cout << "Vlerat e variablave a, x dhe n: ";
  cin >> a
      >> x
      >> n;
  s=0;
  for (i=1;i<=n;i++)
  {
    F=1;
    j=i;
    do
    {
       F=F*j;
       j=j-2;
    }
    while (j>1);
    s=s+(F+x/2);
  }
  y=2*a+3*s;
  cout << "Vlera e llogaritur y="
       << y
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit për vlerat e variablave hyrëse, të cilat janë shfrytëzuar gjatë vizatimit të rrugës së kaluar në *Fig.5.19*, në ekran fitohet:

```
Rezultati y=64
```

Detyra

Të llogariten vlerat e funksioneve:

a.

$$y = \frac{3x}{a} + 4\sum_{i=2}^{n+2}\left[(2i+3)! + \frac{i}{b}\right]^2$$

b.

$$g = \begin{cases} (n+1)! & \text{për } x < (a+b) \\ 3\sum_{i=1}^{m}\left[(2i+1)!! + \frac{a}{i}\right] & \text{për } x = (a+b) \\ (2n)!! & \text{për } x > (a+b) \end{cases}$$

nëse janë dhënë vlerat e variablave a, b, x, m dhe n.

Në shprehjet e funksioneve njëkohësisht mund të paraqiten shuma, prodhime dhe faktoriele, të cilat llogariten duke u mbështetur në algoritmet elementare përkatëse.

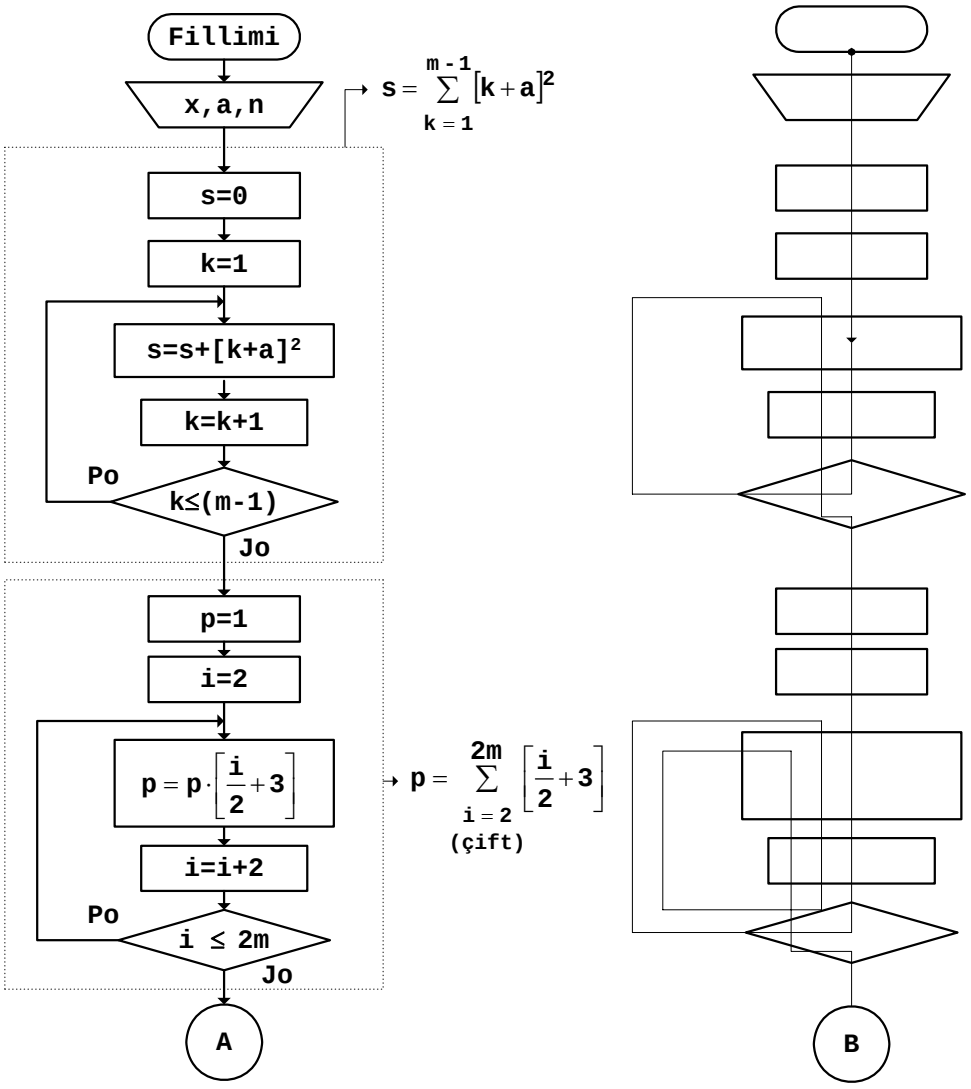
Shembull Llogaritja e vlerës numerike të funksionit:

$$z = 3\sum_{k=1}^{m-1}[k+a]^2 - 4\prod_{\substack{i=2 \\ (\text{çift})}}^{2m}\left[\frac{i}{2}+3\right] + \frac{(m+1)!}{x+2}$$

nëse janë dhënë vlerat e variablave x, a dhe m.

$$z = 3\cdot\left\{[1+a]^2 + [2+a]^2 + \ldots + [(m+1)+a]^2\right\} - 4\left\{\left[\frac{2}{2}+3\right]\cdot\left[\frac{4}{2}+3\right]\cdot\ldots\cdot\left[\frac{2m}{2}+3\right]\right\} + \frac{1\cdot 2\cdot 3\cdot\ldots\cdot(m+1)}{x+2}$$

a. Bllok-diagrami *b. Rruga* - për `x=3`, `a=0.5` dhe `m=3`



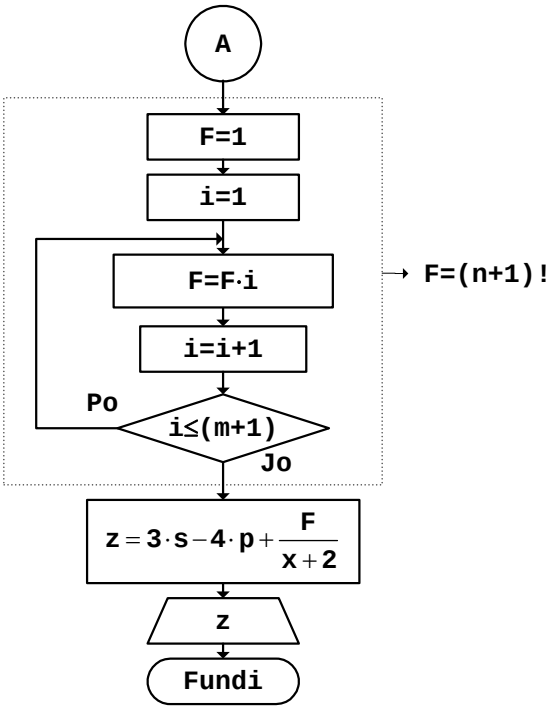


Fig.5.20

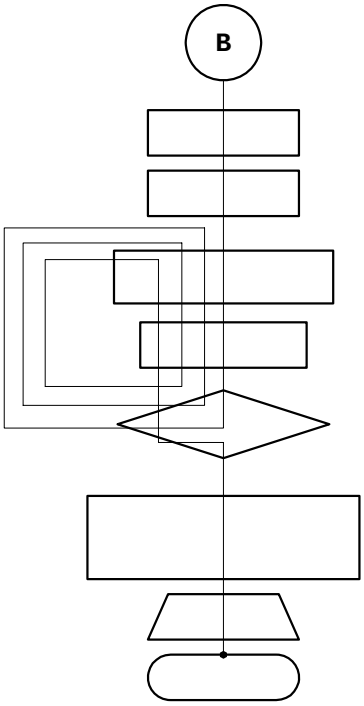


Fig.5.21

c. Programi

```
// Programi Prg5_20
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  const double x=3,a=0.5;
  const int m=3;
  int i,k;
  double s,p,F,z;
  s=0;
  for (k=1;k<=m-1;k++)
      s=s+pow(k+a,2);
  p=1;
  i=2;
  do
  {
    p=p*(i/2.+3);
```

```
    i=i+2;
  }
  while (i<=2*m);
  F=1;
  for (i=1;i<=m+1;i++)
      F=F*i;
  z=3*s-4*p+F/(x+2);
  cout << "Rezultati z="
       << z
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat hyrëse të deklaruara si konstante, rezultati që shtypet në ekran është:

```
Rezultati z=-449.7
```

Detyra

Të llogariten vlerat numerike të funksioneve:

a.

$$f = \frac{x}{3} + 2\sum_{\substack{i=1 \\ (i\neq 3)}}^{n+1}[(2i)!! + 3] - [(n+2)!]^2$$

b.

$$g = \begin{cases} (n+1)! + 3\prod_{\substack{k=1 \\ (tek)}}^{n}[k+1] & \text{për } (a+x) < b \\ (2n+1)!! + 2a - b & \text{për } (a+x) = b \\ (2n)!! + \frac{x}{2} & \text{për } (x+a) > b \end{cases}$$

nëse janë dhënë vlerat e variablave a, b, x dhe n.

Faktorieli brenda prodhimit

Faktorielët mund të paraqiten edhe brenda anëtarëve të serive për të cilat kërkohet prodhimi. Gjatë përpilimit të algoritmeve përkatëse, në këto raste shfrytëzohen procedura plotësisht të njëjta si edhe te shumat.

Shembull Llogaritja e vlerës numerike të funksionit:

$$y = 3x + 2 \prod_{\substack{i=1 \\ (i \neq 2,3)}}^{n} \left[(i+1)! + \frac{i}{3} \right]$$

nëse janë dhënë vlerat e variablave x dhe n.

$$y = 3x + 2 \cdot \left\{ \left[2! + \frac{1}{3} \right] \cdot \left[5! + \frac{4}{3} \right] \cdot \ldots \cdot \left[(n+1)! + \frac{n+1}{3} \right] \right\}$$

a. Bllok-diagrami

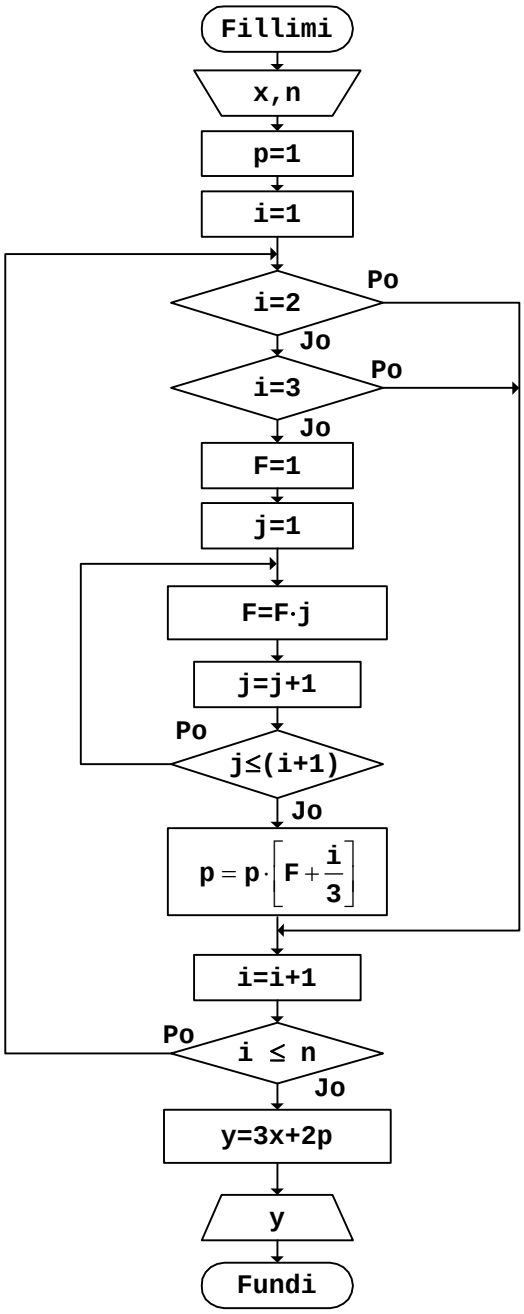


Fig.5.22

b. *Rruga* - për x=2 dhe n=4

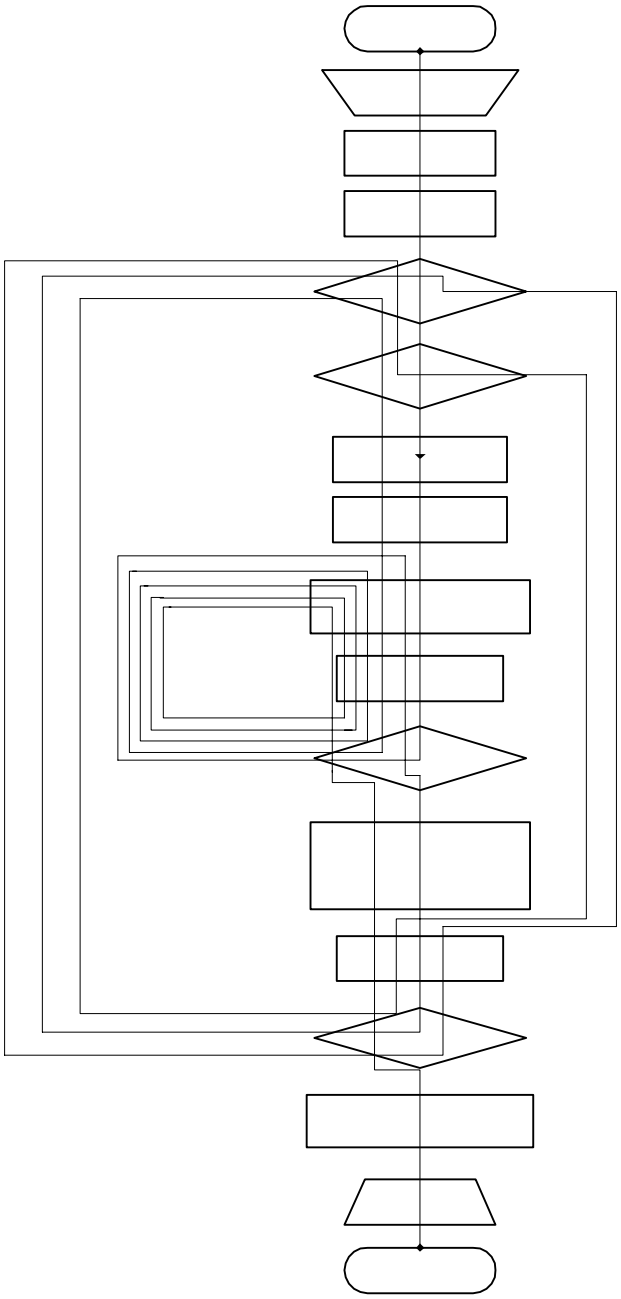


Fig.5.23

c. Programi

```
// Programi Prg5_22
#include <iostream>
using namespace std;
int main()
{
  int n,i,j;
  double x,y,p,F;
  cout << "Vlerat hyrëse x dhe n: ";
  cin >> x
      >> n;
  p=1;
  for (i=1;i<=n;i++)
      if ((i==2) || (i==3))
      {
      }
      else
      {
         F=1;
         for (j=1;j<=i+1;j++)
              F=F*j;
         p=p*(F+i/3.);
      }
  y=3*x+2*p;
  cout << "Vlera e llogaritur y="
       << y
       << "\n";
  return 0;
}
```

Nëse programi ekzekutohet për vlerat hyrëse të shfrytëzuara gjatë vizatimit të rrugës së kaluar, në ekran do të fitohet rezultati:

```
Vlera e llogaritur y=572.222
```

Detyra

Të llogariten vlerat e funksioneve:

a.

$$f = \frac{x}{2} - 3\prod_{k=1}^{m}\left[k!! + \frac{x}{2}\right]$$

b.

$$g = \begin{cases} 3x + 2\sum_{i=1}^{m+1}\left\lfloor \frac{i}{2} + 3 \right\rfloor & \text{për } x \neq a \\ e^{ax} + 3\prod_{\substack{j=2 \\ (\text{çift})}}^{m}\left[(2j+3)! - \frac{x}{a}\right]^2 & \text{për } x = a \end{cases}$$

c.

$$h = \frac{1}{3} - \sum_{\substack{i=1 \\ (i \neq 3,5)}}^{m+2}\left[\frac{i}{3} + 4i\right] - \frac{2}{3}\prod_{j=1}^{m}\left[(j!)^2 - \frac{a}{j}\right]$$

nëse janë dhënë vlerat e variablave `x`, `a` dhe `m`.

Fushat numerike

Grumbulli i numrave të vendosur në një hapësirë në bazë të parimeve të caktuara, quhet *fushë numerike*. Kur pozicionet e numrave në hapësirën e fushës numerike përcaktohen nga një madhësi, për fushën thuhet se është *njëdimensionale* dhe quhet *vektor*. Nëse për përcaktimin e pozicioneve të numrave përdoren dy madhësi, fusha numerike është *dydimensionale* dhe quhet *matricë*. Kurse, kur pozita e numrave në fushë përcaktohet përmes më shumë madhësive, për fushën thuhet se është *shumëdimensionale*.

Për t'i kuptuar më mirë fushat numerike, do ta marrim si shembull grumbullin e notave të 5 nxënësve të parë, në regjistrin e notave të një klase, i cili është dhënë në *Fig.6.1*.

Numri rendor	Emri	1 Gjuha amtare	2 Gjuha e huaj	3 Matematika	4 Kimia	5 Fizika	6 Biologjia	7 Programimi	8 Edukata fizike
1	Agroni	5	5	5	5	5	5	5	5
2	Ardiani	5	5	5	4	4	4	5	5
3	Arianisa	5	5	5	5	5	5	5	5
4	Arta	5	5	5	5	5	5	5	5
5	Besa	5	5	4	4	4	5	5	4

Fig.6.1 Notat e nxënësve të një klase

Këtu, grumbulli i notave të një nxënësi është fushë njëdimensionale dhe e paraqet *vektorin e notave* të tij. Kështu, p.sh., vektori i notave të Ardianit është:

	1	2	3	4	5	6	7	8
A =	5	5	5	4	4	4	5	5

Notat në këtë vektor nuk janë shënuar arbitrarisht, sepse çdo pozicioni në te i përgjigjet nota e një lënde të caktuar.

Grumbulli i notave të nxënësve, të cilat janë shënuar në tabelën e dhënë në *Fig.6.1*:

B =	1	2	3	4	5	6	7	8
1	5	5	5	5	5	5	5	5
2	5	5	5	4	4	4	5	5
3	5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	5	5
5	5	5	4	4	4	5	5	4

paraqetë fushë dydimensionale dhe e formon *matricën e notave* të tyre.

Nga shembulli i dhënë më sipër shihet se te vektorët e notave të nxënësve, notat shënohen në bazë të lëndëve, përkatësisht numrave rendorë të tyre. Kurse, te matrica e notave të nxënësve, vendosja e notave në fushat e veçanta bëhet në bazë të nxënësve dhe lëndëve, ose numrave rendorë të tyre.

Në rastin e përgjithshëm, kur kemi të bëjmë me `m`-nxënës dhe `n`-lëndë, vektori i notave të nxënësit mund të paraqitet kështu:

A =	1	2	...	n
	a_1	a_2	...	a_n

ose shkurt `A(n)`, ku a_1, a_2, ..., a_n janë anëtarët e vektorit. Matrica e notave të nxënësve është:

B = ...	j=1	2	...	n
i=1	b_{11}	b_{12}	...	b_{1n}
2	b_{21}	b_{22}	...	b_{2n}
...	...	...	...	...
m	b_{m1}	b_{m2}	...	b_{mn}

dhe shkurt shënohet `B(m,n)`, ku b_{11}, b_{12}, ..., b_{m1}, b_{m2},..., b_{mm} janë anëtarët e matricës.

Numrat të cilët i shoqërojnë anëtarët e vektorit, ose të matricës, quhen *indekse*. Kështu, p.sh., indeksi i notës së matematikës te vektorët e notave të nxënësve është `3`, kurse anëtari i këtij vektori për notën e `Artës` është a_3=5. Te matrica, p.sh., anëtari b_{53}=4 i përgjigjet notës së `Besës` nga lënda e matematikës.

Vektorët

Mbushja e vektorëve me vlera numerike, përkatësisht operimi me vlerat e anëtarëve të veçantë të tyre bëhet duke i shfrytëzuar indekset përkatës.

Përcaktimi i vektorëve

Vlerat numerike të anëtarëve të vektorëve kompjuterit mund t'i jepen përmes leximit si numra të gatshëm, ose ato mund të llogariten në bazë të ligjshmërisë së dhënë.

Shembull Formimi i vektorit `A(n)`, duke i llogaritur anëtarët a_i të tij kështu:

$$a_i=3i+1$$

nëse dihet vlera e variablës `n`.

a. Bllok-diagrami

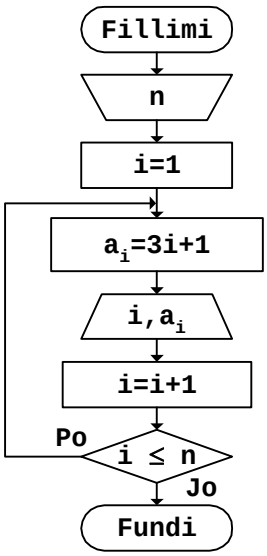


Fig.6.2

b. Rruga - për `n=5`

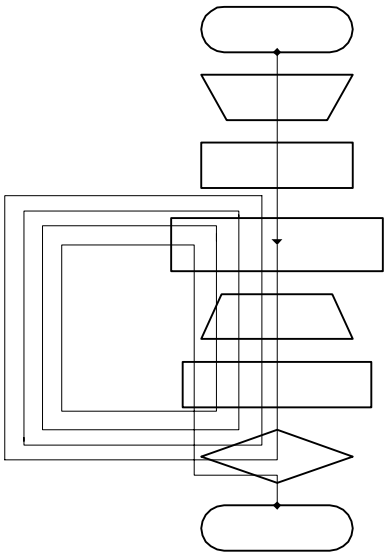


Fig.6.3

Në gjuhën `C++` indekset e anëtarëve të vektorëve dhe matricave fillojnë me vlerën `0`. Për këtë arsye, në pjesën vijuese, gjatë shkruarjes së programeve, indekset fillestare dhe kufijt e tyre do të zvogëlohen për një, krahasuar me vlerat përkatëse në bllok-diagrame.

c. Programi

```
// Programi Prg6_2
#include <iostream>
using namespace std;
int main()
{
  int const n=5;
  int i,A[n];
  for (i=0;i<n;i++)
  {
    A[i]=3*i+1;
    cout << "A["
         << i
         << "]="
         << A[i]
         << "\n";
  }
  return 0;
}
```

Pas ekzekutimit të programit, anëtarët e vektorit do të shtypen në ekran kështu:

```
A[0]=1
A[1]=4
A[2]=7
A[3]=10
A[4]=13
```

Anëtarët e vektorit mund edhe të llogariten, duke shfrytëzuar shprehje të çfarëdoshme.

Shembull

Formimi i vektorit `A(n)`, duke llogaritur anëtarët a_i të tij kështu:

$$a_i = \frac{x}{3} + 3\sum_{j=1}^{i}(j + 2i)$$

nëse janë dhënë vlerat e variablave `x` dhe `n`.

a. Bllok-diagrami

b. Rruga - për `x=1` dhe `n=3`

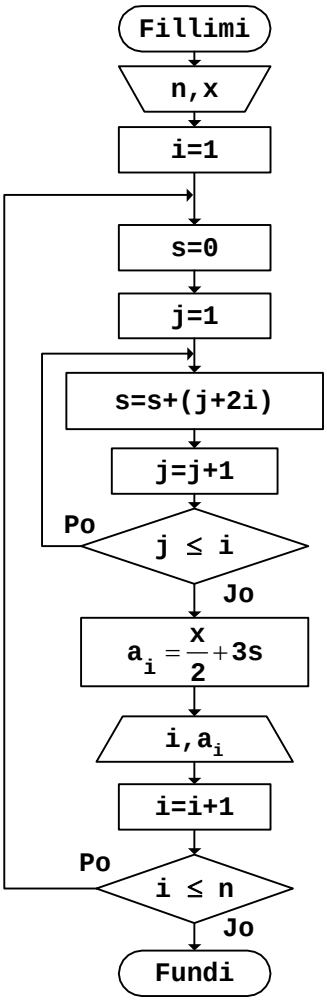


Fig.6.4

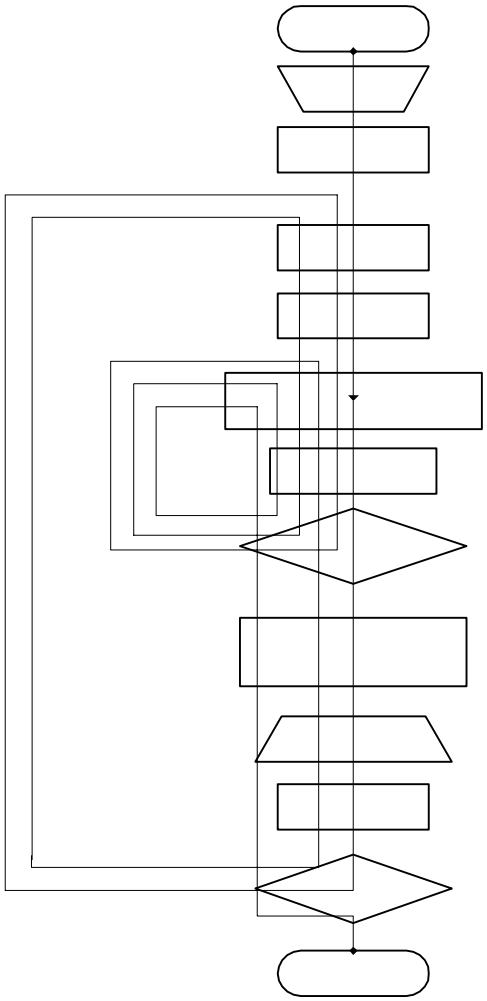


Fig.6.5

Vlerat e llogaritura gjatë rrugës së kaluar:

$$a_1 = \frac{1}{2} + 3 \cdot \{[1 + 2 \cdot 1]\}$$

$$a_2 = \frac{1}{2} + 3 \cdot \{[1 + 2 \cdot 2] + [2 + 2 \cdot 2]\}$$

$$a_3 = \frac{1}{2} + 3 \cdot \{[1 + 2 \cdot 3] + [2 + 2 \cdot 3] + [3 + 2 \cdot 3]\}$$

c. Programi

```
// Programi Prg6_4
#include <iostream>
using namespace std;
int main()
{
  int const n=3,
      x=1;
  int i,j;
  double s,A[n];
  for (i=0;i<n;i++)
  {
    s=0;
    for (j=1;j<=i;j++)
        s=s+(j+2*i);
    A[i]=x/2.+3*s;
    cout << "A["
         << i
         << "]="
         << A[i]
         << "\n";
  }
  return 0;
}
```

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

```
A[0]=0.5
A[1]=9.5
A[2]=33.5
```

Vektorët mund të formohen edhe duke shfrytëzuar anëtarët e vektorëve të tjerë.

Shembull Formimi i vektorit `B(n)`, duke shfrytëzuar vlerat e anëtarëve të vektorit të dhënë `A(n)`, në bazë të ligjshmërisë:

$$b_i = i + 2a_i^2 - 1$$

a. Bllok-diagrami

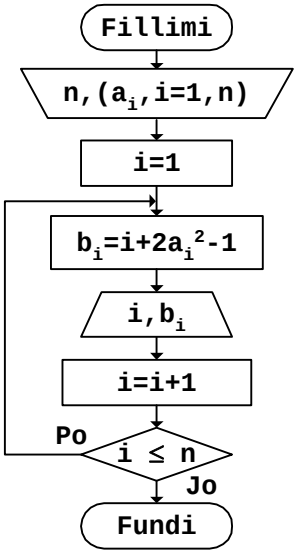


Fig.6.6

b. Rruga - për A=

3	5	-2	8

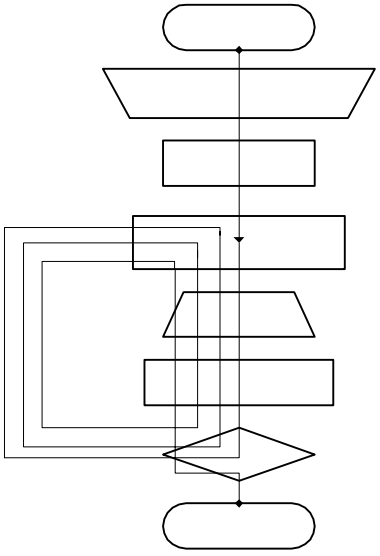


Fig.6.7

Në bllok-diagram, me $(a_i,i=1,n)$ shënohet shkurt vargu i anëtarëve të vektorit, përkatësisht $a_1, a_2, \ldots, a_n$.

c. Programi

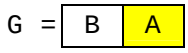
```
// Programi Prg6_6
#include <iostream>
using namespace std;
int main()
{
  int const n=4;
  int i,A[n]={3,5,-2,8},B[n];
  for (i=0;i<n;i++)
  {
    B[i]=i+2*(A[i]*A[i])-1;
    cout << "B["
         << i
         << "]="
         << B[i]
         << "\n";
  }
  return 0;
}
```

Pas ekzekutimit të programit, anëtarët e vektorit të formuar do të shtypen në ekran kështu:

```
B[0]=17
B[1]=50
B[2]=9
B[3]=130
```

Vektorët mund të formohen edhe duke shfrytëzuar vlerat numerike të anëtarëve të disa vektorëve njëkohësisht.

Shembull Formimi i vektorit `G(k)`, duke shfrytëzuar vlerat e anëtarëve të vektorëve `A(n)` dhe `B(m)`, kështu:



a. Bllok-diagrami

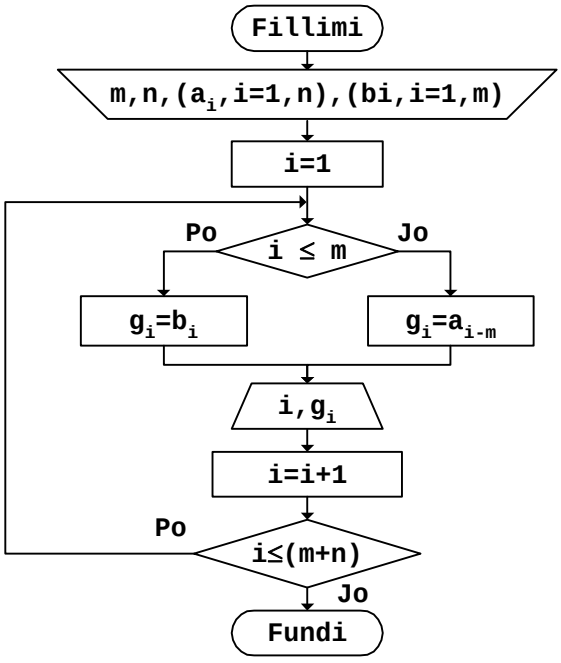


Fig.6.8

b. Rruga

A =	9	-3

B =	6	-2	8

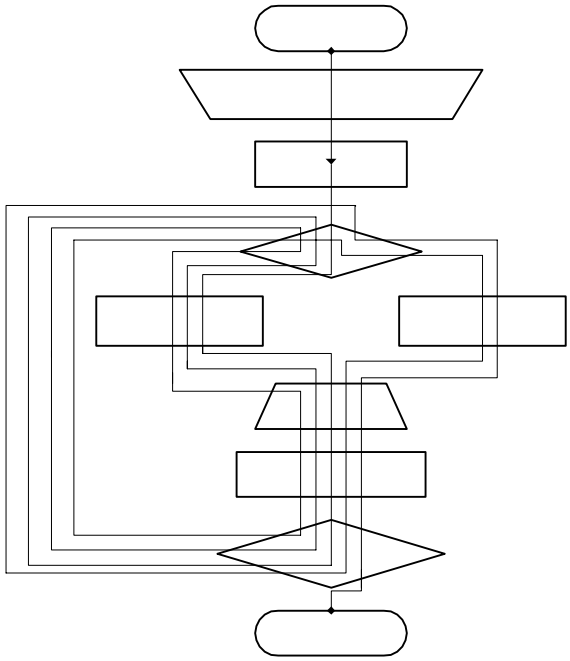


Fig.6.9

c. Programi

```
// Programi Prg6_8
#include <iostream>
using namespace std;
int main()
{
  int const m=3,n=2;
  int A[n]={9,-3},
      B[m]={6,-2,8},
      i,G[m+n];
  for (i=0;i<m+n;i++)
  {
    if (i<m)
       G[i]=B[i];
```

```
    else
       G[i]=A[i-m];
    cout << "G["
         << i
         << "]="
         << G[i]
         << "\n";
  }
  return 0;
}
```

Anëtarët e vektorit të formuar `G`, pas ekzekutimit të programit të dhënë, në ekran do të shtypen kështu:

```
G[0]=6
G[1]=-2
G[2]=8
G[3]=9
G[4]=-3
```

Operacionet aritmetikore

Në praktikë shpesh herë kryhen operacione të ndryshme aritmetikore mbi anëtarët e një vektori, si dhe mes anëtarëve të dy ose më shum vektorëve.

Shembull

Formimi i vektorit `B(n)`, nga anëtarët përkatës të vektorit të dhënë `A(n)`, duke shumëzuar anëtarëve negativë me vlerat e indekseve të tyre, kurse anëtarët pozitivë - me vlerën e konstantes së dhënë `x`.

a. Bllok-diagrami

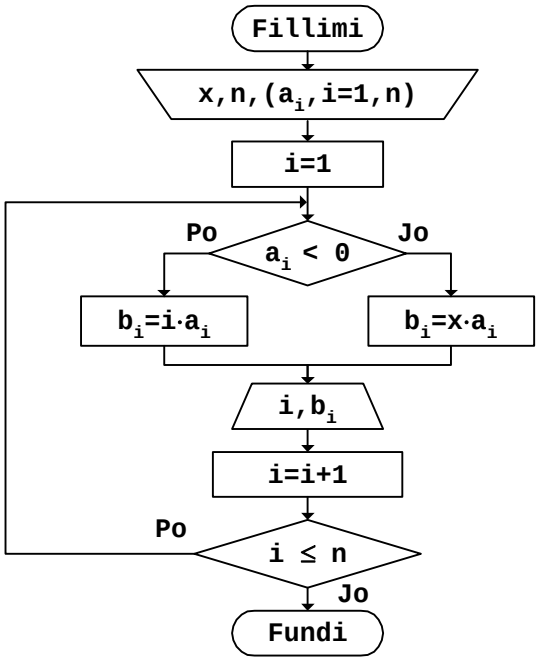


Fig.6.10

b. Rruga e kaluar - për x=2 dhe A = | 6 | 9 | -3 | 4 | -2 |

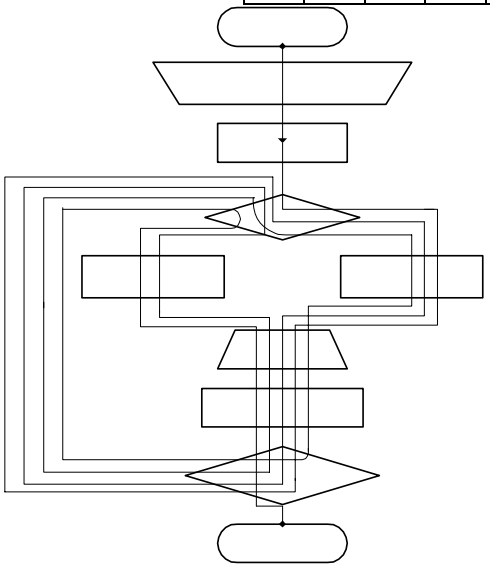


Fig.6.11

Vlerat që llogariten gjatë rrugës së kaluar janë:

$b_1 = x \cdot a_1$
$b_2 = x \cdot a_2$
$b_3 = 3 \cdot a_3$
$b_4 = x \cdot a_4$
$b_5 = 5 \cdot a_5$

c. Programi

```
// Programi Prg6_10
#include <iostream>
using namespace std;
int main()
{
  int const x=2,n=5;
  int A[n]={6,9,-3,4,-2},i,B[n];
  for (i=0;i<n;i++)
  {
    if (A[i]<0)
       B[i]=i*A[i];
    else
       B[i]=x*A[i];
    cout << "B["
         << i
         << "]="
         << B[i]
         << "\n";
  }
  return 0;
}
```

Nëse programi i dhënë ekzekutohet, vlerat e vektorit `B` të cilat shtypen në ekran, janë:

```
B[0]=12
B[1]=18
B[2]=-6
B[3]=8
B[4]=-8
```

Vlerat e anëtarëve të vektorit mund të përcaktohen edhe përmes llogaritjeve të ndryshme më komplekse.

Shembull Formimi i vektorit `Z(m)`, nga anëtarët e vektorit të dhënë

`F(m)`, duke ua shtuar anëtarëve të veçantë vlerat e shumave:

$$z_i = \begin{cases} f_i + i & \text{për } i \geq 3 \\ \sum_{j=1}^{i} f_j^2 & \text{për } i < 3 \end{cases}$$

a. Bllok-diagrami

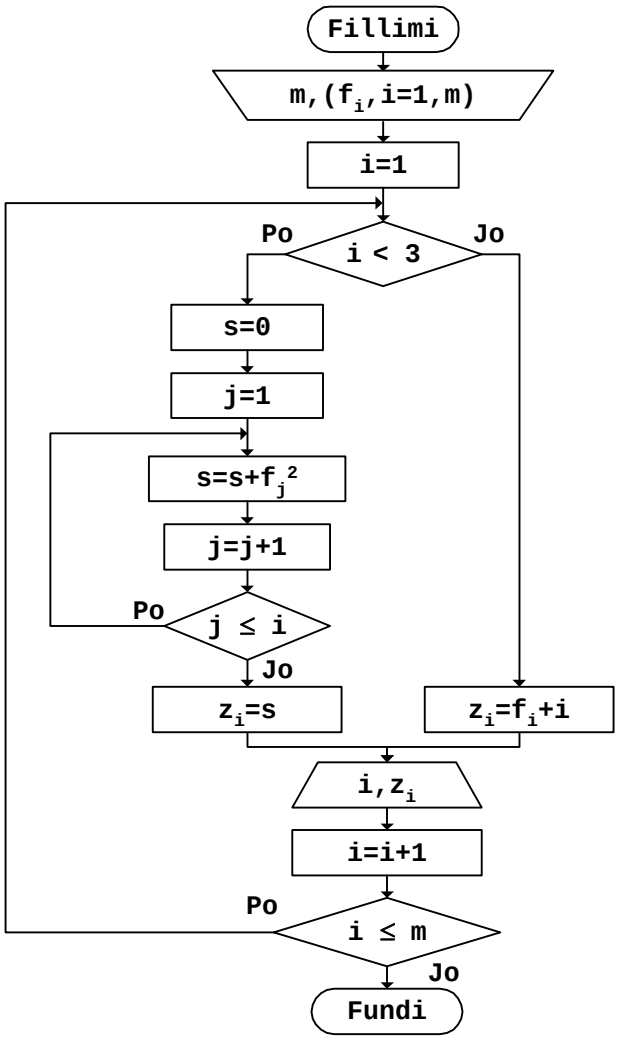


Fig.6.12

b. Rruga - për m=5 dhe F=

3	2	-1	6	-4

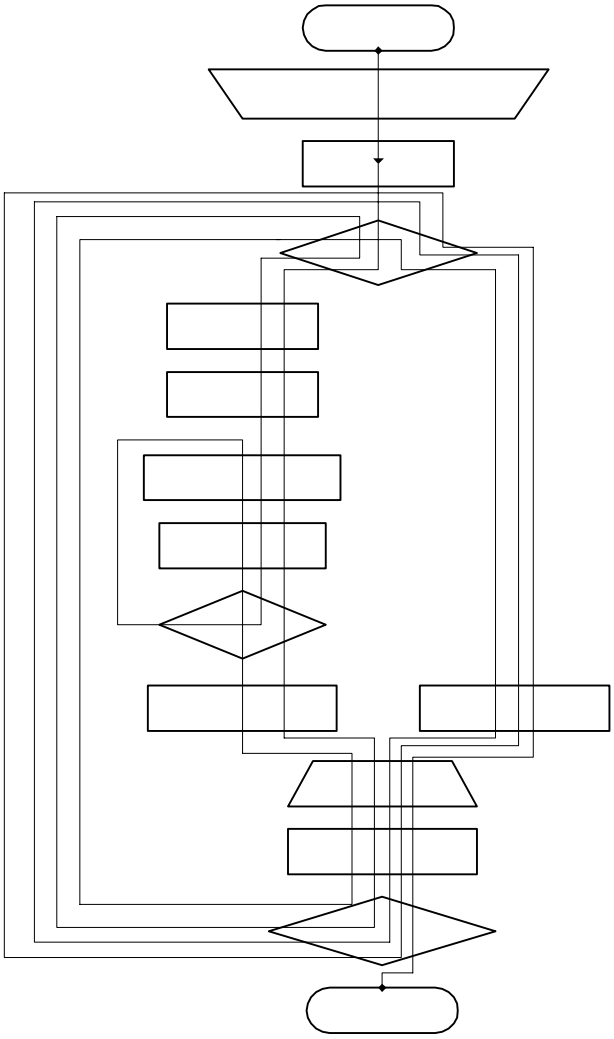


Fig.6.13

Gjatë rrugës së kaluar në *Fig.6.13* anëtarët e vektorit Z(n) llogariten kështu:

$z_1 = f_1^2$

$z_2 = f_1^2 + f_2^2$

$z_3 = f_3 + 3$

$z_4 = f_4 + 4$

$z_5 = f_5 + 5$

c. Programi

```
// Programi Prg6_12
#include <iostream>
using namespace std;
int main()
{
  int const m=5;
  int F[m]={3,2,-1,6,-4},i,j,s,Z[m];
  for (i=0;i<m;i++)
  {
    if (i<2)
    {
       s=0;
       for (j=0;j<=i;j++)
            s=s+F[j]*F[j];
       Z[i]=s;
    }
    else
       Z[i]=F[i]+i;
    cout << "Z["
         << i
         << "]="
         << Z[i]
         << "\n";
  }
  return 0;
}
```

Vektori Z, që formohet pas ekzekutimit të programit, për vlerat e vektorit F të cilat janë shfrytëzuar gjatë rrugës së kaluar në *Fig.6.13*, është:

```
Z[1]=9
Z[2]=13
Z[3]=1
Z[4]=9
Z[5]=0
```

Operacionet aritmetikore mund të zbatohen edhe mbi anëtarët e më shumë vektorëve njëkohësisht.

Shembull Formimi i vektorit `G(m)`, duke shumëzuar katrorët e anëtarëve përkatës të vektorit `A(m)` me vlerat absolute të anëtarëve me indeks të njëjtë të vektorit `B(m)`. Njëkohësisht, anëtarët e vektorit `G(m)`, të cilët janë më të mëdhenj se numri i dhënë `x`, duhet të pjesëtohen me indeksin përkatës.

a. Bllok-diagrami

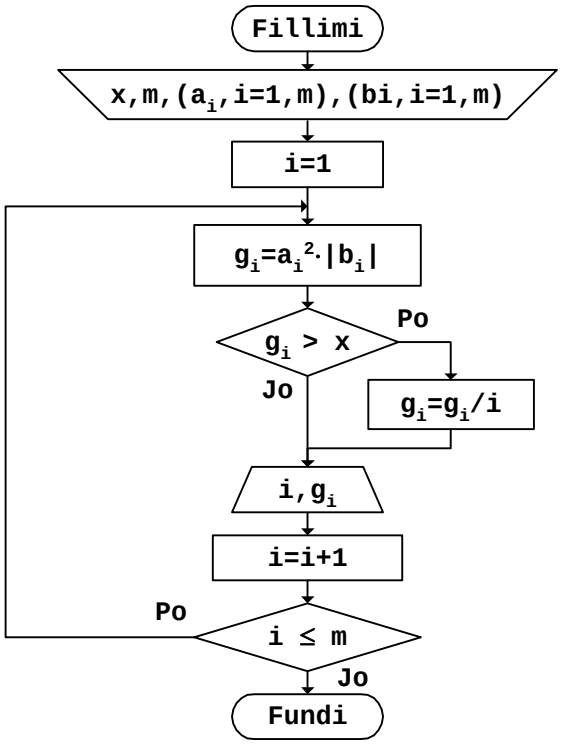


Fig.6.14

b. Rruga - për `x=20` dhe `A=`

-1	7	5	2

`B =`

2	-3	8	-4

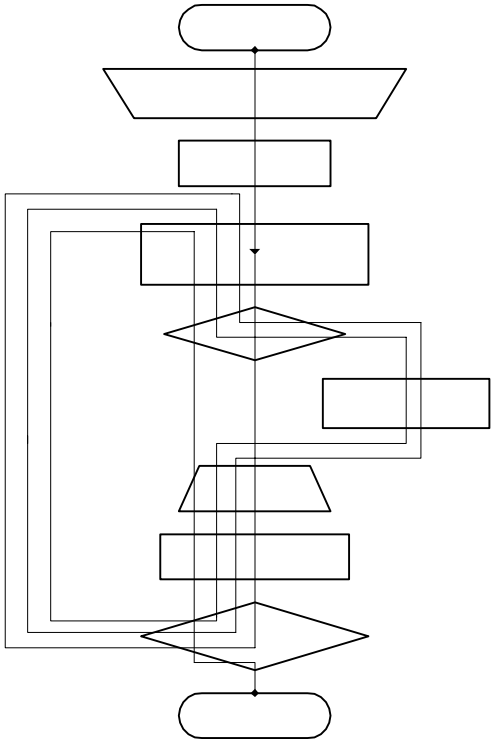


Fig.6.15

c. Programi

```
// Programi Prg6_14
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const x=20,m=4;
  int A[m]={-1,7,5,2},
      B[m]={2,-3,8,4},i;
  double G[m];
  for (i=0;i<m;i++)
```

```
  {
    G[i]=A[i]*A[i]*abs(B[i]);
    if (G[i]>x)
       G[i]=G[i]/i;
    cout << "G["
         << i
         << "]="
         << G[i]
         << "\n";
  }
  return 0;
}
```

Pas ekzekutimit të programit të dhënë për vlerat e shfrytëzuara gjatë vizatimit të rrugës në *Fig.6.15*, vlerat e anëtarëve të vektorit `G`, të cilat shtypen në ekran, janë:

```
G[0]=2
G[1]=147
G[2]=100
G[3]=16
```

Shuma dhe prodhimi

Gjatë llogaritjeve të ndryshme praktike, shpesh herë shfrytëzohet, p.sh., shuma, ose prodhimi, ose shuma e katrorëve, ose shuma e kubeve, përkatësisht shumë kombinime të tjera të mbledhjes, të zbritjes, të shumëzimit dhe të pjesëtimit të anëtarëve të vektorëve.

Shembull Mbledhja e anëtarëve pozitiv dhe e katrorëve të anëtarëve negativ të vektorit të dhënë `A(n)`.

a. Bllok-diagrami

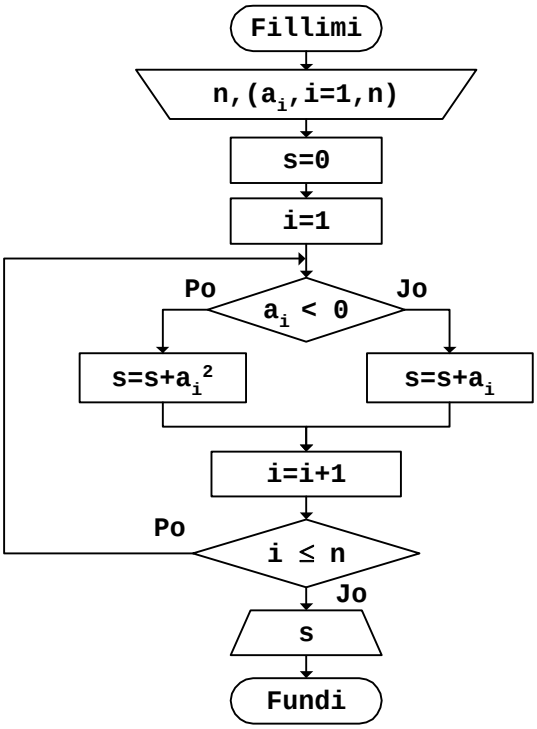


Fig.6.16

b. Rruga - për A =

7	-2	4	6	-3

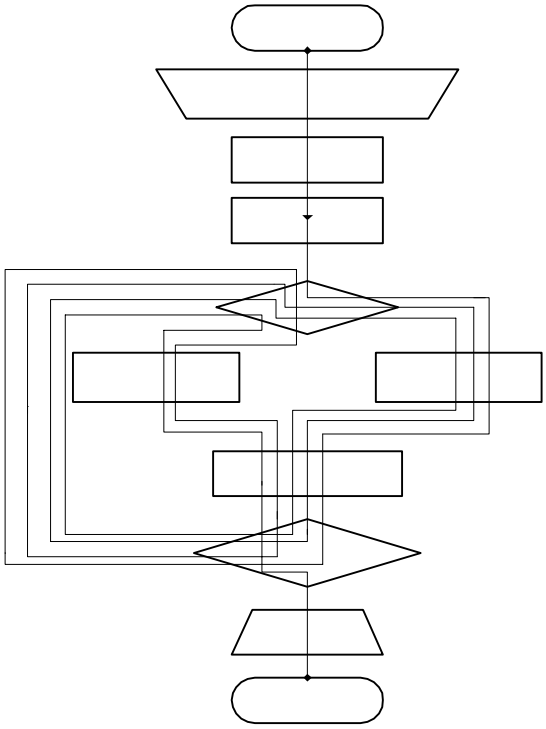


Fig.6.17

Shuma e cila llogaritet gjatë rrugës së kaluar është:

$$s = 7 + (-2)^2 + 4 + 6 + (-3)^2$$

c. Programi

```
// Programi Prg6_16
#include <iostream>
using namespace std;
int main()
{
  int const n=5;
  int A[n]={7,-2,4,6,-3},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];
  }
     cout << "Shuma e kërkuar s="
          << s
          << "\n";
  return 0;
}
```

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

```
Shuma e kërkuar s=30
```

Ngjashëm, vlerat e anëtarëve të vektorëve mund të shfrytëzohen për gjetjen e prodhimeve të ndryshme.

Shembull

Prodhimi i anëtarëve të vektorit `A(n)` të cilët kanë vlerë absolute më të madhe se `5` dhe më të vogël se `12`, duke shtypur anëtarët që nuk marrin pjesë në prodhim si dhe indekset e tyre.

b. Bllok-diagrami

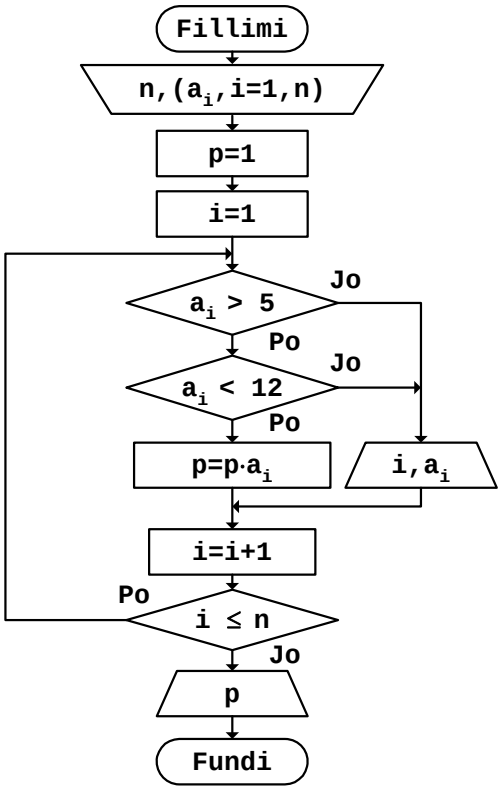


Fig.6.18

b. Rruga - për A =

7	19	8	-2	10

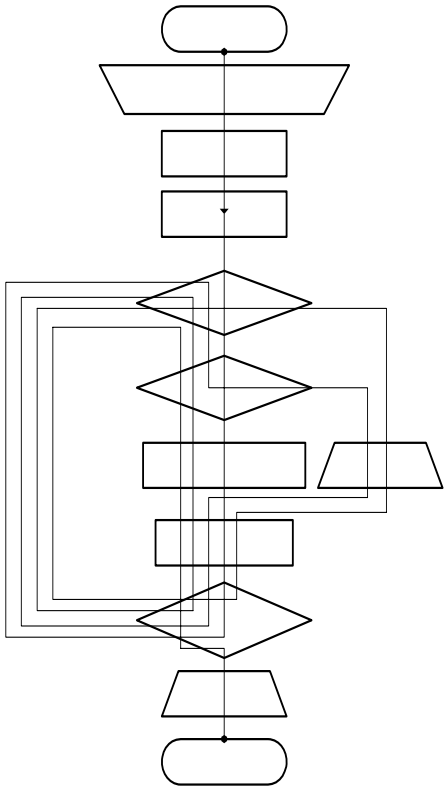


Fig.6.19

Vlera që llogaritet gjatë rrugës së kaluar është:

```
      ↓— Vlera fillestare
p = 1 · 7 · 8 · 10
```

Njëkohësisht, shtypen edhe indekset dhe vlerat e anëtarëve të cilët janë jashtë kufijve të dhënë:

```
2     19
4     -2
```

c. Programi

```
// Programi Prg6_18
#include <iostream>
using namespace std;
int main()
{
  int const n=5;
  char t[20]="--------------";
  int A[n]={7,19,8,-2,10},i,p;
  p=1;
  cout << "  i     A[i]"
       << "\n"
       << t
       << "\n";
  for (i=0;i<n;i++)
  {
    if ((A[i]>5) && (A[i]<12))
       p=p*A[i];
    else
       cout << "  "
            << i
            << "      "
            << A[i]
            << "\n";
  }
  cout << t
       << "\n"
       << "Prodhimi p="
       << p
       << "\n";
  return 0;
}
```

Për vlerat e shfrytëzuara gjatë rrugës së kaluar në *Fig.6.19*, si rezultat në ekran shtypet:

```
  i    A[i]
--------------
  1     19
  3     -2
--------------
Prodhimi p=560
```

Anëtarët e vektorëve mund të shfrytëzohen edhe brenda shprehjeve të ndryshme për llogaritjen e shumave ose të prodhimeve, përmes indekseve përkatëse.

Shembull Llogaritja e vlerës së shprehjes:

$$y = \begin{cases} 3x + 4\prod_{i=1}^{n}\left(|a_i| + 2\right) & \text{për } x < 0.55 \\ e^{2x} - 2\sum_{\substack{i=2 \\ (\text{çift})}}^{n}\left(a_i^2 - x\right) & \text{për } x \geq 0.55 \end{cases}$$

ku me a_i nënkuptohen anëtarët e vektorit të dhënë $A(n)$.

a. Bllok-diagrami

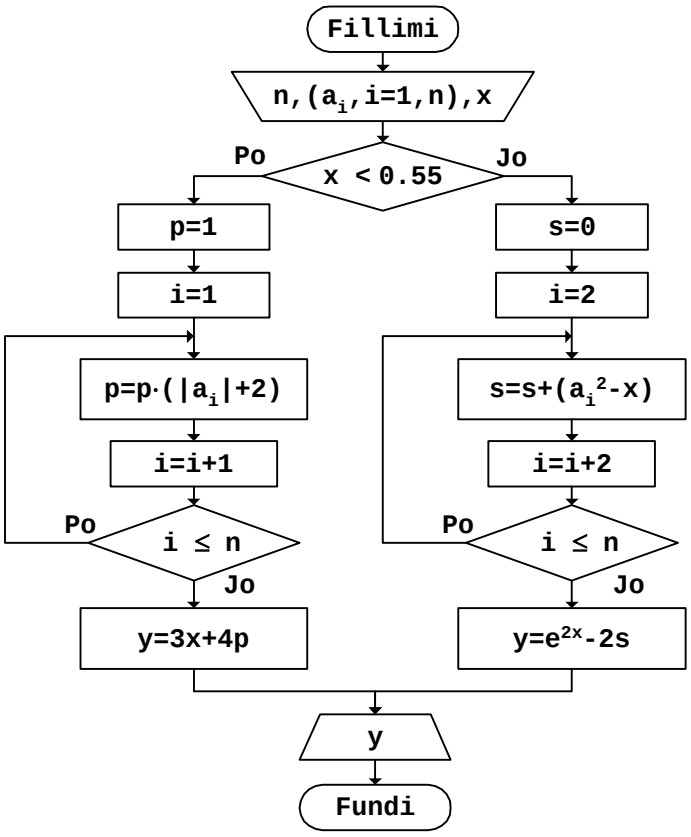


Fig.6.20

b. Rruga - për x = 0.2 dhe A= | 6 | 4 | -9 |

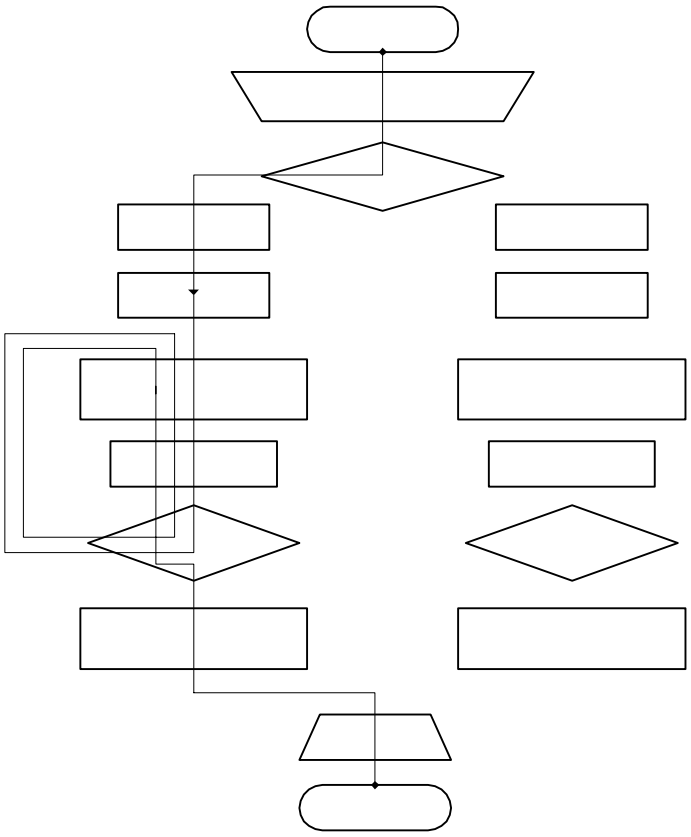


Fig.6.21

c. Programi

```
// Programi Prg6_20
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const n=3;
  int A[n]={6,4,-9},i;
  double s,p,x,y;
  cout << "Vlera e variablës x=";
  cin >> x;
```

```
  if (x<0.55)
  {
     p=1;
     for (i=0;i<n;i++)
         p=p*(abs(A[i])+2);
     y=3*x+4*p;
  }
  else
  {
     s=0;
     i=2;
     do
     {
       s=s+A[i]*A[i]-x);
       i=i+2;
    }
    while (i<=n);
    y=exp(2*x)-2*s;
  }
  cout << "Vlera e llogaritur y="
       << y
       << "\n";
  return 0;
}
```

Rezultati që shtypet në ekran, për vlerat numerike që janë shfrytëzuar gjatë vizatimit të rrugës së kaluar në *Fig.6.21*, është:

```
Vlera e llogaritur y=2112.6
```

Numërimi i anëtarëve të caktuar

Për t'i numëruar anëtarët e caktuar brenda vektorit, të gjithë anëtarët e tij duhet të krahasohen me kushtet e përcaktuara për numërim.

Shembull Numri `n` i anëtarëve me vlerë numerike negative, në vektorin e dhënë `A(m)`.

Këtu, vlera fillestare e numëratorit `n` duhet të merret zero. Pastaj, sa herë që gjendet ndonjë anëtar me vlerë numerike negative, vlera e numëratorit `n` rritet për një, kurse për anëtarët e tjerë kjo madhësi nuk ndryshohet.

a. Bllok-diagrami

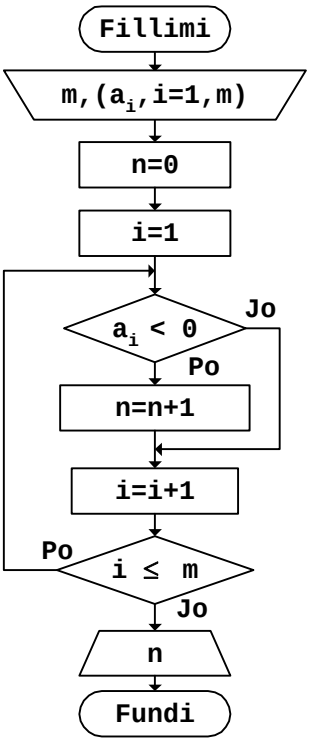


Fig.6.22

b. Rruga - për A= 3 | -2 | 7 | -4 | 5

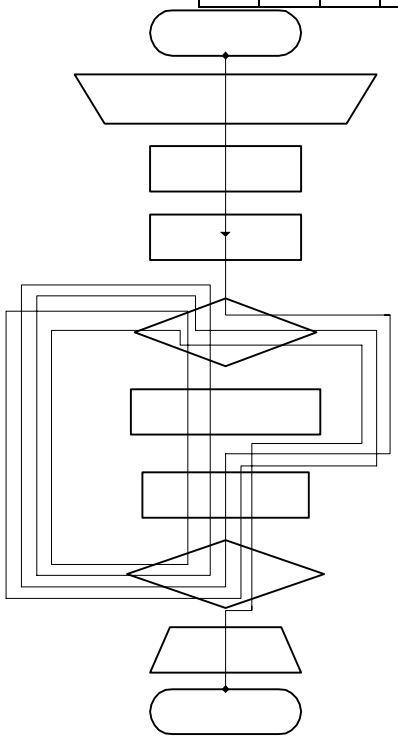


Fig.6.23

c. Programi

```
// Programi Prg6_22
#include <iostream>
using namespace std;
int main()
{
  int const m=5;
  int A[m]={3,-2,7,-4,5},i,n;
  n=0;
  for (i=0;i<m;i++)
      if (A[i]<0)
         n=n+1;
  cout << "Anëtarë negativë n="
       << n
       << "\n";
  return 0;
}
```

Rezultati që shtypet në ekran, për vlerat e shfrytëzuara gjatë vizatimit të bllok-diagramit në *Fig.6.23*, pas ekzekutimit të programit të dhënë, është:

```
Anëtarë negativë n=2
```

gjë që shihet edhe nga rruga e vizatuar, kur nëpër bllokun në të cilin rritet numëratori `n` kalohet vetëm dy herë.

Numërimi mund të bëhet edhe duke shtruar më shumë kushte njëkohësisht.

Shembull Numërimi `k` i anëtarëve negativë të vektorit `F(m)`, por të cilët për nga vlera absolute janë më të mëdhenj se numri pozitiv `x`.

a. Bllok-diagrami

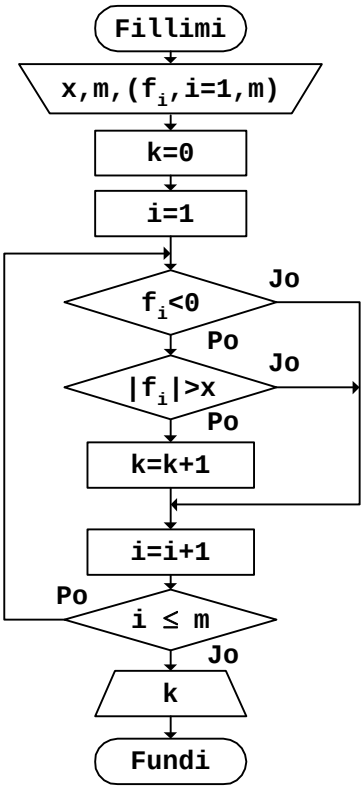


Fig.6.24

b. Rruga - për `x=3` dhe `A=`

-7	4	-5	-1	6

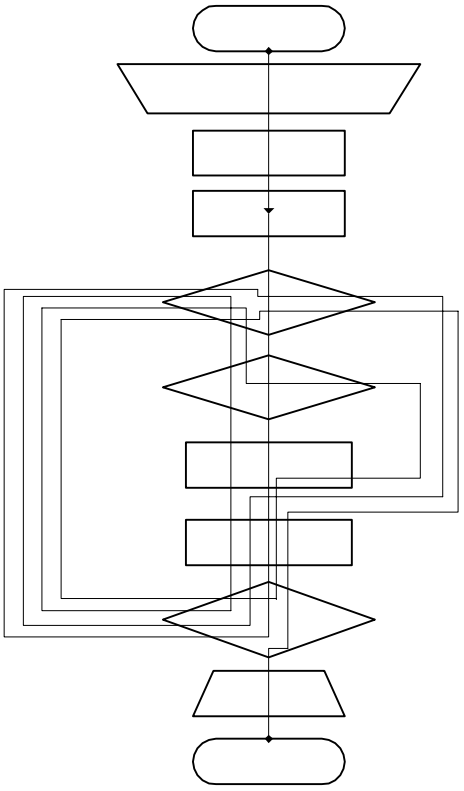


Fig.6.25

c. Programi

```
// Programi Prg6_24
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const m=5,x=3;
  int F[m]={-7,4,-5,-1,6},i,k;
  k=0;
  for (i=0;i<m;i++)
      if ((F[i]<0) && (abs(F[i])>x))
         k=k+1;
  cout << "Numri i kërkuar k="
       << k
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë, për vlerat hyrëse të shfrytëzuara gjatë vizatimit të rrugës së kaluar në *Fig.6.25*, si rezultat në ekran shtypet:

```
Numri i kërkuar k=2
```

meqë vetëm dy vlera numerike i plotësojnë dy kushtet e shtruara (`-7` dhe `-5`).

Bllok-diagrami mund të përpilohet edhe ashtu që brenda tij njëkohësisht të bëhen disa numërime.

Shembull Numri `p` i anëtarëve pozitivë dhe numri `n` i anëtarëve negativë brenda vektorit `A(m)`.

a. Bllok-diagrami

b. Rruga - për `A=`

2	-3	-7	4	1

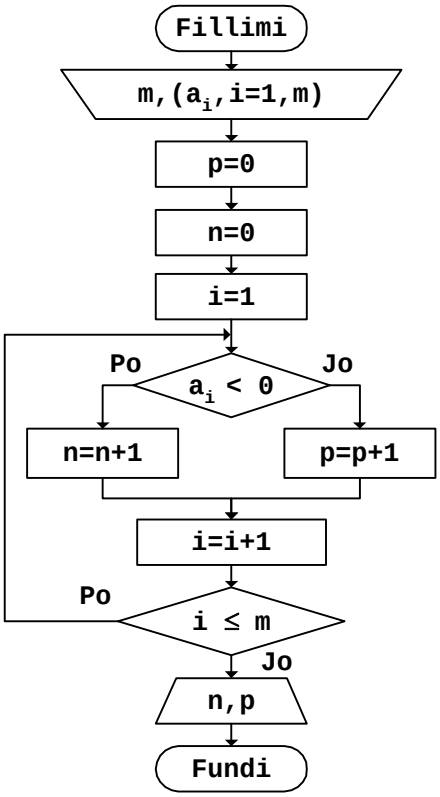


Fig.6.26

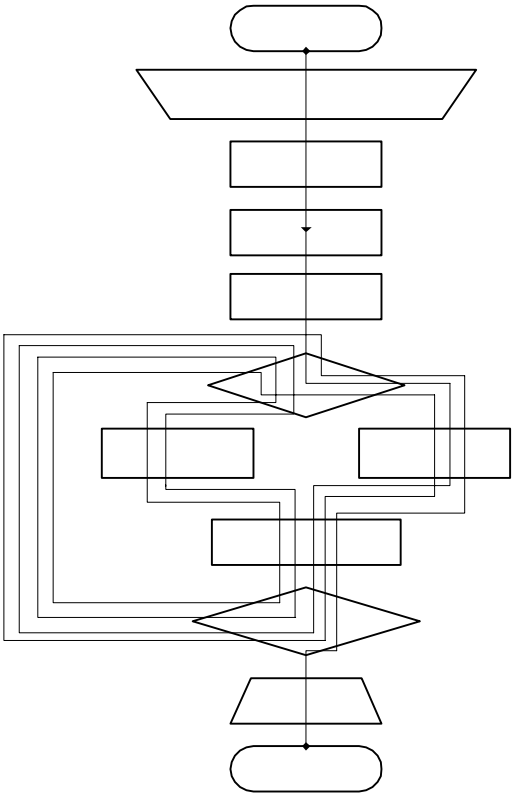


Fig.6.27

c. Programi

```
// Programi Prg6_26
#include <iostream>
using namespace std;
int main()
{
   int const m=5;int A[m]={2,-3,-7,4,1},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 << "Anëtarë pozitiv p="
        << p
        << "\n";
   cout << "Anëtarë negativ n="
        << n
        << "\n";
   return 0;
```

```
}
```

Për vlerat e shfrytëzuara gjatë vizatimit të rrugës së kaluar në *Fig.6.27*, pas ekzekutimit të programit, si rezultat shtypet:

```
Anëtarë pozitivë p=3
Anëtarë negativë n=2
```

Gjatë procedurës së numërimit të anëtarëve të vektorit mund të kërkohet që njëkohësisht të shtypen anëtarët që i plotësojnë, ose anëtarët që nuk i plotësojnë kushtet e numërimit.

Shembull Numëri `k` i anëtarëve të vektorit `A(n)` me vlera absolute mes vlerave `3` dhe `8`, duke shtypur njëkohësisht anëtarët me vlera jashtë këtij diapazoni, si dhe indekset përkatëse.

a. Bllok-diagrami

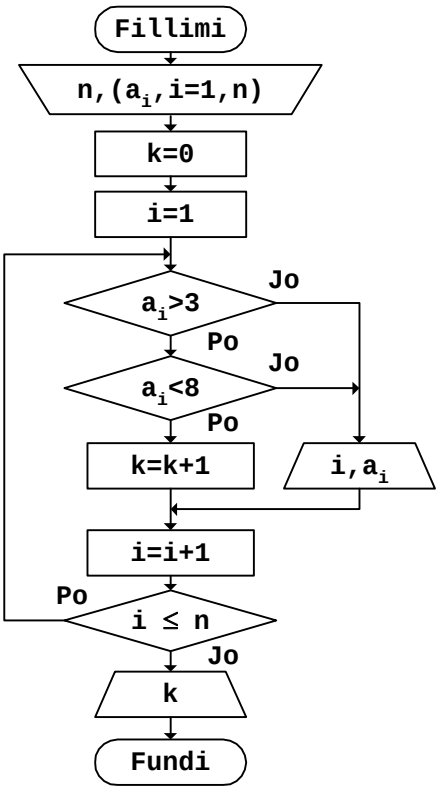


Fig.6.28

b. Rruga - për A=

5	2	4	9	-7

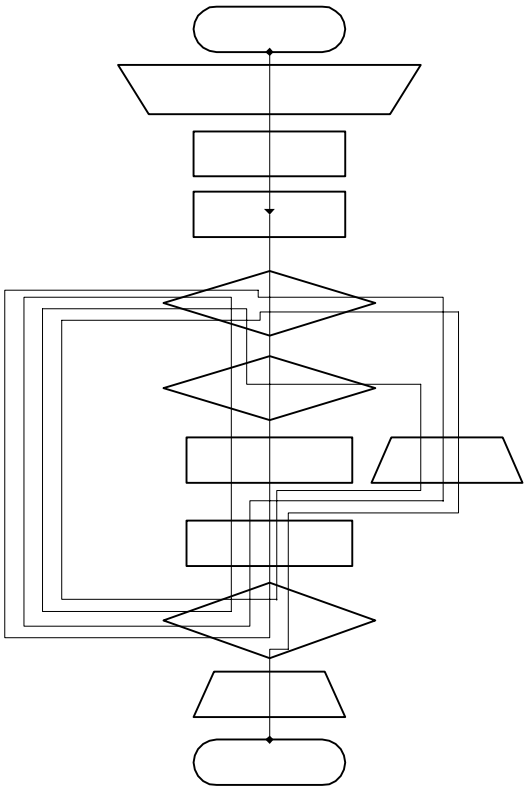


Fig.6.29

c. Programi

```
// Programi Prg6_28
```

```
#include <iostream>
using namespace std;
int main()
{
  int const n=5;int A[n]={5,2,4,9,-7},i,k;
  k=0;
  for (i=0;i<n;i++)
  {
      if ((A[i]>3) && (A[i]<8))
         k=k+1;
      else
         cout << "i="
              << i
              << "  A["
              << i
              << "]="
              << A[i]
              << "\n";
  }
  cout << "Numri i kërkuar k="
       << k
       << "\n";
  return 0;
}
```

Nëse ekzekutohet programi i dhënë për vlerat hyrëse të shfrytëzuara gjatë vizatimit të rrugës së kaluar në *Fig.6.29*, në ekran do të shtypen së pari indekset dhe vlerat e anëtarëve të cilët nuk i plotësojnë kushtet:

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

dhe pastaj edhe numri i anëtarëve që i plotësojnë kushtet:

```
Numri i kërkuar k=2
```

Detyra

Të gjendet:

a. sa anëtarë të vektorit të dhënë `Y(n)`, përnga vlera absolute janë numra më të mëdhenj se numri pozitiv `x`;

b. numri i anëtarëve në vektorin e dhënë `T(m)`, të cilët janë më të mëdhenj se `x` e më të vegjël se `y`, ku `x>y`.

Gjetja e anëtarëve të caktuar

Shpeshherë, gjatë zgjidhjes së problemeve të ndryshme me kompjuter, nevojiten vlera të caktuara brenda fushave numerike, siç është, p.sh., anëtari me vlerë numerike minimale, ose anëtari me vlerë numerike maksimale. Procesi i gjetjes së tyre fillon me përvetësimin e vlerës fillestare të një variable ndihmëse, tek e cila do të ruhet vlera e kërkuar. Si vlerë fillestare e variablës ndihmëse merret kryesisht anëtari i parë në vektor, ose vlera e saj formohet nga ky anëtar i vektorit. Pastaj, krahasohen me vlerën e variablës ndihmëse të gjithë anëtarët e tjerë të vektorit. Sa herë që gjatë krahasimit vlera e anëtarit të vektorit, e cila krahasohet, është më e afërt me vlerën e kërkuar, ajo vlerë ruhet te variabla ndihmëse. Në fund, pasi të krahasohen të gjithë anëtarët e vektorit, variabla ndihmëse e përmban vlerën e kërkuar.

Shembull Gjetja e vlerës maksimale `x` në vektorin e dhënë `A(n)`.

a. Bllok-diagrami

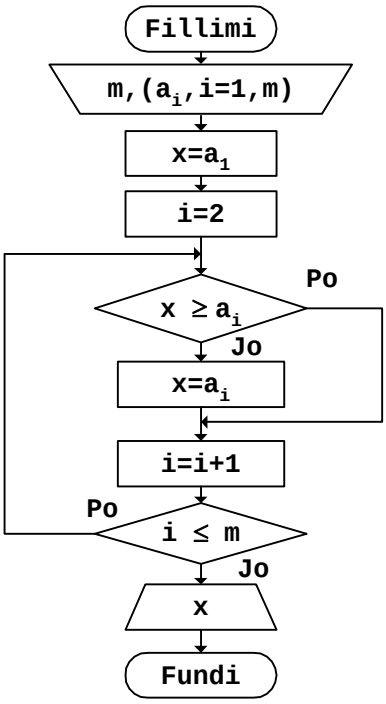


Fig.6.30

b. Rruga - për `A=`

3	7	5	9

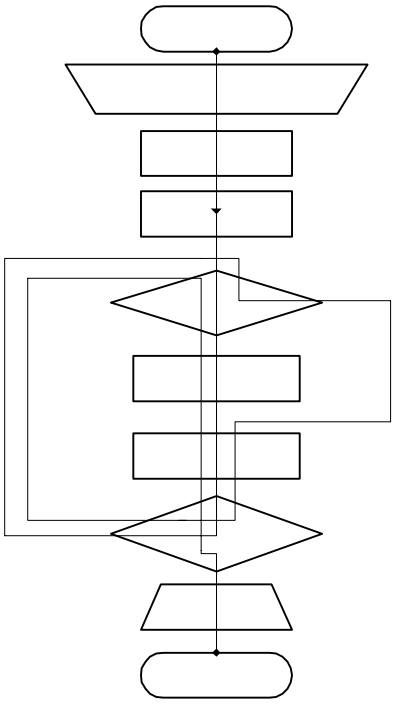


Fig.6.31

c. Testimi – për A=

3	7	5	9

Hapi	Blloku	Urdhëri	Vlerat numerike merren	Rezultati
1	***1***	Fillimi	**-**	Fillimi i algoritmit
2	***2***	Lexo: m (a_i,i=1,m)	prej njësisë hyrëse	m=4, a_1=3, a_2=7, a_3=7, a_4=9
3	***3***	x=a_1	a_1→**2**	x=3
4	***4***	i=2	**-**	i=2
5	***5***	Pyet: x≥a_i	x→**3**, i→**4**, a_i→**2**	Jo
6	***6***	x=a_i	i→**4**, a_i→**2**	X=7
7	***7***	i=i+1	i→**4**,	i=2+1=3
8	***8***	Pyet:i≤m	i→**7**, m→**2**	Po
9	***5***	Pyet: x≥a_i	x→**6**, i→**7**, a_i→**2**	Po
10	***7***	i=i+1	i→**7**,	i=3+1=4
11	***8***	Pyet:i≤m	i→**10**, m→**2**	Po
12	***5***	Pyet: x≥a_i	x→**6**, i→**10**, a_i→**2**	Jo
13	***6***	x=a_i	i→**10**, a_i→**2**	x=9
14	***7***	i=i+1	i→**10**,	i=4+1=5
15	***8***	Pyet:i≤m	i→**14**, m→**2**	Jo
16	***9***	Shtyp:x	x→**13**	Shtypet numri 9
17	***10***	Fundi	**-**	Fundi i algoritmit

Fig.6.32

d. Programi

```
// Programi Prg6_30
#include <iostream>
using namespace std;
int main()
{
  int const m=4;
  int A[m]={3,7,5,9},i,x;
  x=A[0];
  for (i=1;i<m;i++)
  {
    if (x>=A[i])
    {
    }
    else
```

```
      x=A[i];
  }
  cout << "Numri më i madh x="
       << x
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit, për vlerat hyrëse të shfrytëzuara për testimin e bllok-diagramit, rezultati që shtypet në ekran është:

```
Numri më i madh x=9
```

Gjatë gjetjes së vlerës së caktuar, si vlerë fillestare mund të merret edhe anëtari i fundit në vektor.

Shembull Gjetja e anëtarit me vlerë numerike më të vogël `v` për nga vlera absolute, në vektorin e dhënë `G(n)`.

a. Bllok-diagrami *b. Rruga* - për `A=`

-7	3	-6	4	-8

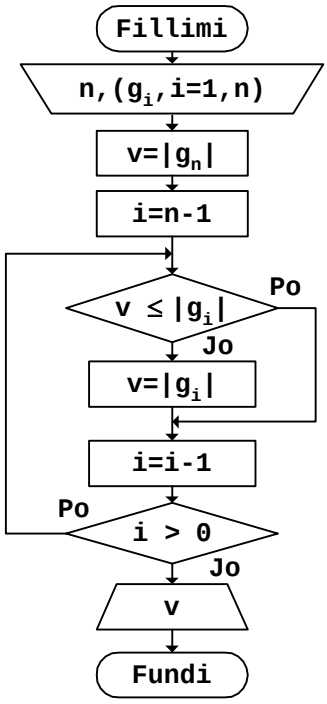


Fig.6.33

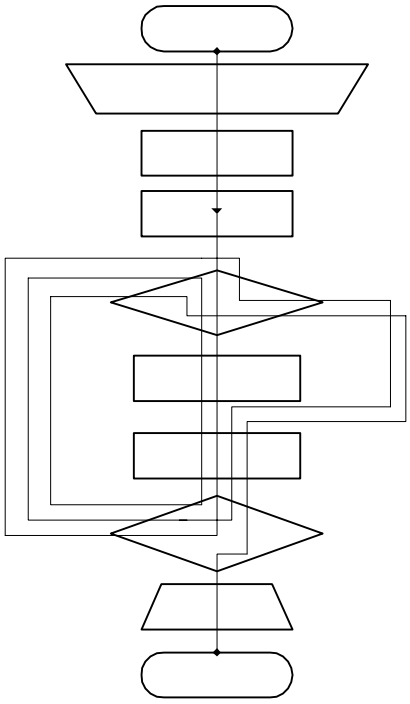


Fig.6.34

c. Programi

```
// Programi Prg6_33
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const n=5;
  int G[n]={-7,3,-6,4,-8},i,v;
  v=abs(G[n-1]);
  i=n-2;
  do
  {
    if (v<=abs(G[i]))
    {
    }
    else
       v=abs(G[i]);
    i--;
  }
  while (i>0);
  cout << "Vlera më e vogël v="
       << v
       << "\n";
  return 0;
}
```

Për vlerat numerike të shfrytëzuara gjatë vizatimit të rrugës së kaluar në bllok-diagramin e dhënë në *Fig.6.34*, si rezultat shtypet:

```
Vlera më e vogël v=3
```

Si vlerë fillestare gjatë gjetjes së anëtarit të caktuar në vektor mund të merret edhe vlera e cilitdo anëtar brenda vektorit. Por, gjatë kësaj duhet gjetur ligjshmërinë se si të bëhet krahasimi i të gjitha vlerave të vektorit me vlerën e variablës ndihmëse.

Përveç vlerës së anëtarit të caktuar, mund të kërkohet edhe pozita e tij në vektor, përkatësisht indeksi i tij.

Shembull Gjetja e anëtarit me vlerë numerike më të madhe `t` për nga vlera absolute, si dhe pozita përkatëse `k` në vektorin e dhënë `Z(n)`.

a. Bllok-diagrami *b. Rruga* - për A=

4	-7	2	9	-5

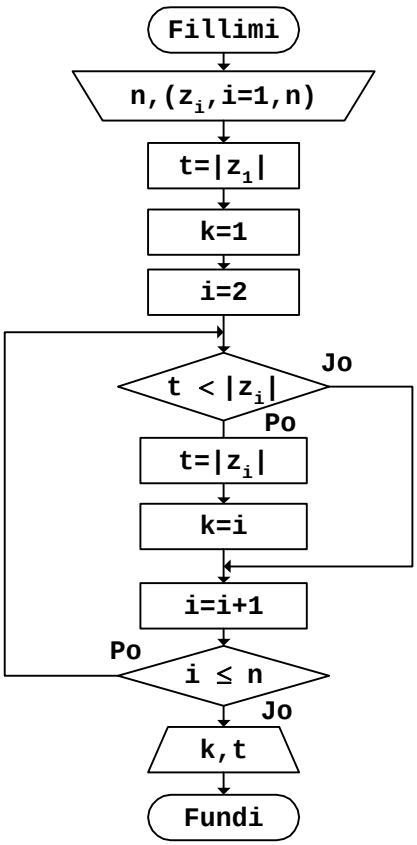


Fig.6.35

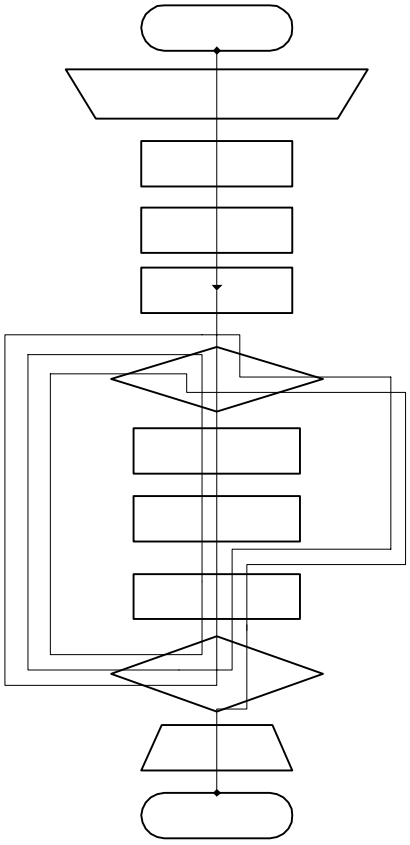


Fig.6.36

c. Programi

```
// Programi Prg6_35
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const n=5;
  int Z[n]={4,-7,2,9,-5},i,t,k;
  t=abs(Z[0]);
  k=0;
  for (i=1;i<n-1;i++)
```

```
  {
    if (t<abs(Z[i]))
    {
       t=abs(Z[i]);
       k=i;
    }
  }
  cout << "Vlera më e madhe ... t="
       << t
       << "\n";
  cout << "Indeksi ............ k="
       << k
       << "\n";
  return 0;
}
```

Nëse programi i dhënë ekzekutohet për vlerat e anëtarëve të vektorit `Z` të shfrytëzuara gjatë vizatimit të rrugës së kaluar në *Fig.6.36*, rezultati që shtypet në ekran është:

```
Vlera më e madhe ... t=9
Indeksi ............ k=3
```

Brenda një vektori mund të gjendet edhe anëtari i caktuar nga grumbulli i vlerave numerike të anëtarëve të cilët plotësojn kushte të fiksuara paraprakisht. Kështu, p.sh., mund të gjendet anëtari më i madh (më i vogël) në grumbullin e anëtarëve pozitivë (negativë) të vektorit, duke i marrë si vlera reale, ose si vlera absolute etj.

Shembull Gjetja e anëtarit me vlerë numerike më të madhe `d` për nga vlera absolute, në grumbullin e anëtarëve negativë të vektorit `A(n)`.

a. Bllok-diagrami

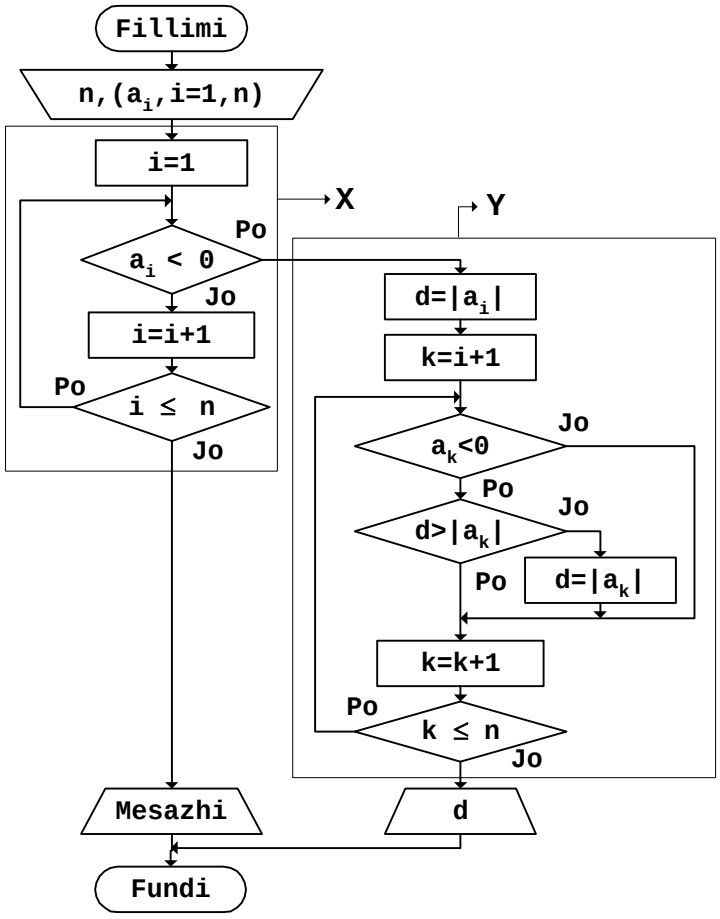


Fig.6.37

Pjesa `X` e bllok-diagramit të dhënë në *Fig.6.37* shfrytëzohet për përcaktimin e vlerës fillestare të variablës `d`, në të cilën si rezultat do ruhet vlera e kërkuar e vektorit. Në këtë pjesë, variablës `d` i shoqërohet vlera e anëtarit negativ, i cili gjendet në procesin e kërkimit, që fillon prej anëtarit të parë të vektorit.

Në pjesën `Y`, pasi paraprakisht është përcaktuar vlera fillestare e variablës `d`, procedura e krahasimit dhe e gjetjes së vlerës absolute më të madhe në grumbullin e anëtarëve negativë fillon prej anëtarit `k=i+1`, ku `i` është indeksi i anëtarit të parë negativ në vektor, i cili fitohet në pjesën `X` të bllok-diagramit.

b. Rruga - për A =

3	9	-2	6	-7	5

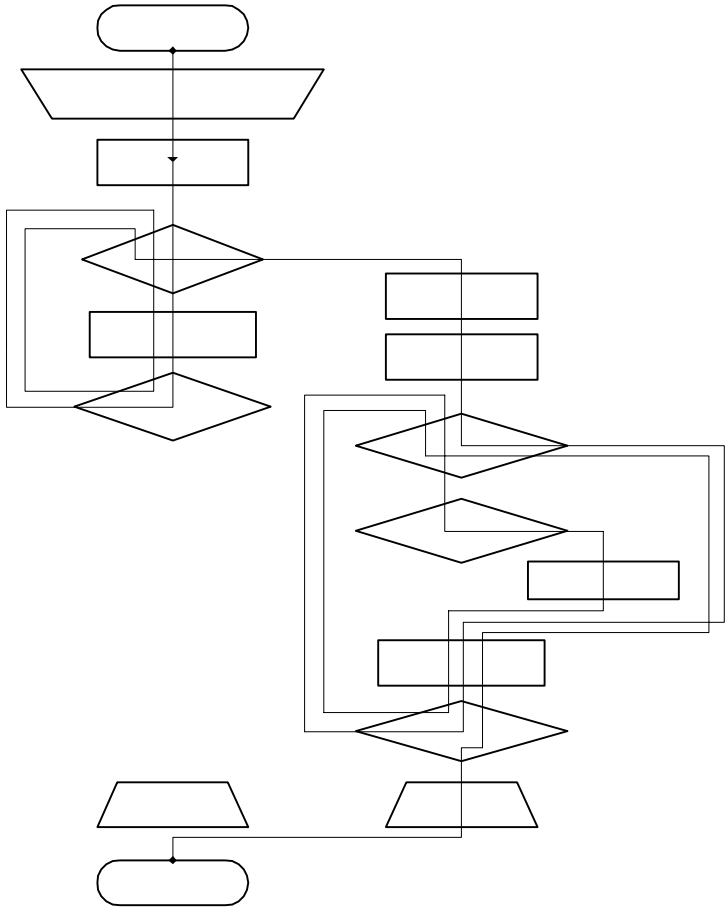


Fig.6.38

c. Programi

```
// Programi Prg6_37
#include <iostream>
#nclude <cmath>
using namespace std;
int main()
{
  int const n=6;
  int A[n]={3,-9,2,-6,7,-15},i,d,k;
  i=0;
```

```
  do
  {
    if (A[i]<0)
       goto Y;
    i++;
  }
  while (i<n);
  cout << "Nuk ka asnjë vlerë negative"
       << "\n";
  goto F;
Y:
  d=abs(A[i]);
  for (k=i+1;k<n;k++)
  {
      if (A[k]<0)
         if (d>abs(A[k]))
         {
         }
         else
            d=abs(A[k]);
  }
  cout << "Vlera e gjetur d="
       << d
       << "\n";
F:
  return 0;
}
```

Programi i dhënë është shkruar ashtu që për nga struktura t'i ngjajë bllok-diagramit. Nëse gjatë kërkimit në pjesën `X` nuk gjendet asnjë anëtar negativ, është paraparë që si rezultat të shtypet mesazhi përkatës:

```
Nuk ka asnjë vlerë negative
```

Për vlerat e marra gjatë vizatimit të rrugës së kaluar në *Fig.6.38* rezultati që shtypet në ekran është:

```
Vlera e gjetur d=7
```

sepse, prej dy vlerave negative (`-2` dhe `-7`), numri i dytë është më i madh si vlerë absolute.

Detyra Në vektorin e dhënë `F(n)`, të gjendet:

a. anëtari me vlerë numerike më të vogël `v`, si dhe indeksi përkatës `k`;

b. anëtari më i madh, në grupin e vlerave numerike, të cilat sillen mes vlerave `x` dhe `y`, nëse `x<y`.

Radhitja e anëtarëve

Anëtarët e vektorit të dhënë `A(n)` mund të radhiten sipas madhësisë, në `(n-1)`-hapa, kështu:

```
Hapi i parë   (i=1)       a1 krahasohet me a2,a3,...,an
Hapi i dytë   (i=2)       a2 krahasohet me a3,a4,...,an
 ..............           ...........
Hapi i fundit (i=n-1)     an-1 krahasohet me an.
```

Në çdo hap të ndërmarrë fiksohet vlera numerike e një anëtari të vektorit, kurse ngelin të pandryshuara vlerat numerike të anëtarëve që janë fiksuar në hapat paraprakë. Gjatë krahasimit të dy anëtarëve, për ndërrimin e vendeve të tyre shfrytëzohet një variabël ndihmëse, duke pasur parasysh radhën e cila është treguar në *Fig.6.39*, ku `b` është variabla ndihmëse, kurse a_i dhe a_j janë anëtarët, vlerat e të cilëve i ndërrojnë vendet mes vete.

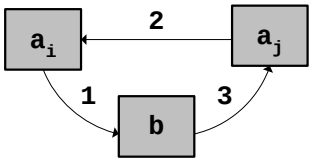


Fig.6.39

Shembull Radhitja sipas madhësisë e vlerave numerike të anëtarëve të vektorit `A(n)`, prej vlerës më të vogël.

a. Bllok-diagrami

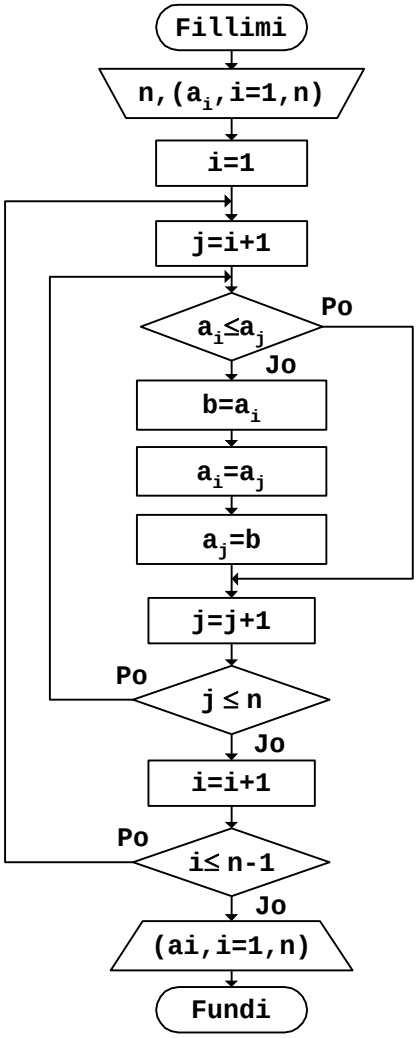


Fig.6.40

b. Rruga - për A=

7	2	8	3

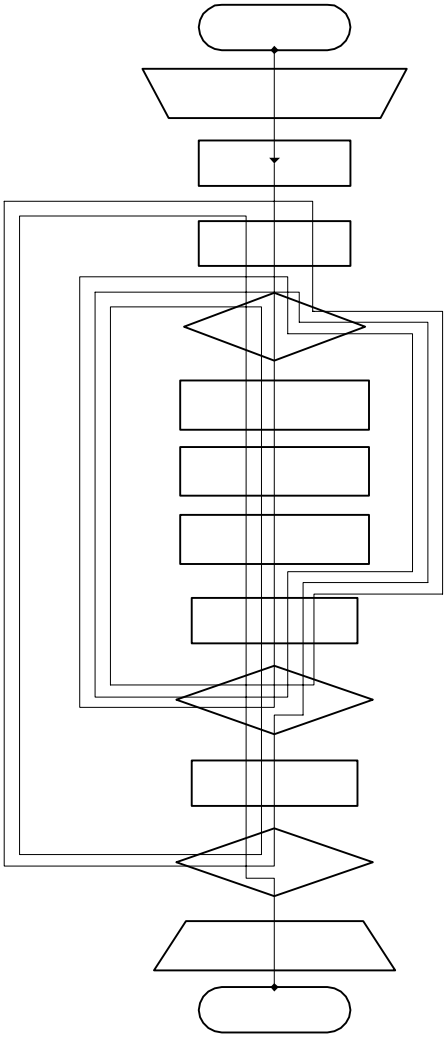


Fig.6.41

c. Programi

```
// Programi Prg6_40
#include <iostream>
using namespace std;
int main()
{
  int const n=4;
```

```
  int A[n]={7,2,8,3},i,j,b;
  for (i=0;i<n-1;i++)
    for (j=i+1;j<n;j++)
    {
        if (A[i]<=A[j])
        {
        }
        else
        {
           b=A[i];
           A[i]=A[j];
           A[j]=b;
        }
    }
  cout << "A=[  ";
  for (i=0;i<n;i++)
  cout << A[i]
       << "  ";
  cout << "]"
       << "\n";
  return 0;
}
```

Këtu, meqë si shembull është marrë vektori me `n=4` anëtarë, radhitja do të kryhet në `3`-hapa. Vlerat e anëtarëve të vektorit në fund të çdo hapi janë:

```
Hapi i parë  (i=0):   2  7  8  3
Hapi i dytë  (i=1):   2  3  8  7
Hapi i tretë (i=2):   2  3  7  8
```

Vlerat e fiksuara në fund të hapit të tretë janë vlerat e radhitura të vektorit sipas madhësive, prej më të voglit kah më i madhi, dhe rezultati që shtypet pas ekzekutimit të programit është:

```
A=[  2  3  7  8  ]
```

Bllok-diagrami i dhënë në *Fig.6.40* gjegjësisht programi përkatës, vlen edhe nëse vektori përmban vlera numerike negative.

Anëtarët e vektorit mund të radhiten edhe në bazë të vlerave absolute.

Shembull Radhitja sipas vlerave absolute të anëtarëve të vektorit `A(n)`, prej vlerës më të madhe.

a. Bllok-diagrami *b. Rruga* - për `A=`

3	-7	5	-4

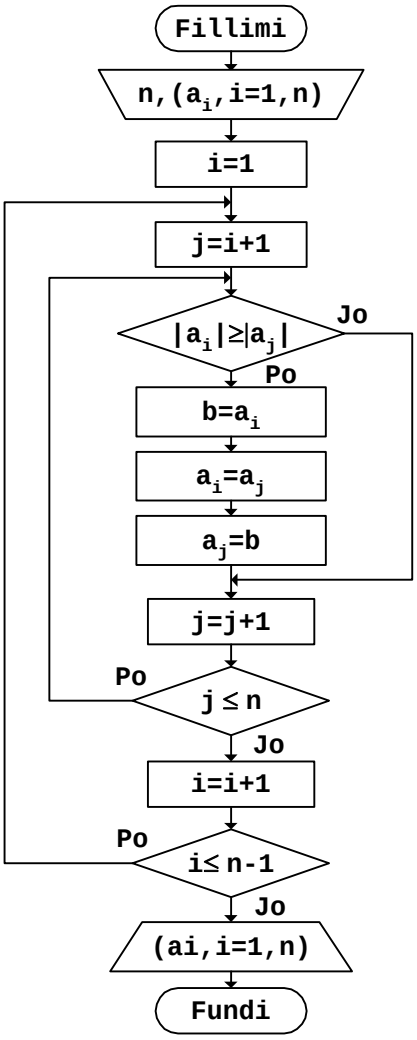


Fig.6.42

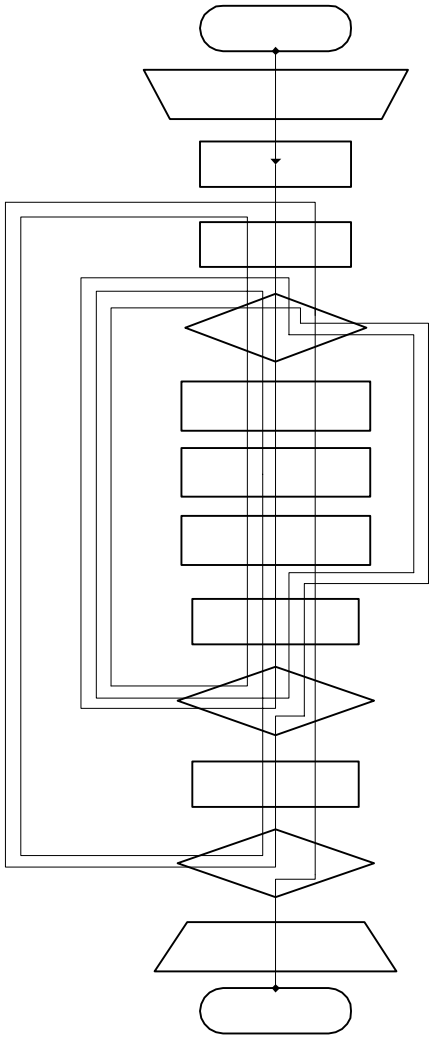


Fig.6.43

c. Programi

```
// Programi Prg6_42
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
```

```
  int const n=4;
  int A[n]={3,-7,5,-4},i,j,b;
  for (i=0;i<n-1;i++)
    for (j=i+1;j<n;j++)
    {
        if (abs(A[i])>=abs(A[j]))
        {
        }
        else
        {
           b=A[i];
           A[i]=A[j];
           A[j]=b;
        }
    }
  cout << "A=[  ";
  for (i=0;i<n;i++)
     cout << A[i]
          << "  ";
  cout << "]"
       << "\n";
  return 0;
}
```

Si rezultat, pas ekzekutimit të programit të dhënë në ekran, do të shtypet vektori me vlerat e radhitura brenda tij, kështu:

```
A=[  -7  5  -4  3  ]
```

meqë është kërkuar radhitja në bazë të vlerave absolute.

Detyra

Të radhiten sipas madhësisë anëtarët e vektorit të dhënë `G(m)`:

a. prej më të madhit kah më i vogli;

b. si vlera absolute, prej më të voglit kah më i madhi.

Matricat

Gjatë punës me matrica, për t'i shfrytëzuar anëtarët e vendosur në fushat e veçanta të tyre, përdoren indekset e rreshtave dhe të kolonave përkatëse.

Përcaktimi i matricave

Vlerat e anëtarëve të matricave kompjuterit mund t'i jepen si vlera të gatshme hyrëse, ose duke i llogaritur ato në bazë të procedurës së dhënë.

Shembull

Formimi i matricës `A(m,n)`, duke i llogaritur anëtarët a_{ij} të saj kështu:

$$a_{ij} = \begin{cases} 2i + 3j & \text{për } i < j \\ i + j & \text{për } i = j \\ 3i - j^2 & \text{për } i > j \end{cases}$$

nëse dihen dimensionet e matricës.

a. Bllok-diagrami

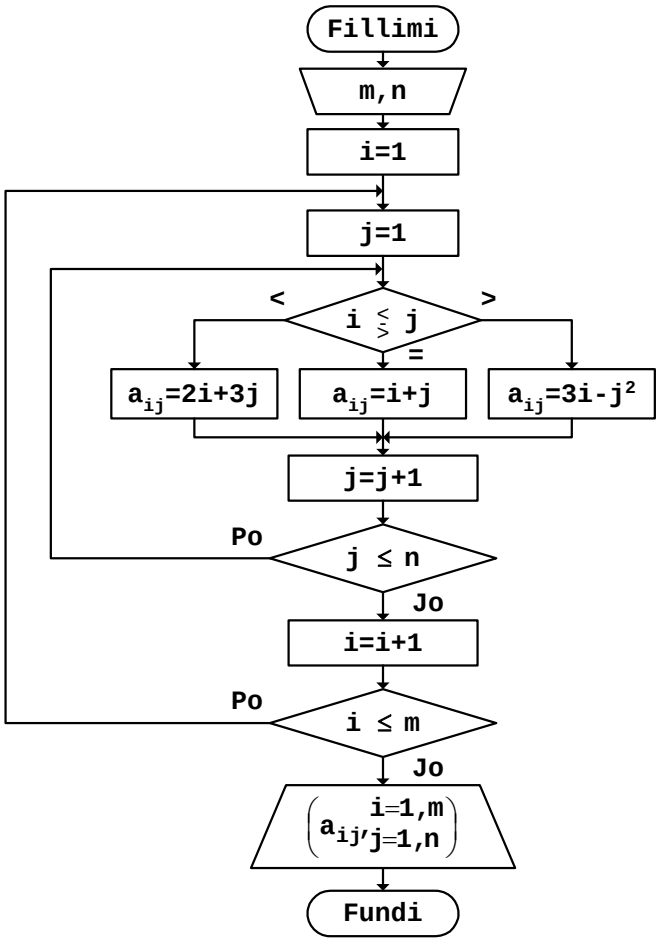


Fig.6.44

b. Rruga - për m=2, n=3

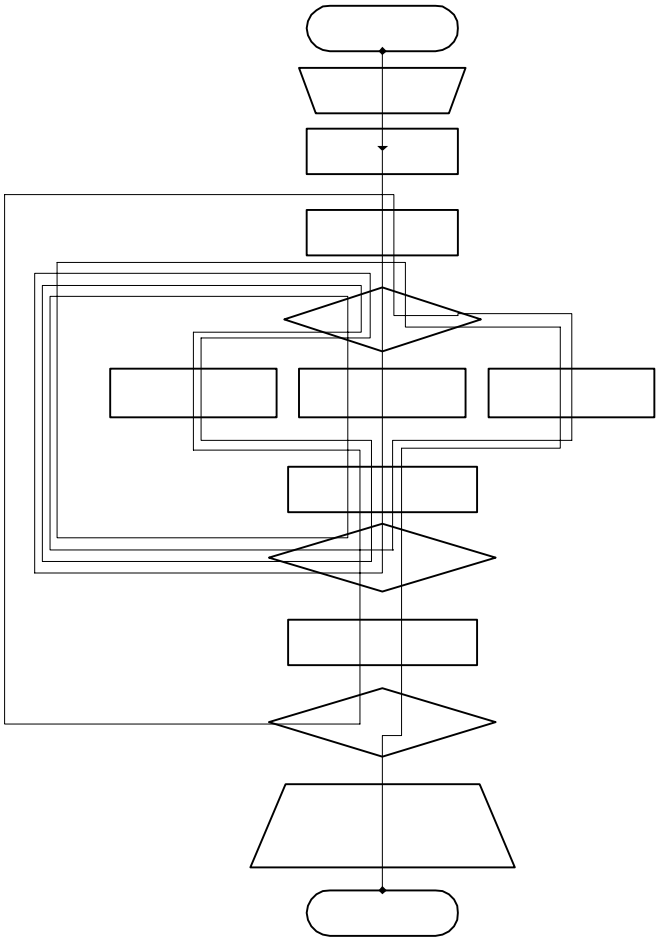


Fig.6.45

c. Programi

```
// Programi Prg6_44
#include <iostream>
using namespace std;
int main()
{
  int const m=2,n=3;
  int i,j,A[m][n];
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
      if (i<j)
```

```
            A[i][j]=2*i+3*j;
         else
            if (i==j)
               A[i][j]=i+j;
            else
               A[i][j]=3*i-j*j;
   cout << "Matrica e formuar"
        << "\n";
   for (i=0;i<m;i++)
   {
       for (j=0;j<n;j++)
           cout << A[i][j]
                << "  ";
       cout << "\n";
   }
   return 0;
}
```

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

```
Matrica e formuar
   0   3   6
   3   2   8
```

Ligjshmëritë për llogaritjen e anëtarëve të matricave, përkatësisht për formimin e tyre, mund të jenë të ndryshme.

Shembull Formimi i matricës `R(m,n)`, duke llogaritur anëtarët e saj kështu:

$$r_{ij} = \begin{cases} 3\sum_{k=1}^{i+j+1}(k + 2i - 3j) & \text{për } i < j \\ (i + j)! & \text{për } i \geq j \end{cases}$$

a. Bllok-diagrami

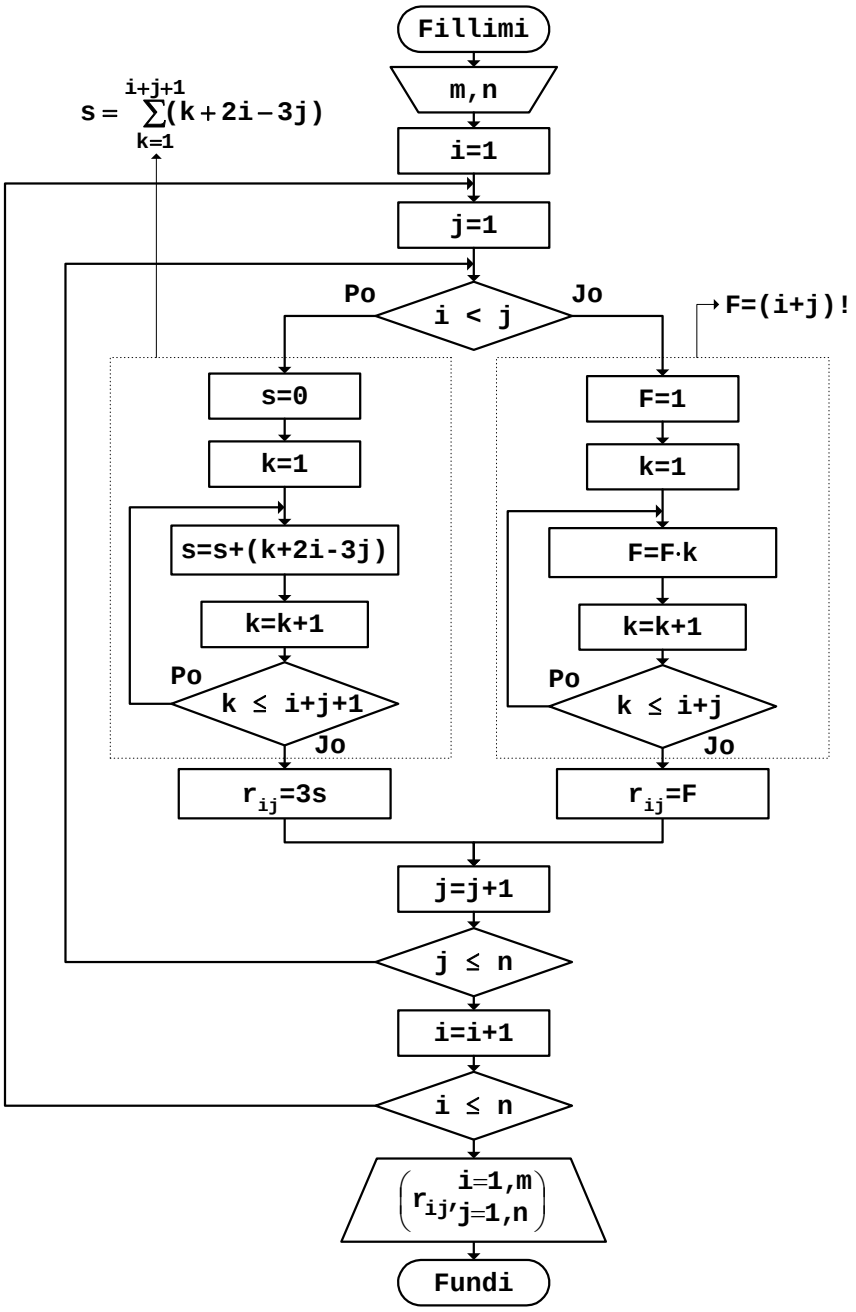

Fillimi
m,n
i=1
j=1
Po
i < j
Jo
F=(i+j)!
s=0
k=1
s=s+(k+2i-3j)
k=k+1
Po
k ≤ i+j+1
Jo
r_ij=3s
F=1
k=1
F=F·k
k=k+1
Po
k ≤ i+j
Jo
r_ij=F
j=j+1
j ≤ n
i=1
i ≤ n
i=1,m
r_ij, j=1,n
Fundi

Fig.6.46

Këtu, nuk është vizatuar edhe rruga që kalohet në bllok-diagram për vlera të caktuara të dimensioneve të matricës `m` dhe `n`, meqë numri i tyre është i madhë, gjë që do të praktikohet edhe në shumicën e shembujve të pjesës vijuese të librit.

b. Programi

```
// Programi Prg6_46
#include <iostream>
using namespace std;
int main()
{
     int const m=2,n=3;
     int i,j,k,s,f,R[m][n];
     for (i=0;i<m;i++)
         for (j=0;j<n;j++)
             if (i<j)
             {
               s=0;
               for (k=1;k<=(i+j+1);k++)
                   s=s+(k+2*i-3*j);
               R[i][j]=3*s;
             }
             else
             {
               f=1;
               for (k=1;k<=(i+j);k++)
                   f=f*k;
               R[i][j]=f;
             }
     cout << "Matrica e formuar"
          << "\n";
     for (i=0;i<m;i++)
     {
          for (j=0;j<n;j++)
          {
               cout.width(5);
               cout << R[i][j];
          }
          cout << "\n";
     }
     return 0;
}
```

Nëse ekzekutohet programi i dhënë për matricën me dimensione `m=2` dhe `n=3`, rezultati që shtypet në ekran është:

```
Matrica e formuar
    1   -9  -36
    1    2  -18
```

Detyra

Të formohet matrica katrore `G(m,m)`, duke përcaktuar elementet e saj kështu:

a.

G =	j=1	2	3	...	m
i=1	1	5	5	...	5
2	-3	1	5	...	5
3	-3	-3	1	...	5
...	...	...	...	...	...
m	-3	-3	-3	...	1

b.

G =	j=1	2	3	...	m
i=1	0	2	3	...	m
2	2	0	3	...	m
3	2	2	0	...	m
...	...	...	...	...	...
m	2	2	2	...	0

Matricat mund të formohen edhe duke shfrytëzuar vlerat e anëtarëve të vektorëve të dhënë.

Shembull Formimi i matricës katrore `A(m,m)`, duke shfrytëzuar anëtarët e vektorit të dhënë `D(m)`, kështu:

$$A = \begin{array}{c|ccccc} & j=1 & 2 & 3 & \dots & m \\ \hline i=1 & d_1 & d_1+2 & d_1+3 & \dots & d_1+m \\ 2 & d_2+8 & d_2 & d_2+3 & \dots & d_2+m \\ 3 & d_3+8 & d_3+8 & d_3 & \dots & d_3+m \\ \dots & \dots & \dots & \dots & \dots & \dots \\ m & d_m+8 & d_m+8 & d_m+8 & \dots & d_m \end{array}$$

a. Bllok-diagrami

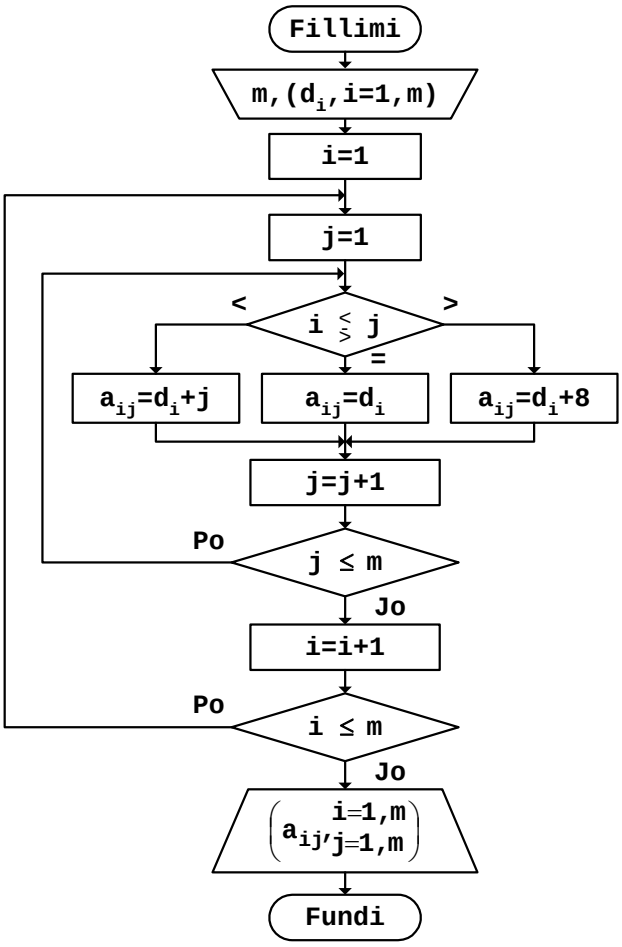


Fig.6.47

Siç shihet edhe nga bllok-diagrami, për `i<j` duhet të mbushen anëtarët mbi diagonalen kryesore, kurse kur është `i>j` kemi të bëjmë me anëtarët nën diagonalen kryesore.

b. Programi

```
// Programi Prg6_47
#include <iostream>
using namespace std;
int main()
{
   int const m=5;
   int D[m]={3,-7,4,9,-2};
   int i,j,A[m][m];
   for (i=0;i<m;i++)
       for (j=0;j<m;j++)
           if (i<j)
              A[i][j]=D[i]+j;
           else
              if (i==j)
                  A[i][j]=D[i];
              else
                  A[i][j]=D[i]+8;
   cout << "Matrica e formuar"
        << "\n";
   for (i=0;i<m;i++)
   {
      for (j=0;j<m;j++)
      {
          cout.width(5);
          cout << A[i][j];
      }
      cout << "\n";
   }
   return 0;
}
```

Rezultati që shtypet në ekran pas ekzekutimit të programit të dhënë është:

```
Matrica e formuar
    3    4    5    6    7
    1   -7   -5   -4   -3
   12   12    4    7    8
   17   17   17    9   13
    6    6    6    6   -2
```

Me vlerat e anëtarëve të vektorit, përveç diagonales kryesore që u tregua në shembullin e mësipërm, mund të mbushet edhe një rresht, ose një kolonë e caktuar e matricës.

Shembull Formimi i matricës `G(m,m)`, duke i shfrytzëzuar edhe anëtarët e vektorit `B(m)`:

		j=1	2	3	...	m
	i=1	b_1	1	1	...	1
	2	b_2	2	2	...	2
G =	3	b_3	3	3	...	3
	...	...	...	...	...	...
	m	b_m	m	m	...	m

a. Bllok-diagrami

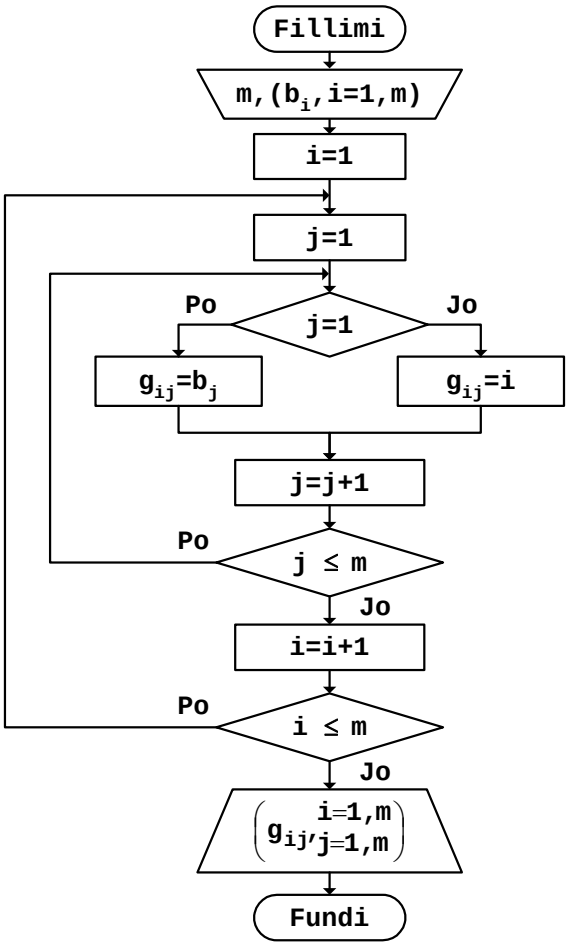


Fig.6.48

b. Programi

```
// Programi Prg6_48
#include <iostream>
using namespace std;
int main()
{
  int const m=5;
  int B[m]={9,2,-4,6,-5};
  int i,j,G[m][m];
  for (i=0;i<m;i++)
      for (j=0;j<m;j++)
          if (j==0)
              G[i][j]=B[i];
          else
              G[i][j]=i;
  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<m;i++)
  {
      for (j=0;j<m;j++)
      {
          cout.width(5);
          cout << G[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

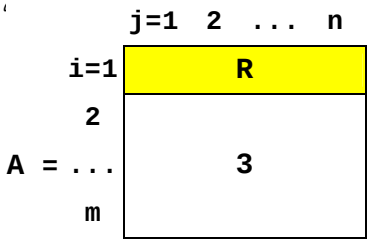
Nëse programi i dhënë ekzekutohet, si rezultat në ekran do të fitohet:

```
Matrica e formuar
    9    0    0    0    0
    2    1    1    1    1
   -4    2    2    2    2
    6    3    3    3    3
   -5    4    4    4    4
```

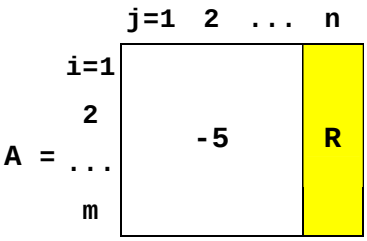
Detyra

Të formohet matrica katrore `A(m,m)`, nëse anëtarët e vektorit të dhënë `R(m)` brenda matricës vendosen kështu:

a.



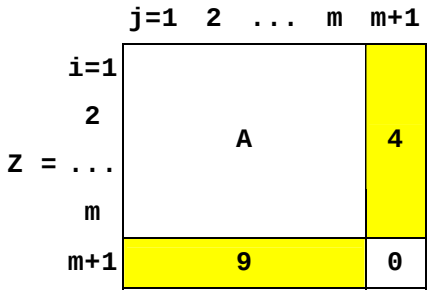
b.



Formimi i matricave mund të mbështetet edhe në shfrytëzimin e anëtarëve të matricave të tjera.

Shembull

Formimi i matricës `Z`, duke shfrytëzuar anëtarët e matricës `A(m,m)`, kështu:



a. Bllok-diagrami

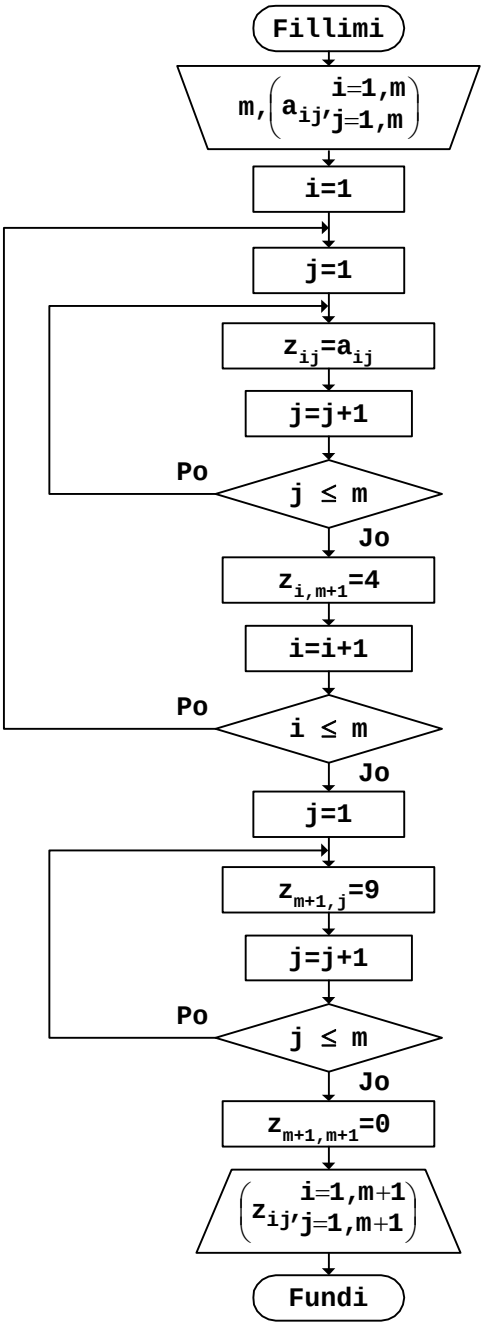


Fig.6.49

b. Programi

```
// Programi Prg6_49
#include <iostream>
using namespace std;
int main()
{
  int const m=4;
  int A[m][m]={ {3,5,-4,2},
                {6,9,-2,8},
                {1,-3,7,4},
                {2,0,-5,3}
              };
  int i,j,Z[m+1][m+1];

  for (i=0;i<m;i++)
  {
      for (j=0;j<m;j++)
          Z[i][j]=A[i][j];
      Z[i][m]=4;
  }
  for (j=0;j<m;j++)
      Z[m][j]=9;
  Z[m][m]=0;
  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<=m;i++)
  {
      for (j=0;j<=m;j++)
      {
          cout.width(5);
          cout << Z[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Pas ekzekutimit të programit, rezultati që shtypet në ekran është:

```
Matrica e formuar
    3    5   -4    2    4
    6    9   -2    8    4
    1   -3    7    4    4
    2    0   -5    3    4
    9    9    9    9    0
```

Algoritmi për formimin e matricës `Z` mund të realizohet edhe ndryshe, duke paraparë mbushjen e të gjitha pjesëve të saj brenda unazave për indekset `i` dhe `j`, të cilat ndryshohen mes vlerave `1` dhe `m+1`, ashtu siç është treguar në vijim.

a. Bllok-diagrami

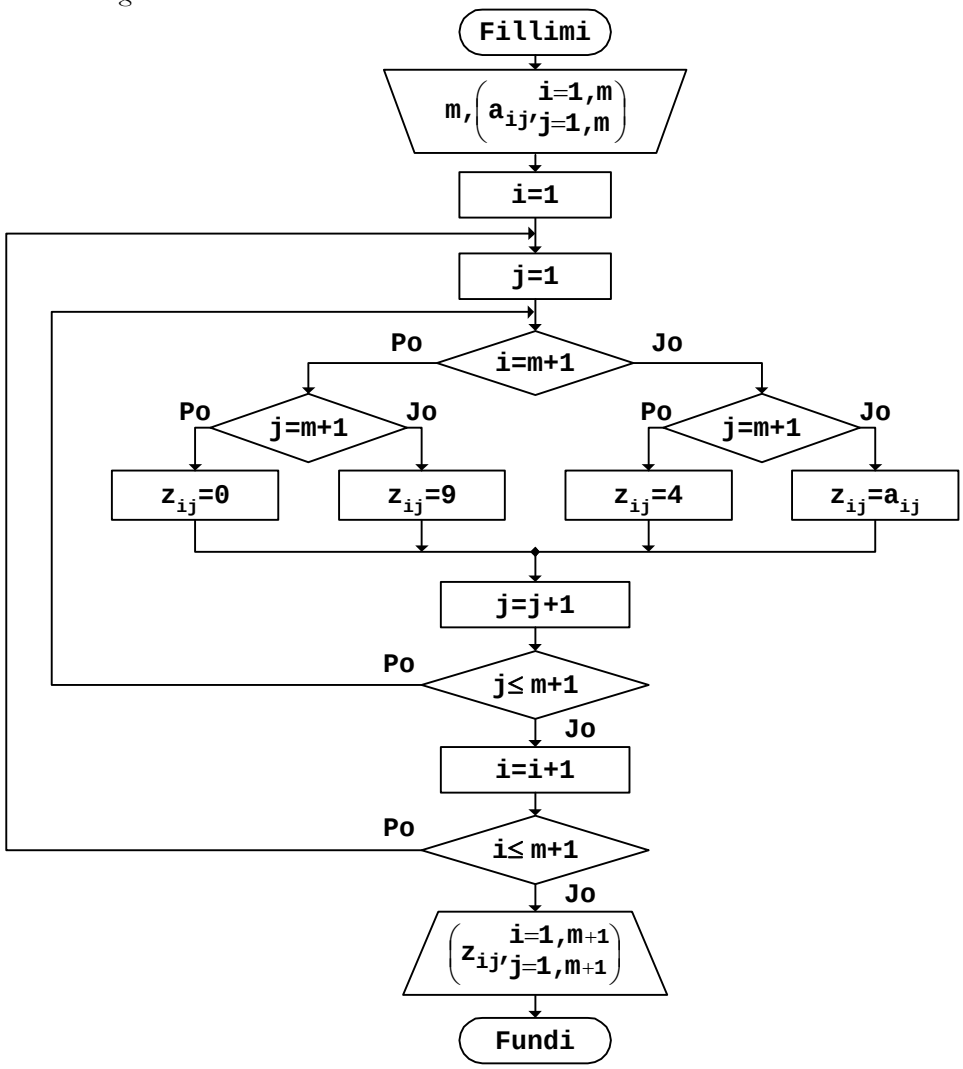


Fig.6.50

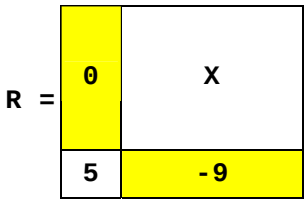
b. Programi

```
// Programi Prg6_50
#include <iostream>
using namespace std;
int main()
{
      int const m=4;
      int A[m][m]={ {3,5,-4,2},
                    {6,9,-2,8},
                    {1,-3,7,4},
                    {2,0,-5,3}
                  };
      int i,j,Z[m+1][m+1];
      for (i=0;i<=m;i++)
          for (j=0;j<=m;j++)
              if (i==m)
                 if (j==m) Z[i][j]=0;
                  else      Z[i][j]=9;
              else
                 if (j==m) Z[i][j]=4;
                 else      Z[i][j]=A[i][j];
      cout << "Matrica e formuar"
           << "\n";
      for (i=0;i<=m;i++)
      {
          for (j=0;j<=m;j++)
          {
                  cout.width(5);
                  cout << Z[i][j];
          }
          cout << "\n";
      }
      return 0;
}
```

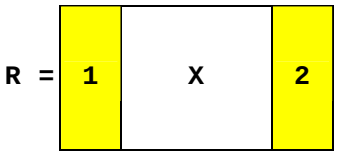
Detyra

Duke shfrytëzuar anëtarët e matricës `X(m,m)`, të formohet matrica `R`, kështu:

a.



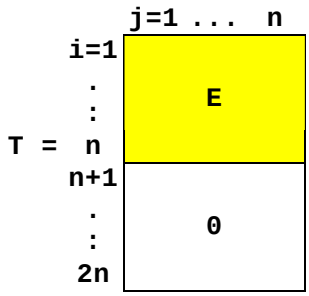
b.



Gjatë formimit të matricave, pjesë të caktuara të tyre mund të mbushen edhe me vlerat e matricave të njohura, p.sh., siç është matrica njësi, ose matrica zero etj.

Shembull

Formimi i matricës `T`, kështu:



ku `E(n,n)` është matrica njësi, kurse `O(n,n)` është matrica zero.

a. Bllok-diagrami

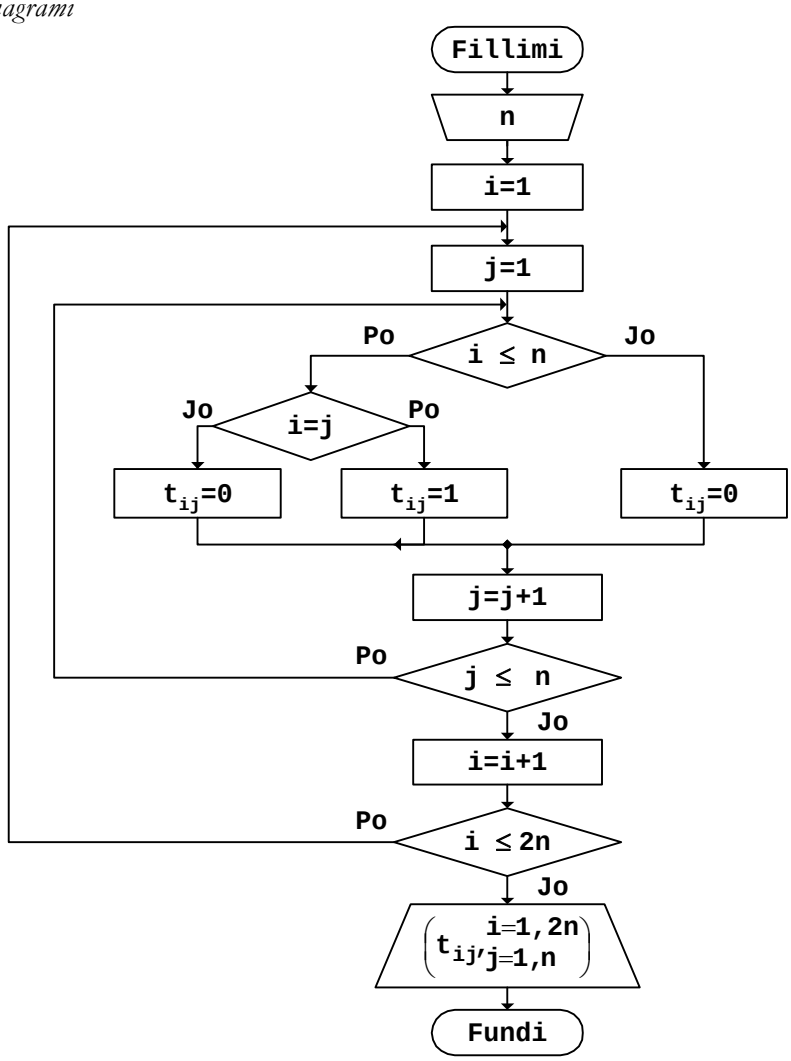


Fig.6.51

b. Programi

```
// Programi Prg6_51
#include <iostream>
using namespace std;
int main()
{
  int const n=5;
  int i,j,T[2*n][n];
```

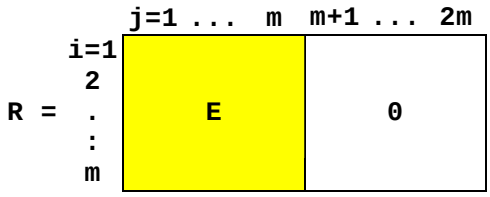
```
  for (i=0;i<(2*n);i++)
    for (j=0;j<n;j++)
        if (i<n)
           if (i==j)
              T[i][j]=1;
           else
              T[i][j]=0;
        else
           T[i][j]=0;

  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<(2*n);i++)
  {
    for (j=0;j<n;j++)
    {
        cout.width(3);
        cout << T[i][j];
    }
    cout << "\n";
  }
  return 0;
}
```

Rezultati që shtypet në ekran pas ekzekutimit të programit të dhënë është:

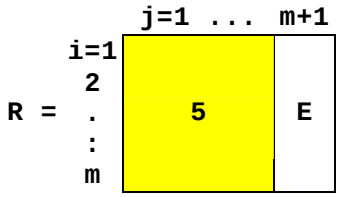
```
Matrica e formuar
  1  0  0  0  0
  0  1  0  0  0
  0  0  1  0  0
  0  0  0  1  0
  0  0  0  0  1
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
```

Detyra Të formohet matrica R, kështu:

a.



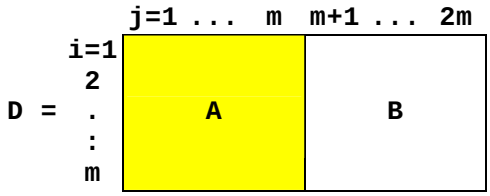
b.



ku `E(m,m)` është matrica njësi, kurse `O(m,m)` është matrica zero.

Matricat mund të formohen edhe duke shfrytëzuar njëkohësisht më shum matrica, me vlera të çfarëdoshme.

Shembull Formimi i matricës `D` nga anëtarët e matricave `A(m,m)` dhe `B(m,m)`, kështu:



a. Bllok-diagrami

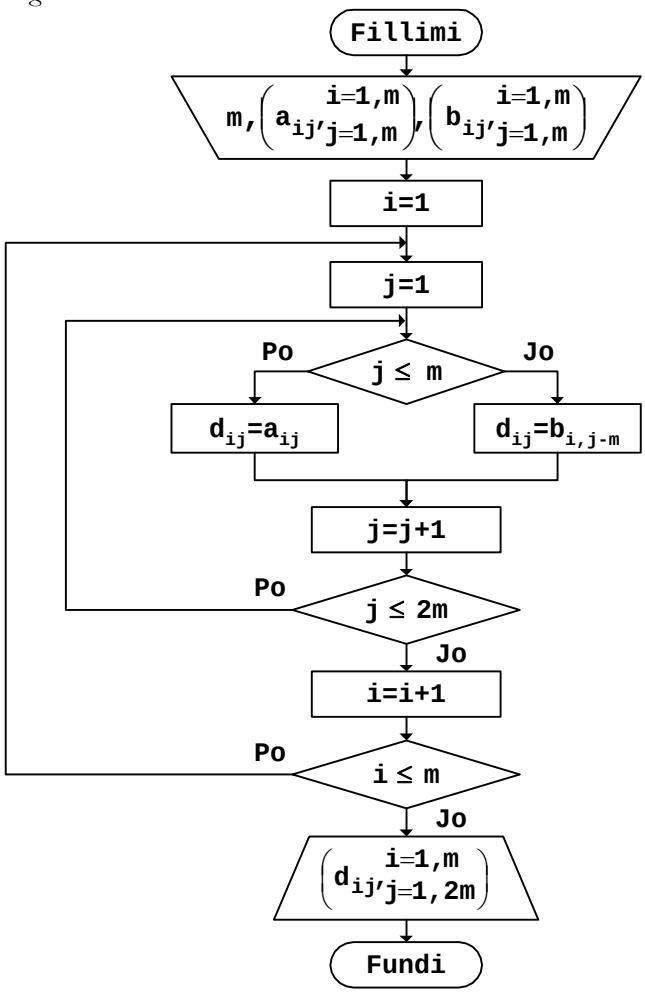


Fig.6.52

b. Programi

```
// Programi Prg6_52
#include <iostream>
using namespace std;
int main()
{
  int const m=3;
  int A[m][m]={ {4,7,3},
                {-2,3,9},
```

```
                {8,-4,2}
              };
  int B[m][m]={ {-2,9,1},
                {4,8,3},
                {6,1,7}
              };
  int i,j,D[m][2*m];
  for (i=0;i<m;i++)
      for (j=0;j<(2*m);j++)
          if (j<m)
             D[i][j]=A[i][j];
          else
             D[i][j]=B[i][j-m];
  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<m;i++)
  {
      for (j=0;j<(2*m);j++)
      {
          cout.width(4);
          cout << D[i][j];
      }
    cout << "\n";
  }
  return 0;
}
```

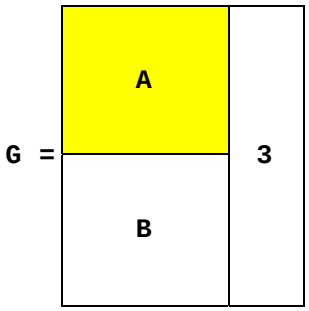
Pas ekzekutimit të programit të dhënë, rezultati që shtypet në ekran është:

```
Matrica e formuar
   4   7   3  -2   9   1
  -2   3   9   4   8   3
   8  -4   2   6   1   7
```

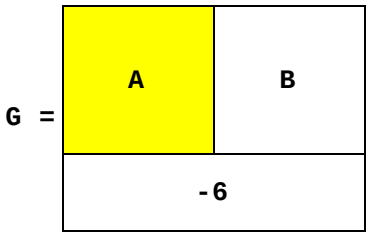
Detyra

Të formohet matrica `G`, duke shfrytëzuar anëtarët e matricave `A(m,n)` dhe `B(m,n)`, kështu:

a.



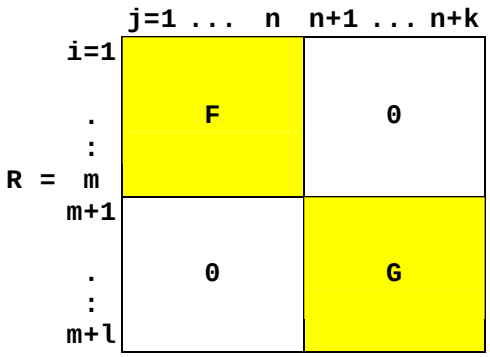
b.



Kur në formimin e matricës përdoren më shum matrica, mbushja e saj në bllok-diagram mund të realizohet duke shfrytëzuar pjesë të veçanta për çdo matricë, ose përmes një tërësie të përbërë për të gjitha matricat.

Shembull

Formimi i matricës `R`, duke shfrytëzuar anëtarët e matricave `F(m,n)` dhe `G(l,k)`, kështu:



ku me `0` është shënuar matrica zero.

a. Bllok-diagrami

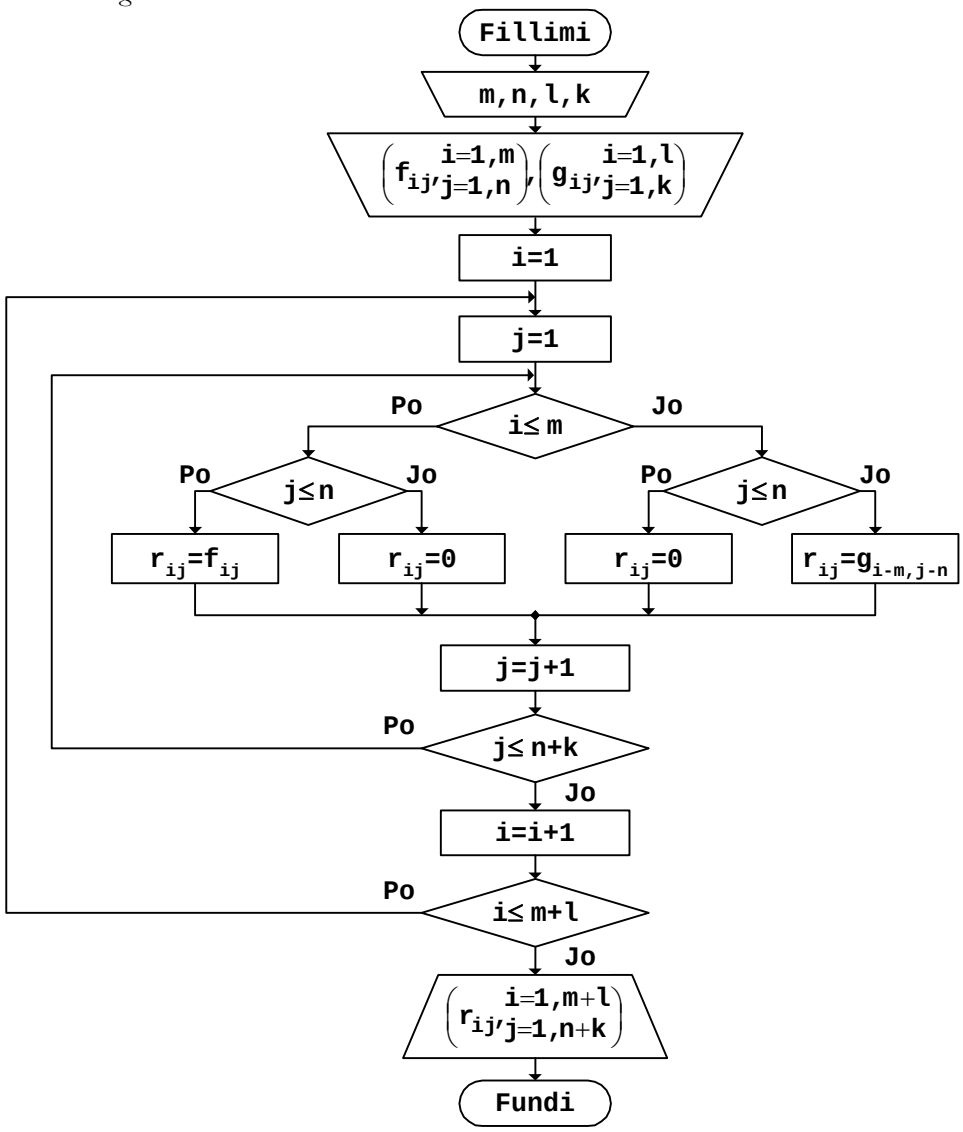


Fig.6.53

b. Programi

```
// Programi Prg6_53
#include <iostream>
using namespace std;
int main()
```

```
{
  int const m=3,n=4,l=4,k=3;
  int F[m][n]={ {4,7,3,5},
                {-2,3,9,2},
                {8,-4,2,7}
              };
  int G[l][k]={ {-2,9,1},
                {4,8,3},
                {6,1,7},
                {-9,4,2}
              };
  int i,j,R[m+l][n+k];
  for (i=0;i<m+l;i++)
    for (j=0;j<n+k;j++)
        if (i<m)
            if (j<n)
               R[i][j]=F[i][j];
            else
               R[i][j]=0;
        else
            if (j<n)
               R[i][j]=0;
            else
               R[i][j]=G[i-m][j-n];
  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<m+l;i++)
  {
      for (j=0;j<n+k;j++)
      {
          cout.width(4);
          cout << R[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

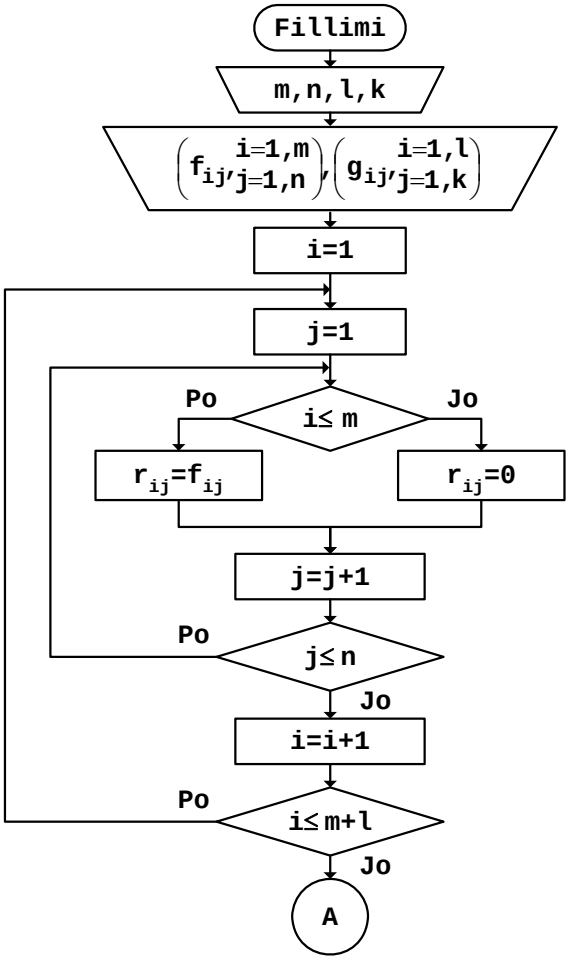
Rezultati që do të shtypet në ekran, nëse ekzekutohet programi i dhënë, është:

```
Matrica e formuar
   4   7   3   5   0   0   0
  -2   3   9   2   0   0   0
   8  -4   2   7   0   0   0
   0   0   0   0  -2   9   1
```

```
0   0   0   0   4   8   3
0   0   0   0   6   1   7
0   0   0   0  -9   4   2
```

Këtu, siç u theksua edhe më sipër, mund të zbatohet procedura e mbushjes parciale të matricës. Kështu, nëse, p.sh., mbushet së pari pjesa e majtë e matricës dhe pastaj pjesa e djathtë e saj, algoritmi përkatës do të duket si në vijim.

a. Bllok-diagrami



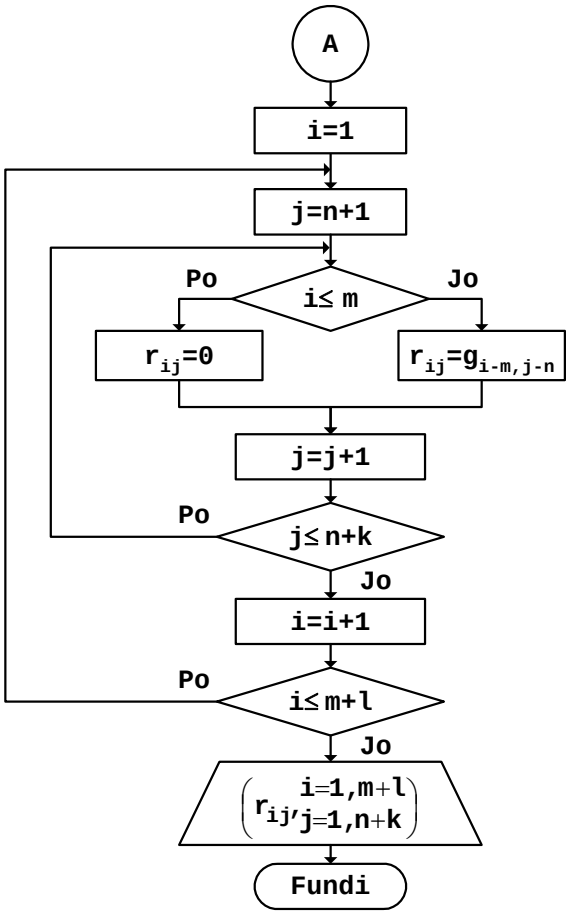


Fig.6.54

b. Programi

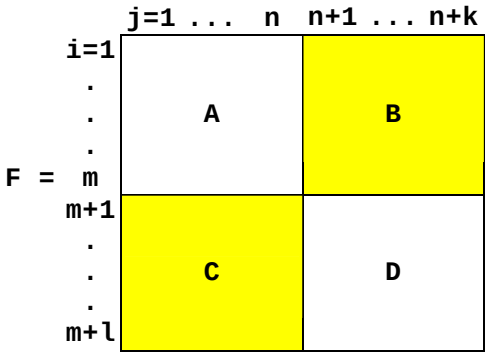
Pjesa e programit për mbushjen e matricës me vlera, e cila ndryshon nga programi që u dha më sipër, duket si në vijim.

```
// Programi Prg6_54
.....................................
      for (i=0;i<m+l;i++)
          for (j=0;j<n;j++)
               if (i<m)
                   R[i][j]=F[i][j];
               else
                  R[i][j]=0;
      for (i=0;i<m+l;i++)
```

```
          for (j=n;j<n+k;j++)
              if (i<m)
                 R[i][j]=0;
              else
                 R[i][j]=G[i-m][j-n];
.........................................
```

Nëse matricat që përdoren për mbushje kanë vlera të caktuara, gjatë vendosjes së vlerave në matricë duhet pasur kujdes në indekset e anëtarëve të cilët shfrytëzohen për mbushje.

Shembull Formimi i matricës `F`, duke shfrytëzuar anëtarët e matricave `A(m,n)`, `B(m,k)`, `C(l,n)` dhe `D(l,k)`, kështu:



a. Bllok-diagrami

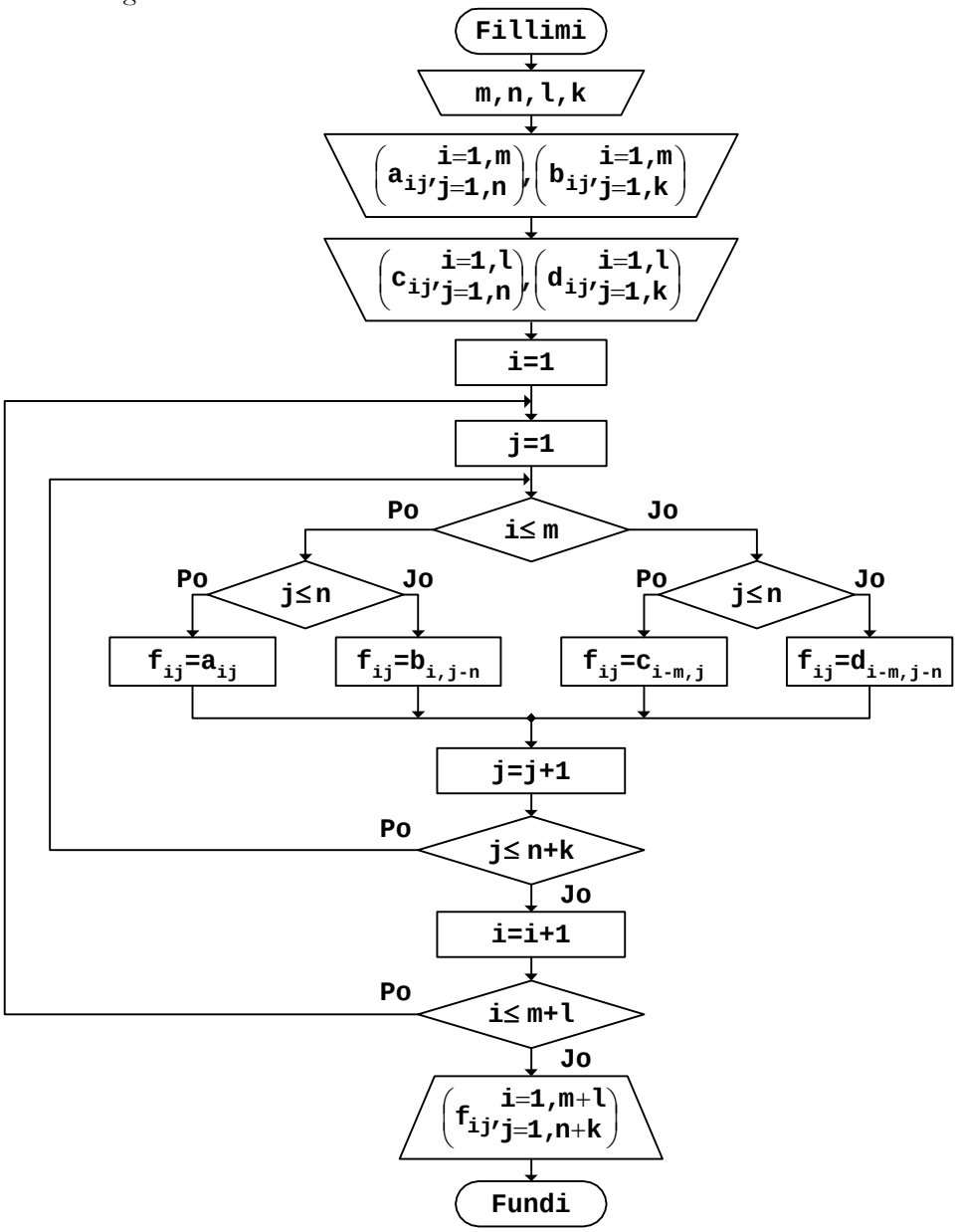


Fig.6.55

b. Programi

```
// Programi Prg6_55
#include <iostream>
using namespace std;
int main()
{
  int const m=3,n=2,l=2,k=3;
  int A[m][n]={ {4,7},
                {-2,5},
                {8,3}};
  int B[m][k]={ {-1,9,6},
                {2,3,8},
                {4,1,5}
              };
  int C[l][n]={ {1,2},
                {3,4}
              };
  int D[l][k]={ {5,4,3},
                {2,1,0}
              };
  int i,j,F[m+l][n+k];
  for (i=0;i<m+l;i++)
      for (j=0;j<n+k;j++)
          if (i<m)
              if (j<n)
                  F[i][j]=A[i][j];
              else
                  F[i][j]=B[i][j-n];
          else
              if (j<n)
                  F[i][j]=C[i-m][j];
              else
                  F[i][j]=D[i-m][j-n];
  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<m+l;i++)
  {
      for (j=0;j<n+k;j++)
      {
          cout.width(4);
          cout << F[i][j];
    }
    cout << "\n";
  }
```

```
  return 0;
}
```

Pas ekzekutimit të programit, matrica F e shtypur në ekran duket kështu:

```
Matrica e formuar
   4    7   -1    9    6
  -2    5    2    3    8
   8    3    4    1    5
   1    2    5    4    3
   3    4    2    1    0
```

Detyra

Të formohet matrica F, duke shfrytëzuar anëtarët e matricave A(m,n) dhe B(k,l), si dhe matricën njësi E dhe matricën zero 0, ashtu siç është treguar më poshtë.

a.

F=

E	**A**
B	**0**

b.

F=

0	**E**
A	**0**

c.

F=

E	**0**
0	**E**

d.

F=

A	**E**
B	**0**

Operacionet aritmetikore

Mbi anëtarët e matricave mund të zbatohen operacionet elementare aritmetikore.

Mbledhja dhe zbritja

Mblidhen vetëm matricat të cilat kanë dimensione të barabarta. Operacioni i mbledhjes kryhet duke i mbledhur anëtarët në pozicionet e njëjta të dy matricave, përkatësisht anëtarët me indekse të njëjtë.

Shembull Gejtja e shumës `C(m,n)` të matricave `A(m,n)` dhe `B(m,n)`.

$$C = A+B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{bmatrix}$$

$$c_{11}=a_{11}+b_{11}$$
$$c_{12}=a_{12}+b_{12}$$
$$\dots\dots\dots$$
$$c_{ij}=a_{ij}+b_{ij}$$
$$\dots\dots\dots$$
$$c_{mn}=a_{mn}+b_{mn}$$

a. Bllok-diagrami

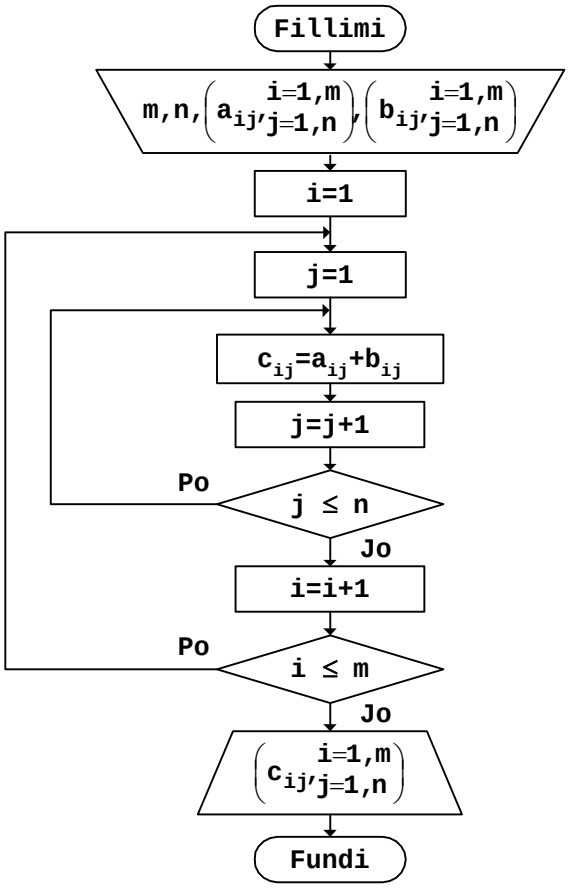


Fig.6.56

c. Programi

```
// Programi Prg6_56
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {4,7,8,-6,9},
                {1,-2,5,4,6},
                {8,3,2,-1,0},
                {-3,5,8,4,1}
              };
  int B[m][n]={ {-1,6,9,6,4},
```

```
                {2,3,7,4,-8},
                {4,3,2,-1,5},
                {9,-2,6,4,1}
              };
  int i,j,C[m][n];
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          C[i][j]=A[i][j]+B[i][j];
  cout << "Matrica e formuar"
       << "\n";
  for (i=0;i<m;i++)
  {
      for (j=0;j<n;j++)
      {
          cout.width(4);
          cout << C[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Duke pasur parasysh vlerat hyrëse për matricat A dhe B, të cilat kompjuterit i janë dhënë si konstante, rezultati që shtypet në ekran pas ekzekutimit të programit është:

```
Matrica e formuar
   3  13  17   0  13
   3   1  12   8  -2
  12   6   4  -2   5
   6   3  14   8   2
```

Operacioni i zbritjes kryhet plotësisht njëlloj si edhe mbledhja, që do të thotë se shfrytëzohet algoritmi, përkatësisht programi i njëjtë.

Shumëzimi

Matrica shumëzohet me vektorë nëse numri i kolonave të matricës është i barabartë me numrin e anëtarëve të vektorit. Rezultati që fitohet pas shumëzimit të matricës me vektorë është vektor, i cili do të ketë aq anëtarë sa ka rreshta matrica.

Shembull Prodhimi `T(m)` i matricës `A(m,n)` me vektorin `B(n)`.

$$T = A \cdot B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} =$$

$$= \begin{bmatrix} a_{11} \cdot b_1 + a_{12} \cdot b_2 + \dots + a_{1n} \cdot b_n \\ a_{21} \cdot b_1 + a_{22} \cdot b_2 + \dots + a_{2n} \cdot b_n \\ \dots \\ a_{m1} \cdot b_1 + a_{m2} \cdot b_2 + \dots + a_{mn} \cdot b_n \end{bmatrix}$$

$$t_1 = a_{11} \cdot b_1 + a_{12} \cdot b_2 + \dots + a_{1n} \cdot b_n$$
$$t_2 = a_{21} \cdot b_1 + a_{22} \cdot b_2 + \dots + a_{2n} \cdot b_n$$
$$\dots\dots\dots\dots\dots\dots\dots$$
$$t_i = a_{i1} \cdot b_1 + a_{i2} \cdot b_2 + \dots + a_{in} \cdot b_n = \sum_{j=1}^{n} a_{ij} \cdot b_j$$
$$\dots\dots\dots\dots\dots\dots\dots$$
$$t_m = a_{m1} \cdot b_1 + a_{m2} \cdot b_2 + \dots + a_{mn} \cdot b_n$$

a. Bllok-diagrami

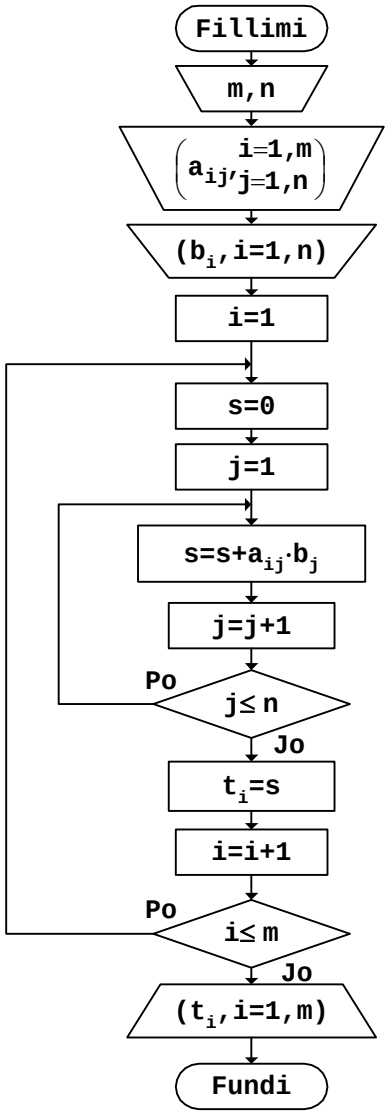


Fig.6.57

b. Programi

```
// Programi Prg6_57
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,5,8,-1,4},
                {7,-4,9,2,1},
                {6,2,1,5,-7},
                {2,4,6,-8,3}
              };
  int B[n]={-1,6,9,6,4};
  int i,j,s,T[m];
  for (i=0;i<m;i++)
  {
      s=0;
      for (j=0;j<n;j++)
      s=s+A[i][j]*B[j];
      T[i]=s;
  }
  cout << "Vektori i prodhimit"
       << "\n";
  cout << "T=[  ";
  for (i=0;i<m;i++)
      cout << T[i]
           << "  ";
  cout << "]"
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultati i prodhimit në ekran shtypet kështu:

```
Vektori i prodhimit
T=[ 109  66  17  40 ]
```

Detyra

Të gjendet:

a. matrica `Y(m,n)` si prodhim i matricës `X(m,n)` me konstanten `k`;

b. vektori `F(n)` si prodhim i vektorit `A(m)` me matricën `B(m,n)`.

Dy matrica shumëzohen, nëse numri i kolonave të matricës së parë është i barabartë me numrin e rreshtave të matricës së dytë. Shumëzimi i dy matricave e jep si rezultat matricën tek e cila ruhet numri i rreshtave të matricës së parë dhe numri i kolonave të matricës së dytë.

Shembull Gjetja e matricës `C(m,n)` si prodhim i matricës `A(m,k)` me matricën `B(k,n)`.

$$C = A \cdot B = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mk} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{k1} & b_{k2} & \dots & b_{kn} \end{pmatrix}$$

$$c_{11}=a_{11}\cdot b_{11}+a_{12}\cdot b_{21}+\dots+a_{1k}\cdot b_{k1}$$
$$c_{12}=a_{11}\cdot b_{12}+a_{12}\cdot b_{22}+\dots+a_{1k}\cdot b_{k2}$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$c_{1k}=a_{11}\cdot b_{1n}+a_{12}\cdot b_{2n}+\dots+a_{1k}\cdot b_{kn} = \sum_{l=1}^{k} a_{il}\cdot b_{lj}$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$c_{ij}=a_{i1}\cdot b_{1j}+a_{i2}\cdot b_{2j}+\dots+a_{ik}\cdot b_{kj}$$
$$\dots\dots\dots\dots\dots\dots\dots\dots$$
$$c_{mn}=a_{m1}\cdot b_{1n}+a_{m2}\cdot b_{2n}+\dots+a_{mk}\cdot b_{kn}$$

a. Bllok-diagrami

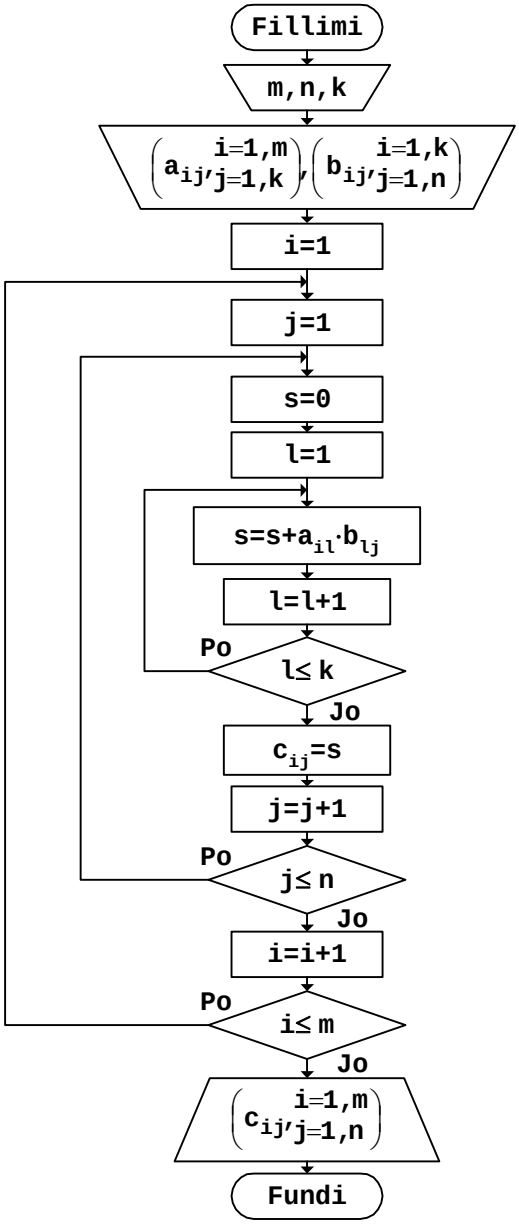


Fig.6.58

b. Programi

```
// Programi Prg6_58
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5,k=3;
  int A[m][k]={ {3,5,8},
                {7,-4,9},
                {6,2,1},
                {2,4,6}
              };
  int B[k][n]={ {-1,6,9,6,4},
                {2,6,8,4,-3},
                {5,3,-2,5,9}
              };
  int i,j,l,s,C[m][n];
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
      {
          s=0;
          for (l=0;l<k;l++)
              s=s+A[i][l]*B[l][j];
          C[i][j]=s;
      }
  cout << "Matrica e prodhimit C"
       << "\n";
  for (i=0;i<m;i++)
  {
      for (j=0;j<n;j++)
      {
          cout.width(5);
          cout << C[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Rezultati që fitohet në ekran pas ekzekutimit të programit të dhënë është:

```
Matrica e prodhimit C
  47  72  51  78  69
  30  45  13  71 121
   3  51  69  49  27
  36  54  38  58  50
```

Ekuacionet matricore

Duke u mbështetur në rregullat e ekzekutimit të operacioneve elementare aritmetikore të dhëna më sipër, mund të formohen ekuacione matricore.

Shembull Llogaritja e anëtarëve të matricës `R(m,k)`, nga matricat `X(m,k)`, `Y(k,n)` dhe `Z(m,n)`, përmes ekuacionit matricor:

```
R=X·Y-Z
```

Nga shprehja e ekuacionit të dhënë shihet se së pari duhet llogaritur anëtarët e prodhimit `X·Y` dhe pastaj, përmes zbritjes përkatëse, gjendet rezultati `R` i ekuacionit matricor.

a. Bllok-diagrami

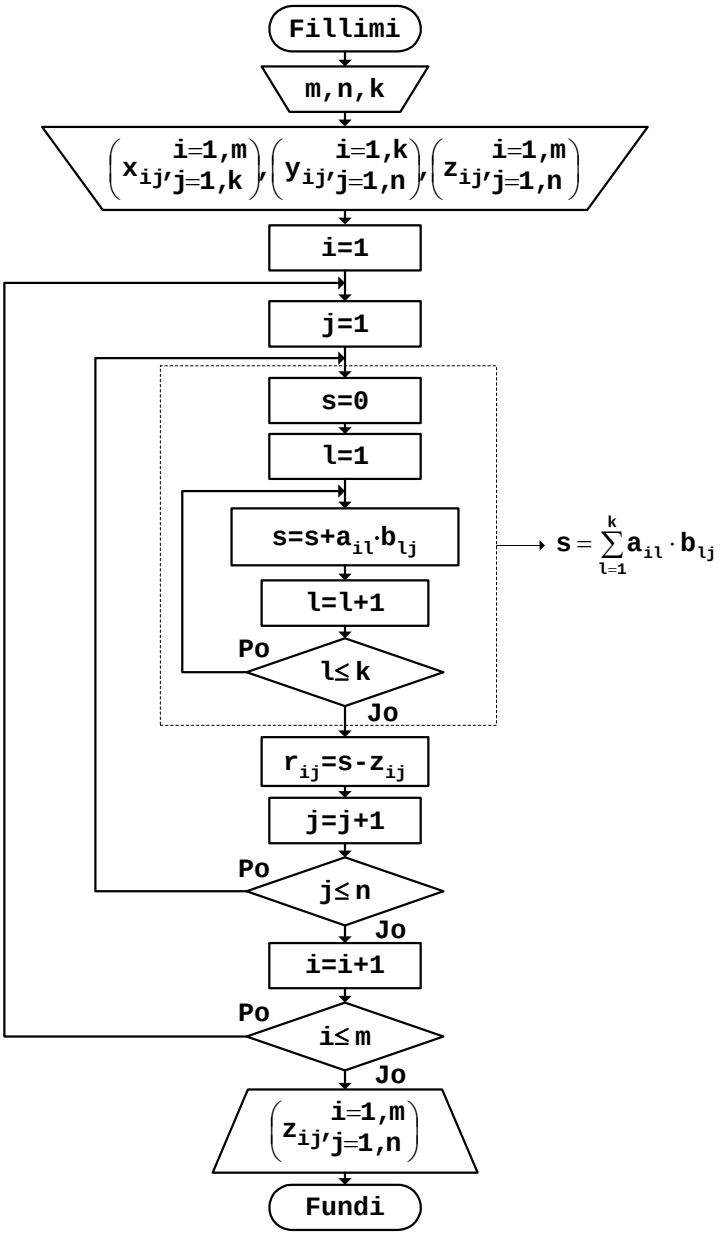


Fig.6.59

b. Programi

```
// Programi Prg6_59
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5,k=3; int i,j,l,s,R[m][n];
  int X[m][k]={ {3,5,8},
                {7,-4,9},
                {6,2,1},
                {2,4,6}};
  int Y[k][n]={ {-1,6,9,6,4},
                {2,6,8,4,-3},
                {5,3,-2,5,9}};
  int Z[m][n]={ {3,4,6,2,-1},
                {9,1,3,-5,2},
                {7,-2,1,4,8},
                {-2,5,3,1,4}};
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
      {
          s=0;
          for (l=0;l<k;l++)
              s=s+X[i][l]*Y[l][j];
          R[i][j]=s-Z[i][j];
      }
  cout << "Matrica e rezultatit R"
       << "\n";
  for (i=0;i<m;i++)
  {
      for (j=0;j<n;j++)
      {
          cout.width(4);
          cout << R[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Zgjidhja e ekuacionit do të shtypet në ekran si matricë:

```
Matrica e rezultatit R
  44  68  45  76  70
  21  44  10  76 119
  -4  53  67  45  19
  38  49  35  57  46
```

Anëtarët në vektor

Duke shfrytëzuar anëtarët e matricave, mund të formohen vektorë të ndryshëm. P.sh., me qëllim të sortimit sipas madhësisë të anëtarëve të matricës së dhënë, vlerat e tyre mund të vendosen në vektorë, p.sh., duke shkuar sipas rreshtave të matricës. Pastaj, sortimi i vlerave të anëtarëve të vektorit bëhet ashtu siç u tregua më parë.

Shembull Vendosja e anëtarëve të matricës `A(m,n)` në vektorin `Z(m·n)`.

a. Bllok-diagrami

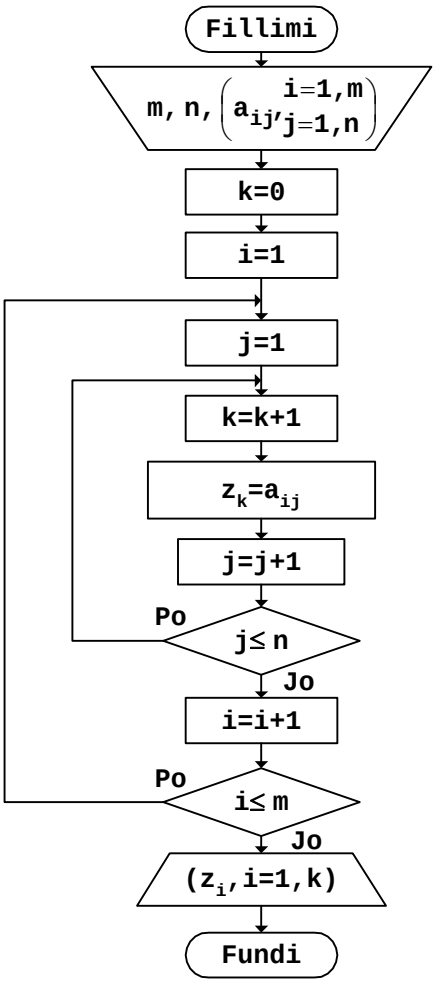


Fig.6.60

Këtu, numëratori k shfrytëzohet për përcaktimin e indeksit të pozitës në vektor të anëtarëve të veçantë të matricës.

b. Programi

```
// Programi Prg6_60
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=3;
  int A[m][n]={ {3,-4,6},
                {-9,1,2},
                {7,-8,1},
                {-2,5,-3}
              };
  int i,j,k,Z[m*n];
  k=-1;
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
      {
          k=k+1;
          Z[k]=A[i][j];
      }
  cout << "Z=[";
  for (i=0;i<=k;i++)
  {
      cout.width(3);
      cout << Z[i];
  }
  cout << "  ]\n";
  return 0;
}
```

Vektori që shtypet në ekran pas ekzekutimit të programit të dhënë është:

```
Z=[  3 -4  6 -9  1  2  7 -8  1 -2  5 -3  ]
```

Mbushja e vektorit mund të bëhet vetëm me anëtarë të matricës që kanë vlera të caktuara.

Shembull Mbushja e vektorit Z me anëtarët e matricës A(m,n) që kanë vlera negative.

Për caktimin e pozitave të anëtarëve që marrin pjesë në formimin e vektorit duhet të shfrytëzohet një numërator, p.sh. `k`, i cili në këtë rast rritet për `1` sa herë që gjendet anëtar me vlerë numerike negative. Vlera përfundimtare e numëratorit, vetëm nëse të gjithë anëtarët e matricës janë numra negativë, do të jetë e barabartë me numrin e anëtarëve të matricës.

a. Bllok-diagrami

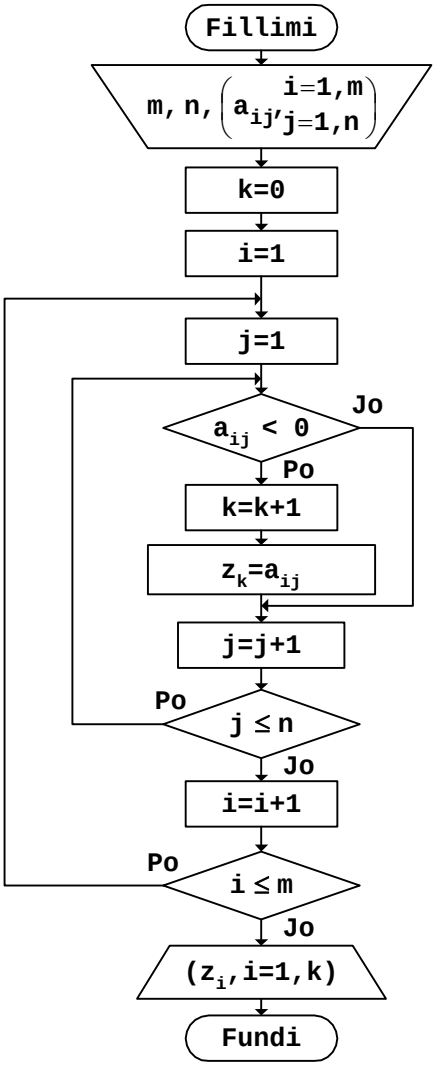


Fig.6.61

b. Programi

```
// Programi Prg6_61
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=3;
  int A[m][n]={ {3,-4,6},
                {-9,1,2},
                {7,-8,1},
                {-2,5,-3}
              };
  int i,j,k,Z[m*n];
  k=-1;
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          if (A[i][j]<0)
          {
              k=k+1;
              Z[k]=A[i][j];
          }
  cout << "Z=[";
  for (i=0;i<=k;i++)
  {
      cout.width(3);
      cout << Z[i];
  }
  cout << "  ]\n";
  return 0;
}
```

Rezultati që shtypet në ekran pas ekzekutimit të programit, për shembullin e vlerave të matricës `A`, do të duket kështu:

```
Z=[ -4 -9 -8 -2 -3 ]
```

Me vlerat e ndyshme të matricave mund të mbushen njëkohësisht disa vektorë.

Shembull Mbushja e vektorëve `F` dhe `G` me anëtarë pozitivë dhe negativë të matricës `A(m,n)`.

Këtu, për secilin vektor duhet të shfrytëzohet një numërator i veçantë i anëtarëve të matricës të cilët janë vendosur në vektorin përkatës.

a. Bllok-diagrami

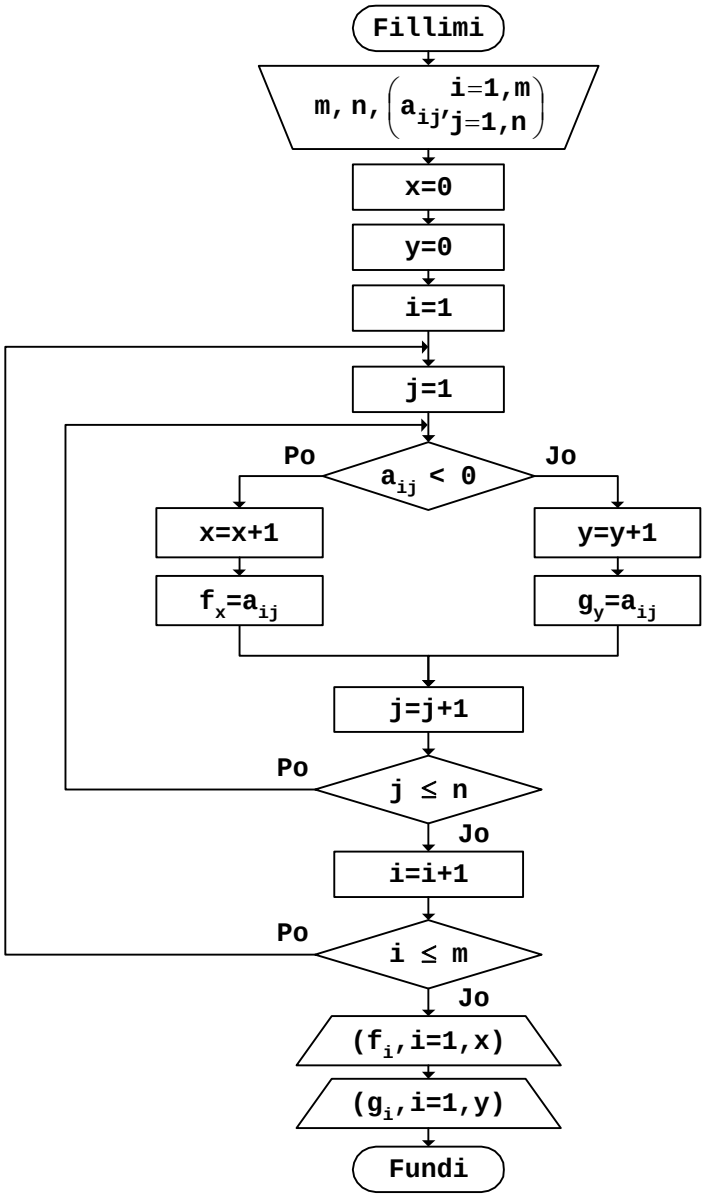


Fig.6.62

b. Programi

```
// Programi Prg6_62
#include <iostream>
using namespace std;
int main()
{
   int const m=4,n=3; int i,j,x,y,F[m*n],G[m*n];
   int A[m][n]={ {3,-4,6},
                 {-9,1,2},
                 {7,-8,1},
                 {-2,5,-3}};
   x=-1;y=-1;
   for (i=0;i<m;i++)
       for (j=0;j<n;j++)
           if (A[i][j]<0)
           {
               x=x+1; F[x]=A[i][j];
           }
           else
           {
               y=y+1; G[y]=A[i][j];
           }
   cout << "F=[";
   for (i=0;i<=x;i++)
   {
       cout.width(3);
       cout << F[i];
   }
   cout << "  ]\n";
   cout << "G=[";
   for (i=0;i<=y;i++)
   {
       cout.width(3);
       cout << G[i];
   }
   cout << "  ]\n";
   return 0;
}
```

Vektorët të cilët shtypen në ekran pas ekzekutimit të programit të dhënë duken kështu:

```
Z=[ -4 -9 -8 -2 -3 ]
G=[  3  6  1  2  7  1  5 ]
```

Vlerat e anëtarëve të matricave mund të vendosen në vektor edhe të ndryshuar në bazë të një ligjshmërie të caktuar.

Shembull Mbushja e vektorit F me katrorët e anëtarëve pozitivë të

matricës A(m,n), por të cilët janë më të vegjël se numri pozitiv x.

a. Bllok-diagrami

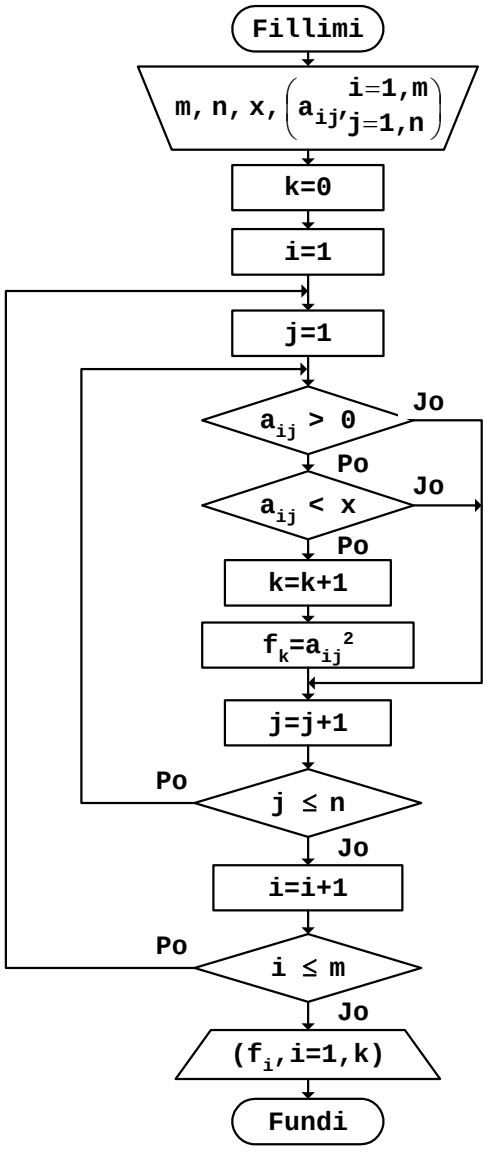


Fig.6.63

b. Programi

```
// Programi Prg6_63
#include <iostream>
using namespace std;
int main()
{
  int const x=6,m=4,n=3;
  int A[m][n]={ {3,-4,6},
                {-9,1,2},
                {7,-8,4},
                {-2,5,-3}
              };
  int i,j,k,F[m*n];
  k=-1;
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          if ((A[i][j]>=0) && (A[i][j]<x))
          {
             k=k+1;
             F[k]=A[i][j]*A[i][j];
          }
  cout << "F=[";
  for (i=0;i<=k;i++)
  {
      cout.width(3);
      cout << F[i];
  }
  cout << "  ]\n";
  return 0;
}
```

Rezultati i programit të dhënë në ekran shtypet kështu:

```
F=[  9  1  4 16 25 ]
```

Vektorët mund të mbushen edhe duke i shfrytëzuar anëtarët e matricës të cilët gjenden në rreshtat ose në kolonat e caktuara të matricës.

Shembull Mbushja e vektorit `F` me sinuset e anëtarëve pozitivë të matricës `A(m,n)`, por të cilët gjenden në rreshtat çiftë.

a. Bllok-diagrami

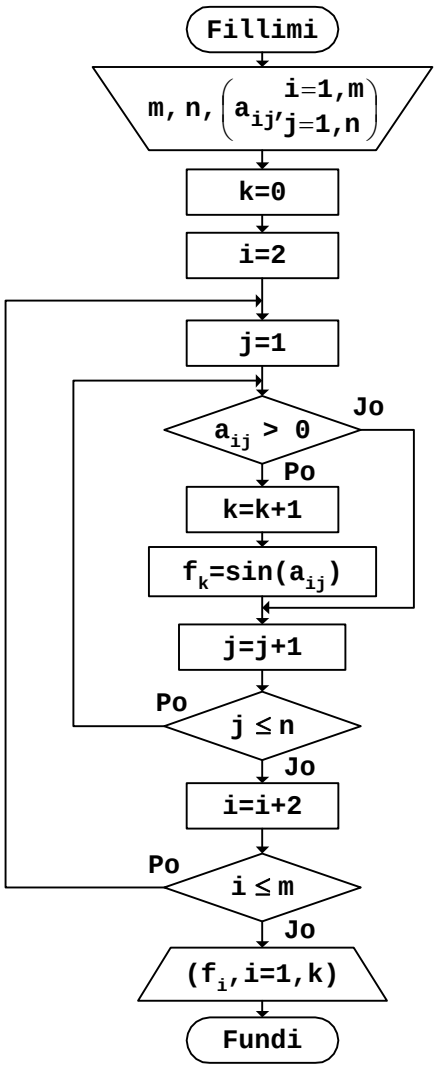


Fig.6.64

b. Programi

```
// Programi Prg6_64
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   int const m=4,n=3;
   int A[m][n]={ {3,-5,6},
```

```
                {-9,1,2},
                {7,-8,4},
                {-2,5,-3}
            };
  int i,j,k;
  double F[m*n];
  k=-1;
  i=1;
  do
  {
      for (j=0;j<n;j++)
          if (A[i][j]>=0)
          {
              k=k+1;
              F[k]=sin(A[i][j]);
          }
      i=i+2;
  }
  while (i<=m);
  cout << "F=[";
  for (i=0;i<=k;i++)
  {
      cout.precision(3);
      cout.width(7);
      cout << F[i];
  }
  cout << "  ]\n";
  return 0;
}
```

Vektori që formohet përmes programit të dhënë, në ekran do të shtypet kështu:

```
F=[  0.841  0.909 -0.959 ]
```

ku tre anëtarët e vektorit në fakt janë sinuset e vlerave numerike 1, 2 dhe 5, të anëtarëve pozitivë të matricës A të cilët gjenden në rreshtat çiftë.

Veprime të tjera

Përveç veprimeve të përmendura më sipër, për pjesë të caktuara të matricës, siç janë rreshtat, kolonat, pjesa mbi, në ose nën diagonalen kryesore të matricës, ose për komplet matricën, mund të gjendet shuma, prodhimi, të numërohen ose të gjenden anëtarë të caktuar etj.

Shuma dhe prodhimi

Procedura e llogaritjes së shumës ose e prodhimit të anëtarëve të matricës nuk dallon nga ajo që zbatohet te vektorët, me përjashtim të asaj se këtu operohet me dy indekse.

Shembull Llogaritja e shumës së anëtarëve të matricës `A(m,n)`.

$$s = a_{11} + a_{12} + \ldots + a_{1n} + a_{21} + a_{22} + \ldots + a_{2n} + \ldots\ldots\ldots\ldots\ldots + a_{m1} + a_{m2} + \ldots + a_{mn} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}$$

a. Bllok-diagrami

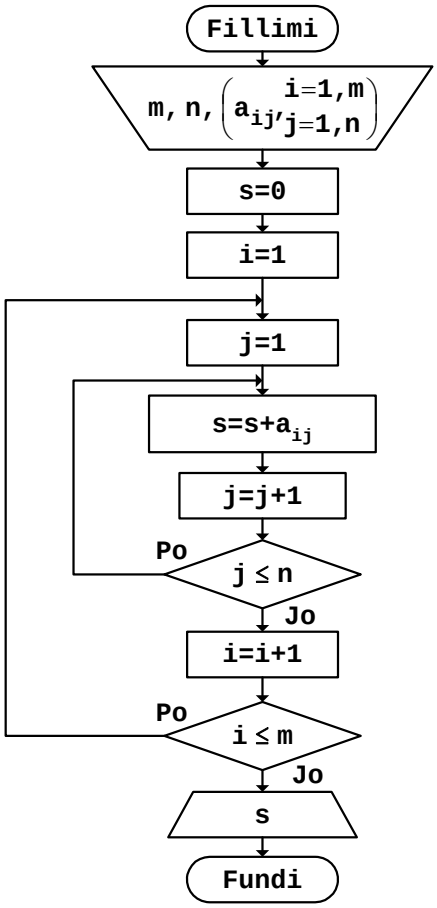


Fig.6.65

b. Programi

```
// Programi Prg6_65
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,6,7,2},
                {-9,1,2,5,8},
                {7,-8,4,3,9},
                {6,5,-3,4,8}
              };
  int i,j,s;
  s=0;
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          s=s+A[i][j];
  cout << "Shuma s="
       << s
       << "\n";
  return 0;
}
```

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

```
Shuma s=55
```

dhe numri i fituar e paraqet shumën e të gjithë anëtarëve të matricës.

Për matricën e dhënë mund të llogaritet shuma ose prodhimi i pjesëve të caktuara të matricës, gjë që përcaktohet përmes zgjedhjes së indekseve përkatëse.

Shembull Llogaritja e shumës s të anëtarëve në rreshtat e matricës `A(m,n)`.

a. Bllok-diagrami

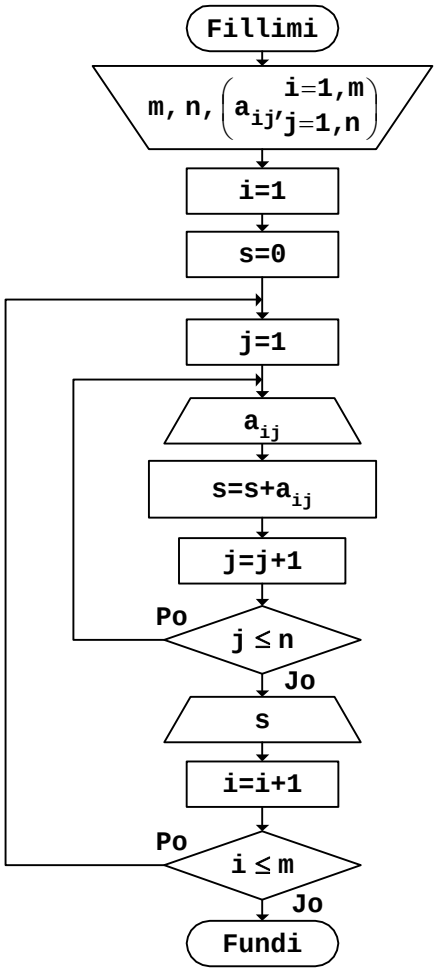


Fig.6.66

b. Programi

```
// Programi Prg6_66
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,6,7,2},
                {-9,1,2,5,8},
                {7,-8,4,3,9},
```

```
                {6,5,-3,4,8}
            };
  int i,j,s;
  for (i=0;i<m;i++)
  {
      s=0;
      for (j=0;j<n;j++)
      {
          cout.width(4);
          cout << A[i][j];
          s=s+A[i][j];
      }
      cout << "   s="
           << s
           << "\n";
  }
  return 0;
}
```

Nëse ekzekutohet programi i dhënë, për vlerat e matricës së marrë si shembull, rezultati që shtypet në ekran është:

```
  3 -5  6  7  2   s=13
 -9  1  2  5  8   s=7
  7 -8  4  3  9   s=15
  6  5 -3  4  8   s=20
```

Njëkohësisht mund të llogariten edhe më shumë vlera për pjesë të caktuara të matricës, siç janë, p.sh., anëtarët mbi diagonalë, në diagonalë ose nën diagonalë.

Shembull Llogaritja e shumës `s` të anëtarëve mbi diagonalen kryesore dhe prodhimit `p` të anëtarëve në diagonalen kryesore të matricës `A(m,m)`.

a. Bllok-diagrami

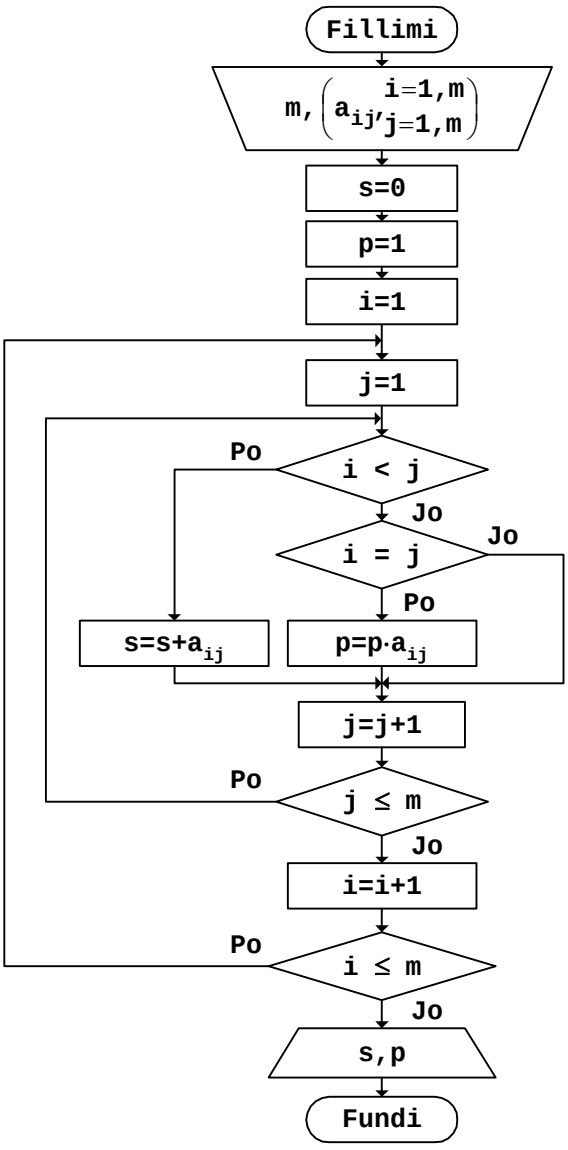


Fig.6.67

b. Programi

```
// Programi Prg6_67
#include <iostream>
using namespace std;
int main()
{
  int const m=5;
  int A[m][m]={ {4,3,5,-7,1},
                {-5,6,4,9,2},
                {3,-4,7,6,1},
                {8,3,-2,5,9},
                {6,4,8,-3,7}
              };
  int i,j,s,p;
  s=0;
  p=1;
  for (i=0;i<m;i++)
      for (j=0;j<m;j++)
          if (i<j)
             s=s+A[i][j];
          else
              if (i==j)
                  p=p*A[i][j];
  cout << "Shuma    s="
       << s
       << "\n"
       << "Prodhimi p="
       << p
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit, për vlerat e anëtarëve të matricës, e cila është marrë si shembull, rezultati që shtypet në ekran është:

```
Shuma    s=33
Prodhimi p=5880
```

Numërimi i anëtarëve të caktuar

Brenda një matrice mund të numërohen anëtarë të caktuar të matricës, p.sh., anëtarët negativë, pozitivë, më të mëdhenj ose më të vegjël se vlera të dhëna etj.

Shembull Numërimi i anëtarëve pozitivë `x`, negativë `y` dhe zero `z`, në matricën e dhënë `A(m,n)`.

a. Bllok-diagrami

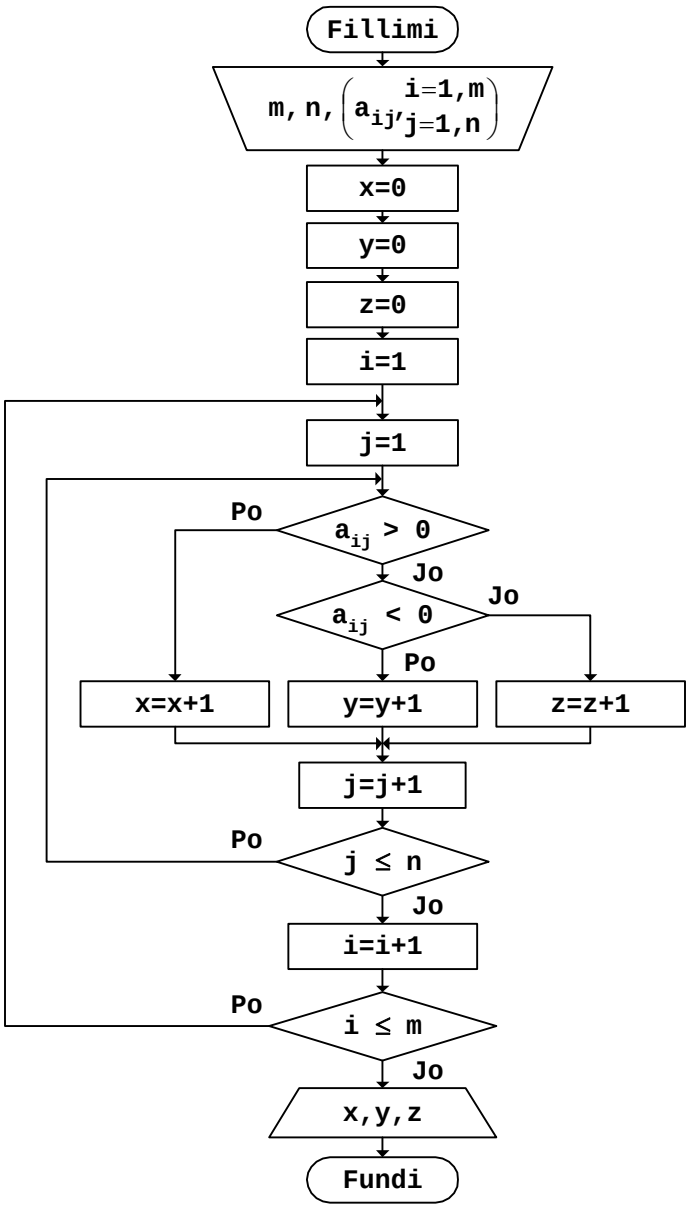


Fig.6.68

b. Programi

```
// Programi Prg6_68
#include <iostream>
```

```
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,6,7,2},
                {-9,1,0,-7,8},
                {0,-8,4,3,9},
                {6,5,-3,0,8}
              };
  int i,j,x,y,z;
  x=0;y=0;z=0;
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          if (A[i][j]>0)
              x=x+1;
          else
              if (A[i][j]<0)
                  y=y+1;
              else
                  z=z+1;
  cout << "Anëtarë pozitivë x="
       << x
       << "\n"
       << "Anëtarë negativë y="
       << y
       << "\n"
       << "Anëtarë zero     z="
       << z
       << "\n";
  return 0;
}
```

Për shembullin e matricës së marrë në program, pas ekzekutimit të programit, numrat e kërkuar do të shtypen kështu:

```
Anëtarë pozitivë x=12
Anëtarë negativë y=5
Anëtarë zero     z=3
```

Numërimi i anëtarëve mund të zbatohet edhe në pjesë të veçanta të matricës.

Shembull Numërimi i anëtarëve pozitivë `x` dhe negativë `y` në rreshtat e veçantë të matricës `A(m,n)`, pa anëtarët me vlerë zero.

a. Bllok-diagrami

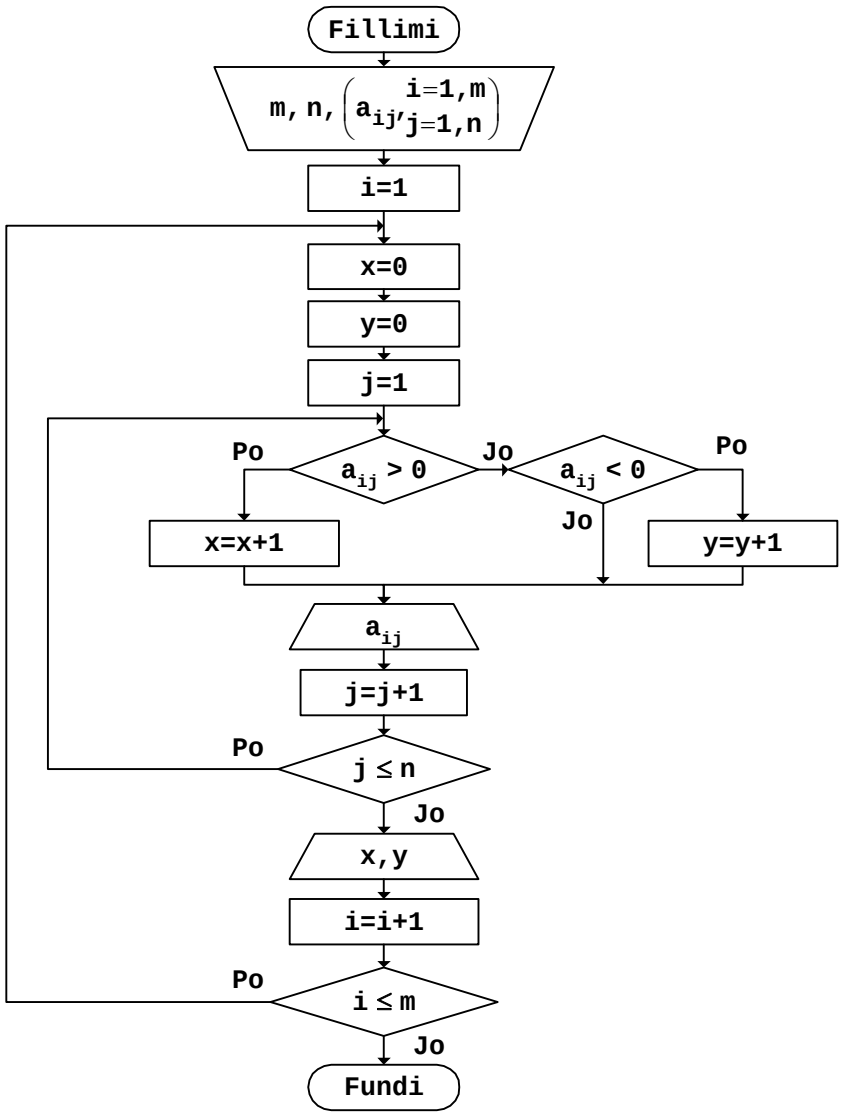


Fig.6.69

b. Programi

```
// Programi Prg6_69
#include <iostream>
using namespace std;
```

```
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {1,-8,-5,7,2},
                {-6,0,-4,3,-8},
                {4,-8,0,3,2},
                {-1,0,-3,9,0}
              };
  int i,j,x,y;
  for (i=0;i<m;i++)
  {
      x=0;
      y=0;
      for (j=0;j<n;j++)
      {
          if (A[i][j]>0)
              x=x+1;
          else
              if (A[i][j]<0)
                  y=y+1;
          cout.width(3);
          cout << A[i][j];
      }
    cout << "   x="
         << x
         << "  y="
         << y
         << "\n";
  }
  return 0;
}
```

Për shembullin e matricës së marrë në program, rezultati që shtypet në ekran është:

```
  1 -8 -5  7  2   x=3  y=2
 -6  0 -4  3 -8   x=1  y=3
  4 -8  0  3  2   x=3  y=1
 -1  0 -3  9  0   x=1  y=2
```

Anëtarët e caktuar mund të numërohen edhe vetëm në një pjesë të matricës.

Shembull Numërimi i anëtarëve nën diagonalen kryesore të matricës `A(m,m)`, të cilët janë më të mëdhenj se `2` dhe më të vegjël se `7`.

a. Bllok-diagrami

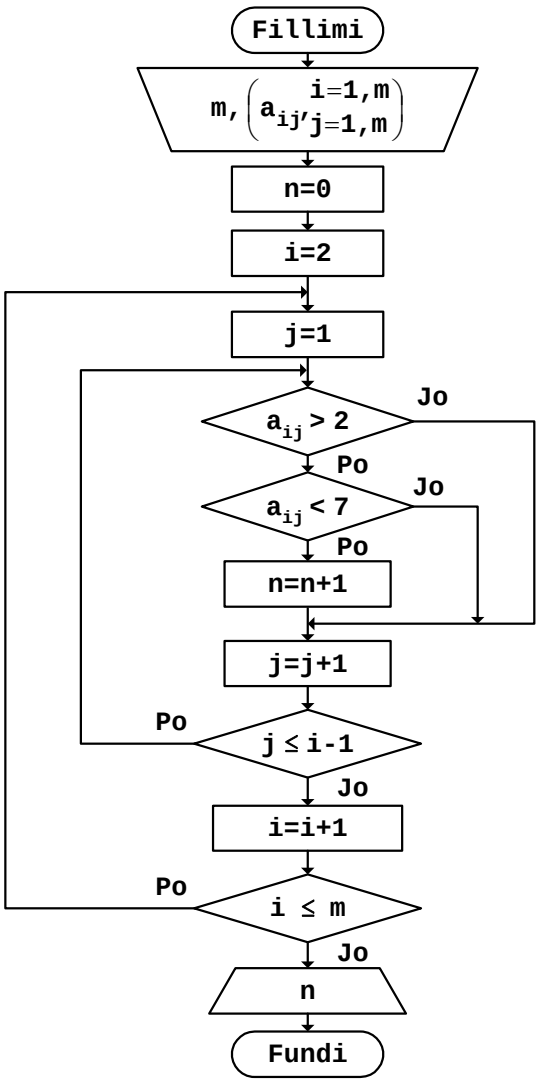


Fig.6.70

b. Programi

```
// Programi Prg6_70
#include <iostream>
using namespace std;
```

```
int main()
{
  int const m=5;
  int A[m][m]={ { 4, 3, 5,-7, 1},
                {-5, 6, 4, 9, 2},
                { 3, 9, 7, 6, 1},
                { 1,-3,-2, 5, 9},
                { 6, 5, 8, 4, 7}
              };
  int i,j,n;
  n=0;
  for (i=1;i<m;i++)
      for (j=0;j<=i-1;j++)
      {
          if ((A[i][j]>2) && (A[i][j]<7))
             n=n+1;
      }
  cout << "Numri i kërkuar n="
       << n
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit, për shembullin e matricës së marrë, rezultati që shtypet në ekran është:

```
Numri i kërkuar n=4
```

Gjetja e vlerës së caktuar

Plotësisht njëlloj si te vektorët, edhe te matricat mund të gjendet anëtari i caktuar, siç është, p.sh., anëtari minimal, ose anëtari maksimal etj. Procedura e gjetjes edhe këtu fillon me përvetësimin e një vlere fillestare të një variable ndihmëse, tek e cila në fund merret edhe rezultati i kërkuar. Si vlerë fillestare rregullisht merret anëtari i parë në matricë, ose kjo vlerë formohet nga ky anëtar i matricës.

Shembull Gjetja e vlerës minimale `z` në matricën `A(m,n)`.

a. Bllok-diagrami

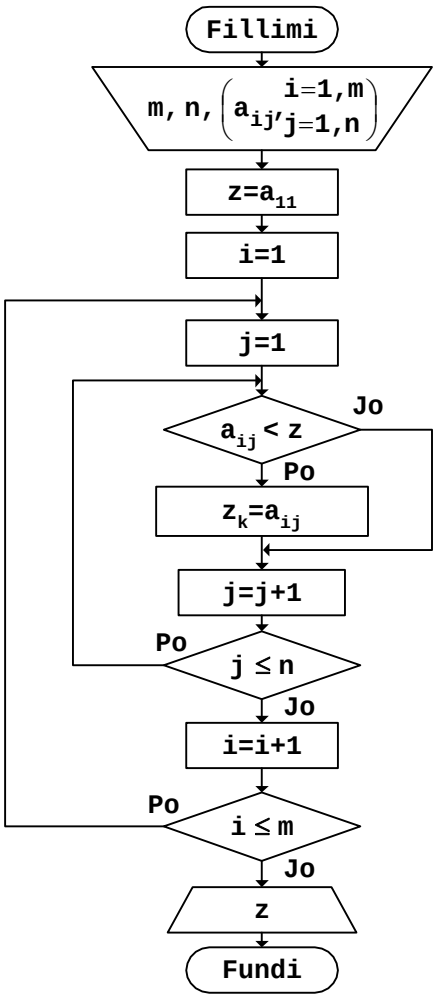


Fig.6.71

Këtu, për `i=1` dhe `j=1` do të krahasohet anëtari i parë i matricës me vetveten, gjë që imponohet si krahasim i domosdoshëm, meqë nëse merret `i=1` dhe `j=2`, për vlerat e tjera të variablës `i`, gjatë krahasimit do të kapërcehen anëtarët e kolonës së parë të matricës.

b. Programi

```
// Programi Prg6_71
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,6,7,2},
                {-9,1,2,5,8},
                {7,-8,4,3,9},
                {6,5,-3,4,8}
              };
  int i,j,z;
  z=A[0][0];
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          if (A[i][j]<z)
             z=A[i][j];
  cout << "Vlera minimale z="
       << z
       << "\n";
  return 0;
}
```

Rezultati që shtypet në ekran pas ekzekutimit të programit është:

```
Vlera minimale z=-9
```

Në matricë, përveç anëtarit të caktuar, mund të gjendet edhe pozita e tij, përkatësisht indekset përkatëse.

Shembull Anëtari më i madh për nga vlera absolute `x` si dhe pozita e tij, përkatësisht indekset `f` dhe `g`, në matricën e dhënë `A(m,n)`.

a. Bllok-diagrami

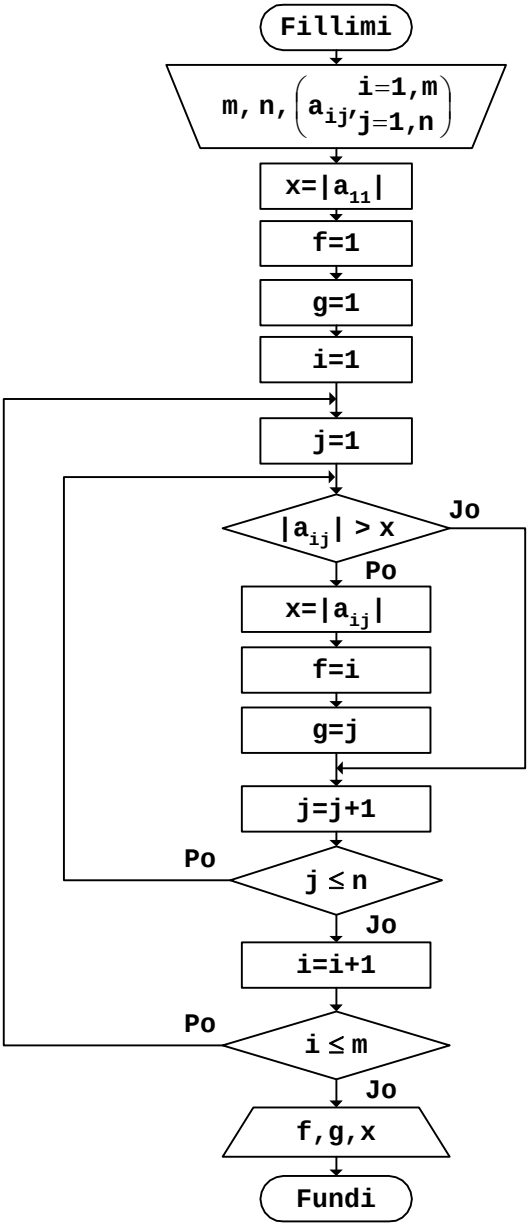


Fig.6.72

b. Programi

```
// Programi Prg6_72
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,8,7,2},
                {-4,1,2,5,8},
                {7,-9,4,3,2},
                {6,5,-3,4,7}
              };
  int i,j,f,g,x;
  x=abs(A[0][0]);
  f=1;g=1;
  for (i=0;i<m;i++)
      for (j=0;j<n;j++)
          if (abs(A[i][j])>x)
          {
             x=abs(A[i][j]);
             f=i;
             g=j;
          }
  cout << "Vlera maksimale absolute x: "
       << x
       << "\n"
       << "Pozita ............... f,g: "
       << f
       << ","
       << g
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit, për shembullin e matricës së marrë në program, rezultati që shtypet në ekran është:

```
Vlera maksimale absolute x: 9
Pozita ............... f,g: 2,1
```

Anëtari i caktuar mund të gjendet edhe për çdo rresht ose kolonë, si dhe për pjesë të caktuar të matricës.

Shembull Anëtari më i madh për çdo rresht të matricës `A(m,n)`.

a. Bllok-diagrami

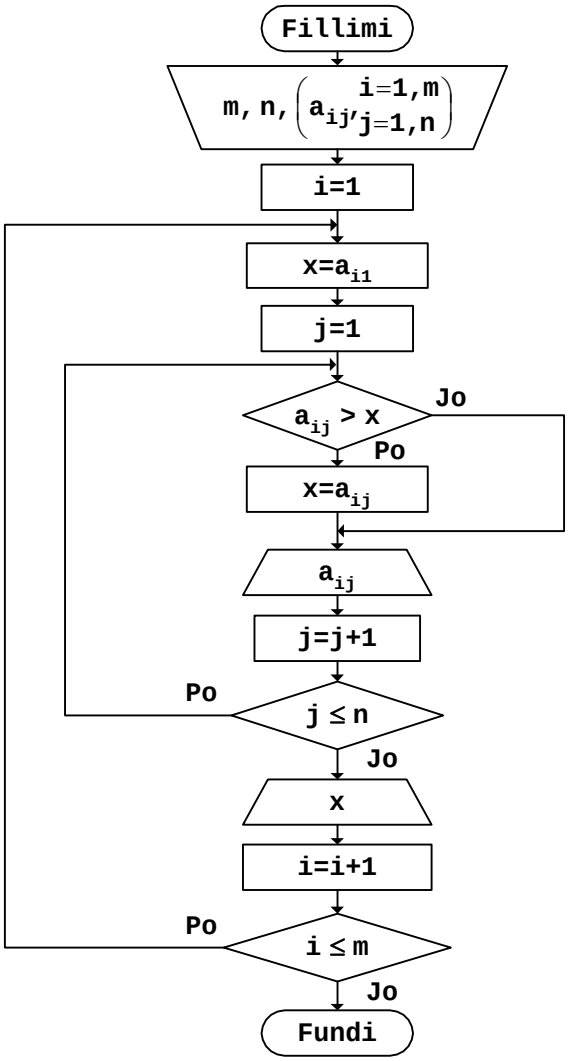


Fig.6.73

b. Programi

```
// Programi Prg6_73
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,8,7,2},
                {-4,1,2,6,-3},
                {7,-9,4,3,2},
                {9,5,-3,4,1}
              };
  int i,j,x;
  for (i=0;i<m;i++)
  {
      x=A[i][1];
      for (j=0;j<n;j++)
      {
          if (A[i][j]>x)
             x=A[i][j];
          cout.width(3);
          cout << A[i][j];
      }
      cout << "   x="
           << x
           << "\n";
  }
  return 0;
}
```

Pas ekzekutimit të programit, rezultati në ekran duket kështu:

```
  3 -5  8  7  2   x=8
 -4  1  2  6 -3   x=6
  7 -9  4  3  2   x=7
  9  5 -3  4  1   x=9
```

Vlera e caktuar mund të gjendet edhe vetëm në një pjesë të matricës.

Shembull Anëtari më i madh mbi diagonalën kryesore `x` dhe nën diagonalen kryesore `y`, të matricës së dhënë `A(m,n)`.

a. Bllok-diagrami

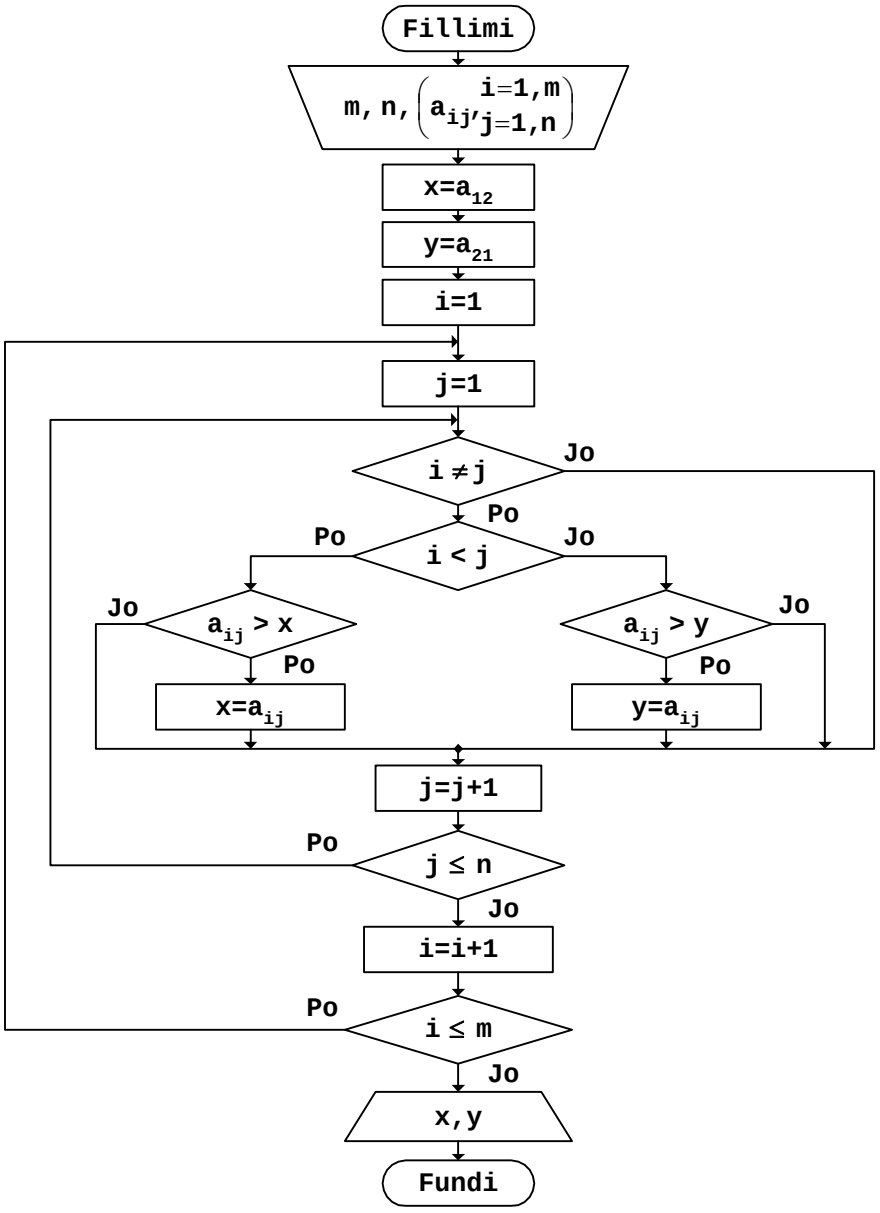


Fig.6.74

b. Programi

```
// Programi Prg6_74
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {3,-5,8,7,2},
                {-4,1,2,6,-3},
                {7,-9,6,3,4},
                {9,5,-3,4,1}
              };
  int i,j,x,y;
  x=A[0][1];
  y=A[1][0];
  for (i=0;i<m;i++)
    for (j=0;j<n;j++)
        if (i!=j)
            if (i<j)
               if (A[i][j]>x)
                  x=A[i][j];
               else
                  {
                  }
            else
               if (A[i][j]>y)
                  y=A[i][j];
               else
                  {
                  }
  cout << "Mbi diagonale x="
       << x
       << "\n"
       << "Nën diagonale y="
       << y
       << "\n";
  return 0;
}
```

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

```
Mbi diagonale x=8
Nën diagonale y=9
```

Detyra Në matricën e dhënë `A(m,n)`, të gjendet:

a. anëtari me vlerë numerike më të madhe;

b. anëtari me vlerë absolute më të vogël si dhe pozita e tij në matricë;

c. anëtari më i vogël për çdo kolonë;

d. anëtari më i madh në çdo rresht tek;

e. anëtari më i vogël për nga vlera absolute në kolonat çiftë të matricës;

f. anëtari më i madh për çdo rresht, duke marrë vetëm vlerat të cilat gjenden mbi diagonalen kryesore.

Në matricë mund të gjenden edhe vlerat mesatare të anëtarëve të matricës, anëtarëve në rreshtat ose në kolonat e veçanta, anëtarëve mbi, në ose nën diagonalen kryesore etj.

Shembull Vlerat mesatare `x` të rreshtave të veçantë të matricës `A(m,n)`.

$$x = \frac{1}{n} \sum_{j=1}^{n} a_{ij} \text{ , për } i = 1,2,\ldots,m.$$

a. Bllok-diagrami

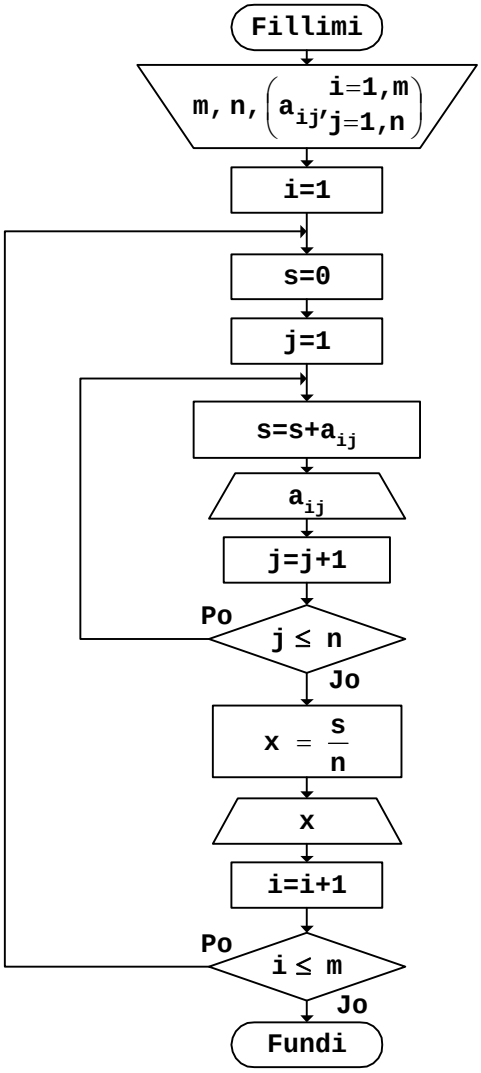


Fig.6.75

b. Programi

```
// Programi Prg6_75
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5;
  int A[m][n]={ {7,3,4,9,5},
                {6,1,2,4,8},
                {7,6,4,1,2},
                {9,5,3,8,7}
              };
  int i,j;
  double s,x;
  for (i=0;i<m;i++)
  {
      s=0;
      for (j=0;j<n;j++)
      {
          s=s+A[i][j];
          cout.width(3);
          cout << A[i][j];
      }
      x=s/n;
      cout << "   x="
           << x
           << "\n";
  }
  return 0;
}
```

Rezultati i programit të dhënë, pas ekzekutimit të tij, në ekran do të shtypet kështu:

```
  7  3  4  9  5   x=5.6
  6  1  2  4  8   x=4.2
  7  6  4  1  2   x=4
  9  5  3  8  7   x=6.4
```

Detyra Për matricën `A(m,m)`, të gjendet vlera mesatare:

a. e të gjithë anëtarëve;

b. e anëtarëve pozitivë;

c. e vlerave absolute të anëtarëve negativë;

d. e anëtarëve të diagonales kryesore;

e. e anëtarëve mbi diagonalen kryesore;

f. e anëtarëve nën diagonalen kryesore;

g. e kolonave të veçanta.

Fshirja e rreshtave dhe e kolonave

Gjatë zgjidhjes së problemeve të ndryshme, duke shfrytëzuar matricat, shpesh herë në matrica duhet fshirë rreshta, kolona, ose rreshta e kolona njëkohësisht. Problemi kryesor këtu qëndron në zhvendosjen lart të rreshtave që gjenden pas rreshtit që fshihet, përkatësisht zhvendosjen majtë të kolonave që gjenden pas kolonës që fshihet.

Shembull Formimi i matricës `B(m-1,n)` nga anëtarët e matricës `A(m,n)`, pasi në të të fshihet rreshti i `l`-të.

a. Bllok-diagrami

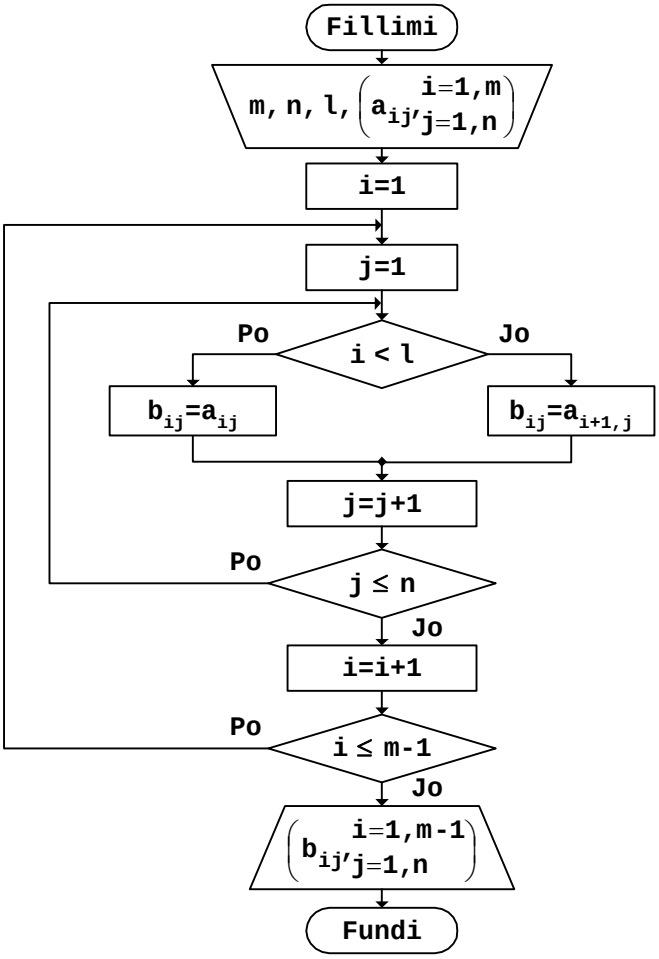


Fig.6.76

b. Programi

```
// Programi Prg6_76
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5,l=2;
  int A[m][n]={ {7,-3,4,-9,5},
                {6,1,-2,4,-8},
                {-7,6,-4,1,2},
```

```
                {9,-5,3,8,-6}
              };
  int i,j,B[m][n];
  for (i=0;i<m-1;i++)
      for (j=0;j<n;j++)
          if (i<1)
              B[i][j]=A[i][j];
          else
              B[i][j]=A[i+1][j];
  cout << "Matrica B"
       << "\n";
  for (i=0;i<m-1;i++)
  {
      for (j=0;j<n;j++)
      {
          cout.width(3);
          cout << B[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Nëse ekzekutohet programi i dhënë, pas fshirjes së rreshtit të dytë në matricën `A`, matrica që fitohet do të duket kështu:

```
  Matrica B
  7 -3  4 -9  5
 -7  6 -4  1  2
  9 -5  3  8 -6
```

Plotësisht njëlloj veprohet edhe gjatë fshirjes së kolonës në matricë.

Shembull Formimi i matricës `B(m,n-1)` nga anëtarët e matricës `A(m,n)`, pasi në të të fshihet kolona e `k`-të.

a. Bllok-diagrami

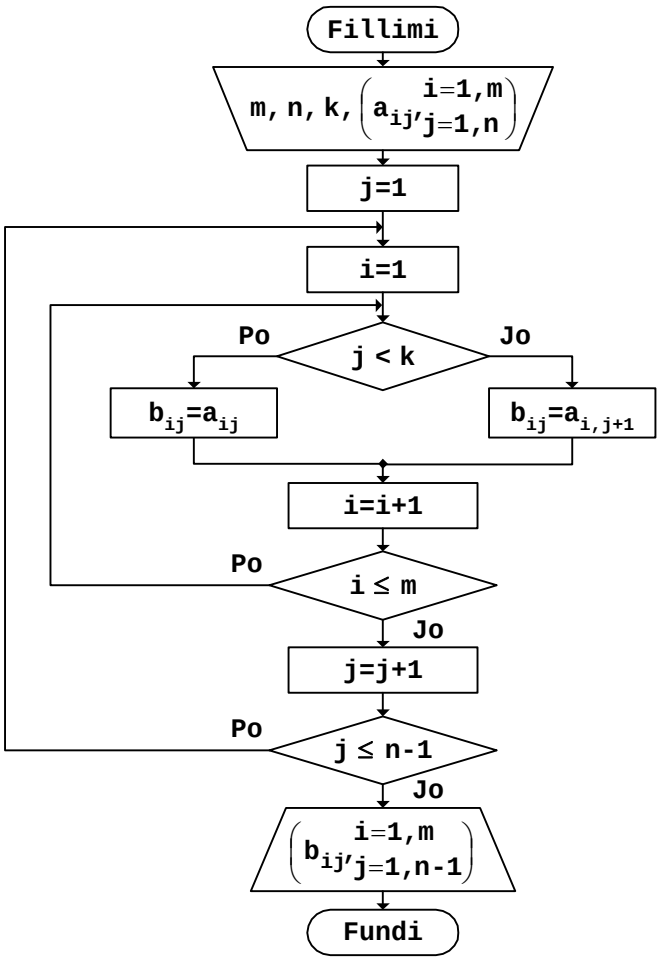


Fig.6.77

b. Programi

```
// Programi Prg6_77
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5,k=2;
  int A[m][n]={ {7,-3,4,-9,5},
                {6,1,-2,4,-8},
                {-7,6,-4,1,2},
                {9,-5,3,8,-6}
              };
```

```
  int i,j,B[m][n];
  for (j=0;j<n-1;j++)
      for (i=0;i<m;i++)
          if (j<k)
             B[i][j]=A[i][j];
          else
             B[i][j]=A[i][j+1];
  cout << "Matrica B"
       << "\n";
  for (i=0;i<m;i++)
  {
      for (j=0;j<n-1;j++)
      {
          cout.width(3);
          cout << B[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Matrica që formohet pas fshirjes së kolonës `k=2` përmes programit të dhënë më sipër, në ekran do të shtypet kështu:

```
  Matrica B
  7 -3 -9  5
  6  1  4 -8
 -7  6  1  2
  9 -5  8 -6
```

Nëse fshihen njëkohësisht rreshti dhe kolona e caktuar në matricën e dhënë, duhet të zhvendosen rreshtat dhe kolonat të cilat gjenden pas atyre që fshihen.

Shembull Formimi i matricës `B(m-1,n-1)` nga anëtarët e matricës `A(m,n)`, pasi në të të fshihet rreshti i `l`-të dhe kolona e `k`-të.

a. Bllok-diagrami

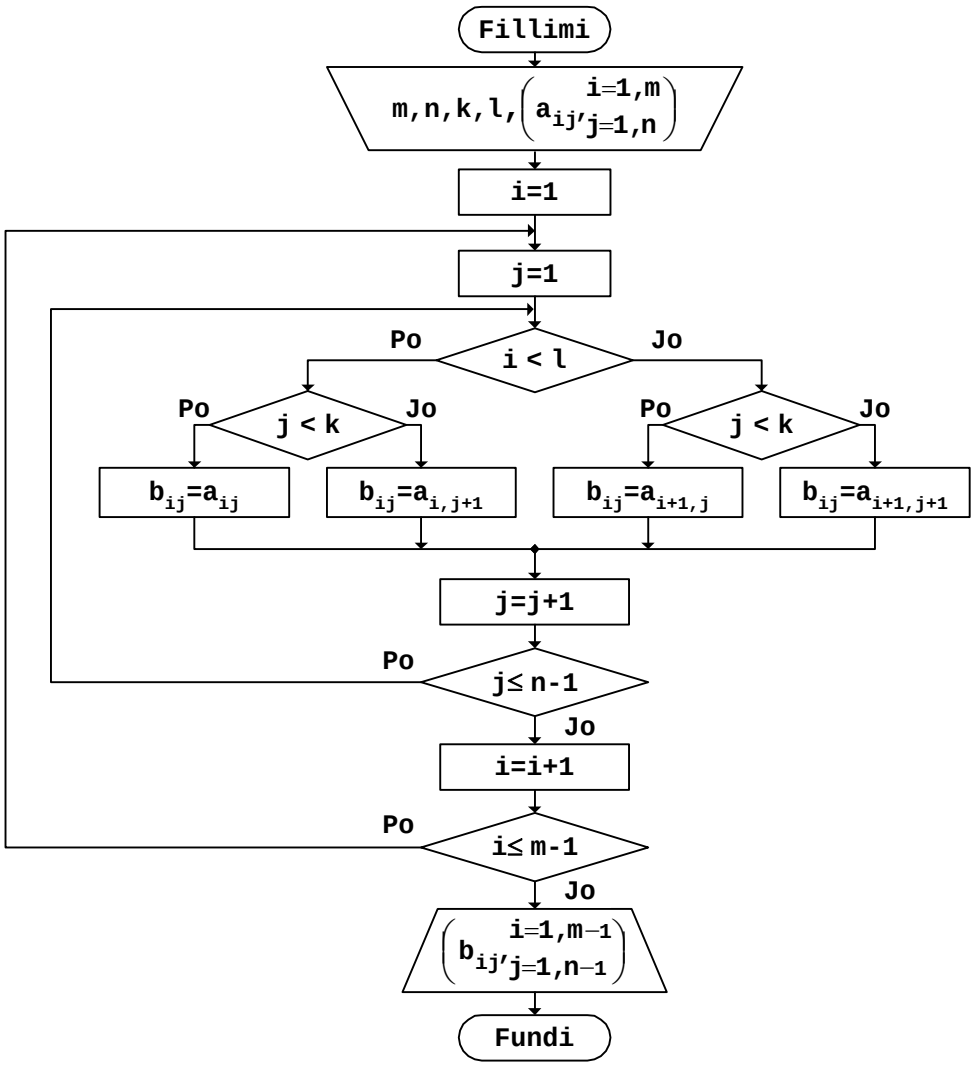


Fig.6.78

b. Programi

```
// Programi Prg6_78
#include <iostream>
using namespace std;
int main()
{ int const m=4,n=5,l=1,k=2;
  int A[m][n]={ {7,-3,4,-9,5},
                {6,1,-2,4,-8},
                {-7,6,-4,1,2},
                {9,-5,3,8,-6}
              };
  int i,j,B[m-1][n-1];
  for (i=0;i<n-1;i++)
      for (j=0;j<m;j++)
          if (i<1)
              if (j<k)
                  B[i][j]=A[i][j];
              else
                  B[i][j]=A[i][j+1];
          else
              if (j<k)
                  B[i][j]=A[i+1][j];
              else
                  B[i][j]=A[i+1][j+1];
  cout << "Matrica B"
       << "\n";
  for (i=0;i<m-1;i++)
  {
      for (j=0;j<n-1;j++)
      {
          cout.width(3);
          cout << B[i][j];
      }
      cout << "\n";
  }
  return 0;
}
```

Nëse ekzekutohet programi i dhënë, matrica e cila fitohet pas fshirjes së rreshtit `l=1` dhe `k=2` të matricës `A`, duket kështu:

```
  Matrica B
  7 -3 -9  5
 -7  6  1  2
  9 -5  8 -6
```

Algoritmet për fshirjen e njëkohshme të më shumë rreshtave, ose të më shumë kolonave në matricë, do të jenë të ngjashme me ato që u dhanë më sipër.

Detyra

Formimi i matricës `B` nga anëtarët e matricës `A(m,n)`, pasi në të të fshihen:

a. `x`-rreshta, duke filluar prej rreshtit të `l`-të;

b. `y`-kolona, duke filluar prej kolonës së `k`-të;

c. `x`-rreshta dhe `y`-kolona, duke filluar prej rreshtit të `l`-të dhe kolonës së `k`-të.

Këtu, para se të fillojë fshirja duhet të kontrollohet se a ka për fshirje `x`-rreshta dhe `y`-kolona, prej rreshtit të `l`-të dhe kolonës së `k`-të.

Fushat tridimensionale

Gjatë zgjidhjes së problemeve me kompjuter përdoren edhe fusha shumëdimensionale. Kështu, p.sh., operimi me të dhëna, të cilat përcaktohen përmes tri madhësive të ndryshme, thjeshtohet dukshëm, nëse shfrytëzohen fusha tridimensionale.

Në algoritme, me fushat tridimensionale operohet duke i shikuar ato si grumbuj matricash, ku me dimensionin e tretë të tyre përcaktohet numri i matricës në strukturën tridimensionale të fushës, ashtu siç shihet në `Fig.6.79`.

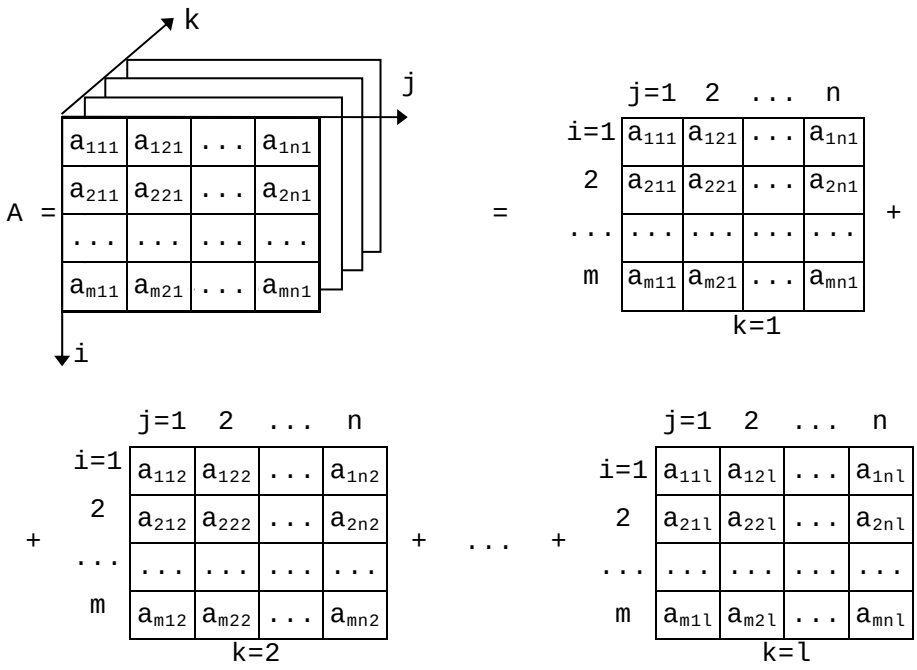


Fig.6.79 Fusha tridimensionale

Shembull Formimi i fushës tridimensionale `C(m,n,2)`, nga anëtarët e matricave `A(m,n)` dhe `B(m,n)`.

a. Bllok-diagrami

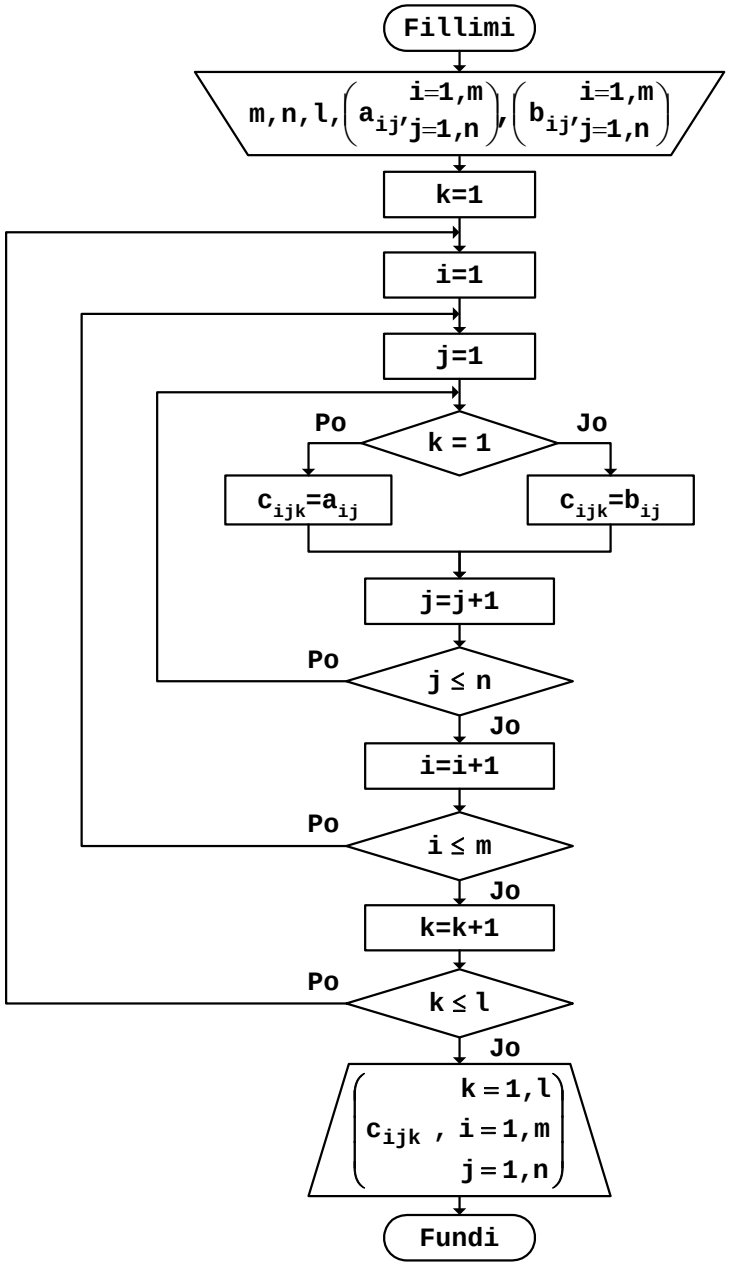


Fig.6.80

b. Programi

```
// Programi Prg6_80
#include <iostream>
using namespace std;
int main()
{
  int const m=4,n=5,l=2;
  int A[m][n]={ { 4,-2, 3, 7, 1},
                { 2, 5,-9, 1,-5},
                { 6, 4, 2, 9, 3},
                { 6,-2, 1, 8,-4}
              };
  int B[m][n]={ { 1, 3,-4,-6, 2},
                { 6, 5,-2, 4,-8},
                {-2, 3,-7, 1, 8},
                { 9,-6, 2, 1, 7}
              };
  int i,j,k,C[m][n][l];
  for (k=0;k<l;k++)
      for (i=0;i<m;i++)
          for (j=0;j<n;j++)
              if (k==0)
                  C[i][j][k]=A[i][j];
              else
                  C[i][j][k]=B[i][j];

  cout << "   Fusha C"
       << "\n";
  for (k=0;k<l;k++)
  {
      cout << "  k="
           << k
           << "\n";
      for (i=0;i<m;i++)
      {
          for (j=0;j<n;j++)
          {
              cout.width(3);
              cout << C[i][j][k];
          }
          cout << "\n";
      }
      cout << "\n";
  }
  return 0;
```

```
}
```

Fusha tridimensionale, e cila formohet përmes programit të dhënë, në ekran do të shtypet kështu:

```
Fusha C

k=0
4 -2  3  7  1
2  5 -9  1 -5
6  4  2  9  3
6 -2  1  8 -4

k=1
1  3 -4 -6  2
6  5 -2  4 -8
-2  3 -7  1  8
9 -6  2  1  7
```

Plotësisht njëlloj si edhe gjatë punës me matrica, mund të operohet edhe me anëtarët e fushave tridimensionale.

Shembull Llogaritja e shumës `s` të katrorëve të anëtarëve negativë të fushës tridimensionale `A(m,n,l)`.

a. Bllok-diagrami

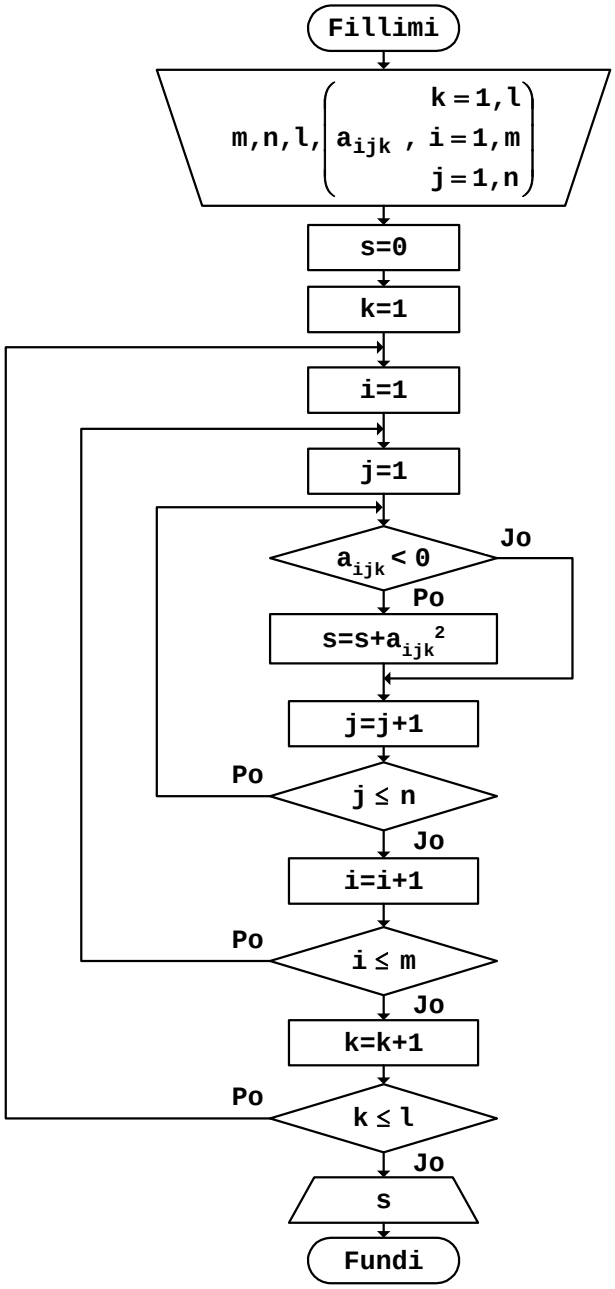


Fig.6.81

b. Programi

```
// Programi Prg6_81
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const m=3,n=4,l=2;
  int A[m][n][l]={    {{3,5},{-2,4},{5,2},{-7,-1}},
                      {{-4,9},{6,1},{-2,7},{4,8}},
                      {{7,3},{6,-5},{8,1},{-2,9}}
                 };
  int i,j,k;
  double s;
  s=0;
  for (k=0;k<l;k++)
      for (i=0;i<m;i++)
          for (j=0;j<n;j++)
               if (A[i][j][k]<0)
                   s=s+A[i][j][k]*A[i][j][k];
  cout << "Fusha A"
       << "\n";
  for (k=0;k<l;k++)
  {
      cout << "  k="
           << k
           << "\n";
      for (i=0;i<m;i++)
      {
          for (j=0;j<n;j++)
          {
               cout.width(3);
               cout << A[i][j][k];
          }
          cout << "\n";
      }
      cout << "\n";
  }
  cout << "Shuma s="
       << s
       << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë, fusha e lexuar `A` si dhe shuma e katrorëve të anëtarëve negativë të saj në ekran do të shtypen kështu:

```
Fusha A

k=0
 3 -2  5 -7
-4  6 -2  4
 7  6  8 -2

k=1
 5  4  2 -1
 9  1  7  8
 3 -5  1  9

Shuma s=103
```

Detyra

Për fushën tridimensionale `A(m,n,l)`, të gjendet shuma:

a. e kubeve të anëtarëve të saj;

b. e anëtarëve të saj, të pjesëtuar me shumën e indekseve përkatëse;

c. e anëtarëve të diagonales së saj;

d. e anëtarëve me indeks të parë tek e indeks të tretë çift;

e. e anëtarëve me të tri indekset tek;

f. e anëtarëve pozitivë, por me indeks të tretë tek.

Për gjetjen e anëtarit të caktuar, edhe te fushat tridimensionale mund të veprohet plotësisht njëlloj si edhe te vektorët ose matricat.

Shembull

Gjetja e anëtarit më të madh për nga vlera absolute `e`, në fushën tredimensionale `A(m,n,l)`.

a. Bllok-diagrami

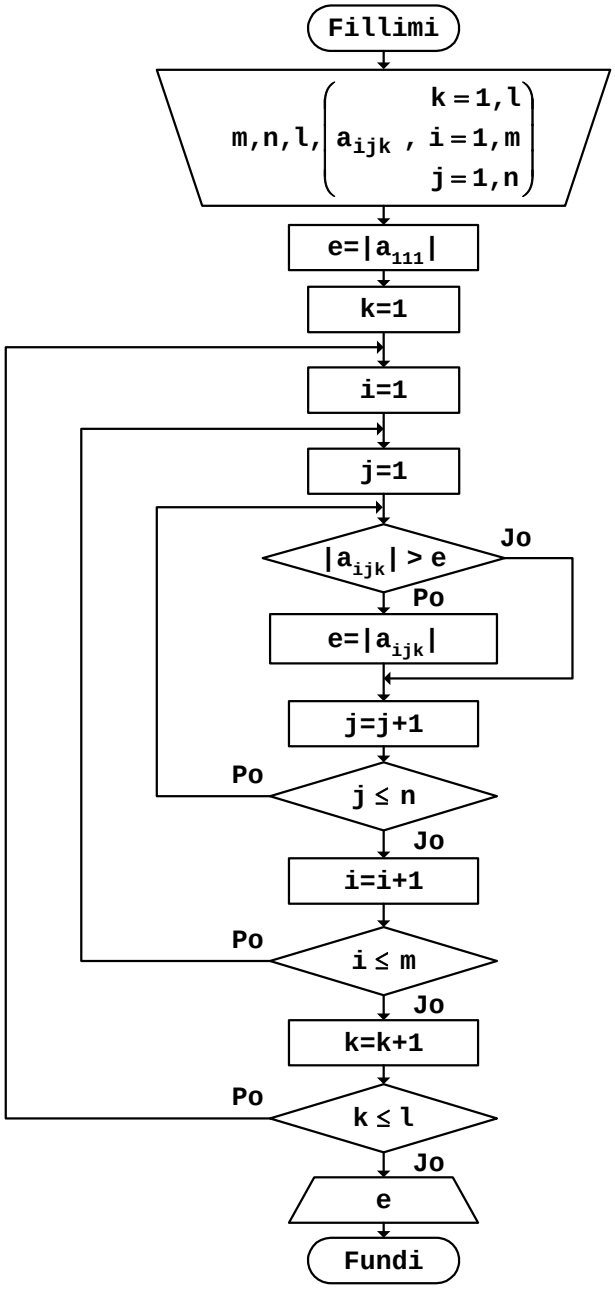


Fig.6.82

b. Programi

```
// Programi Prg6_82
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
  int const m=3,n=4,l=2;
  int A[m][n][l]={   {{3,-9},{-2,3},{4,2},{-5,-1}},
                     {{-4,8},{6,1},{-2,7},{-4,-8}},
                     {{7,3},{6,-5},{8,1},{-2,1}}
                 };
  int i,j,k,e;
  e=abs(A[0][0][0]);
  for (k=0;k<l;k++)
      for (i=0;i<m;i++)
          for (j=0;j<n;j++)
              if (abs(A[i][j][k])>e)
                  e=abs(A[i][j][k]);
  cout << "Fusha A"
       << "\n";
  for (k=0;k<l;k++)
  {
      cout << "  k="
           << k
           << "\n";
      for (i=0;i<m;i++)
      {
          for (j=0;j<n;j++)
          {
              cout.width(3);
              cout << A[i][j][k];
          }
          cout << "\n";
      }
      cout << "\n";
  }
  cout << "Vlera absolute më e madhe e="
       << e
       << "\n";
  return 0;
}
```

Për vlerat numerike të anëtarëve të fushës tridimensionale, që është marr si shembull në program, rezultati që shtypet në ekran është:

```
  Fusha A

  k=0
  3 -2  4 -5
 -4  6 -2 -4
  7  6  8 -2

  k=1
 -9  3  2 -1
  8  1  7 -8
  3 -5  1  1

Vlera absolute më e madhe e=9
```

Edhe te fushat tridimensionale mund të numërohen anëtarët e caktuar, plotësisht njëlloj si edhe te matricat ose vektorët.

Shembull Numri i anëtarëve pozitivë `p` brenda fushës tridimensionale `A(m,n,l)`, por të cilët gjenden mes vlerave `5` dhe `9`, duke i shtypur njëkohësisht anëtarët negativë dhe indekset e tyre.

a. Bllok-diagrami

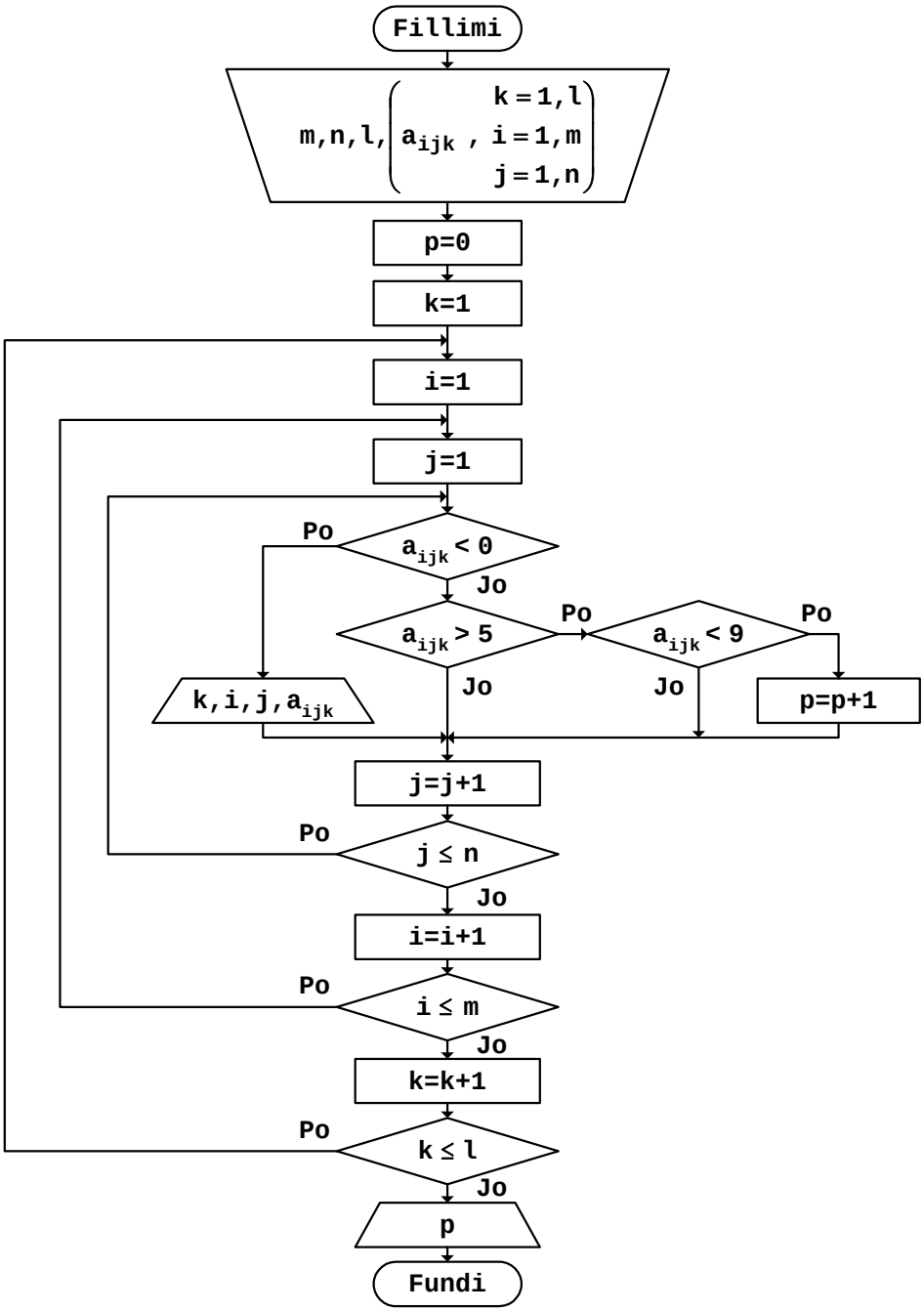

Fillimi
m,n,l, a_ijk , k=1,l ; i=1,m ; j=1,n
p=0
k=1
i=1
j=1
Po
a_ijk < 0
Jo
a_ijk > 5
Po
a_ijk < 9
Po
k,i,j,a_ijk
Jo
Jo
p=p+1
j=j+1
Po
j ≤ n
Jo
i=i+1
Po
i ≤ m
Jo
k=k+1
Po
k ≤ l
Jo
p
Fundi

Fig.6.83

b. Programi

```
// Programi Prg6_83
#include <iostream>
using namespace std;
int main()
{
      int const m=3,n=4,l=2;
      int A[m][n][l]={ {{3,-9},{-2,3},{4,2},{-5,-1}},
                       {{-4,8},{6,1},{-2,7},{-4,-8}},
                       {{7,3},{6,-5},{8,1},{-2,1}}};
      int i,j,k,p;
      p=0;
      cout << "        k       i       j   A[i][j][k]"
           <<"\n";
      for (k=0;k<l;k++)
          for (i=0;i<m;i++)
              for (j=0;j<n;j++)
                  if (A[i][j][k]<0)
                     {
                       cout << "\t"
                            << k
                            << "\t"
                            << i
                            << "\t"
                            << j
                            << "\t"
                            << A[i][j][k]
                            << "\t";
                     }
                  else
                      if (A[i][j][k]>5 && A[i][j][k]<9)
                         p=p+1;
      cout << "Numri i kërkuar p="
           << p
           << endl;
      return 0;
}
```

Rezultati që shtypet në ekran, pas ekzekutimit të programit të dhënë, është:

```
k       i       j   A[i][j][k]
0       0       1       -2
0       0       3       -5
0       1       0       -4
0       1       2       -2
0       1       3       -4
0       2       3       -2
```

```
        1       0       0       -9
        1       0       3       -1
        1       1       3       -8
        1       2       1       -5
Numri i kërkuar p=6
```

ku në pjesën e parë janë shtypur indekset dhe vlerat e anëtarëve negativë të fushës, kurse `p=6` është numri i anëtarëve pozitivë të fushës të cilët gjenden mes vlerave `5` dhe `9`.

Detyra

Në fushën tridimensionale `A(m,n,l)` të gjendet:

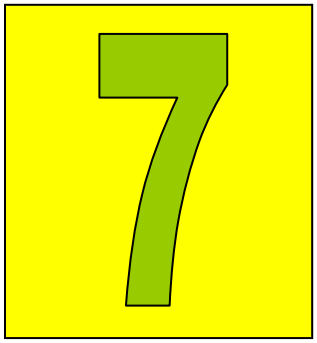
a. numri i anëtarëve pozitivë;

b. numri i anëtarëve më të mëdhenj se vlera `x`, por të ndryshëm nga vlera `y`;

c. shuma e anëtarëve pozitivë;

d. prodhimi i anëtarëve më të vegjël se vlera `x`;

e. anëtari me vlerë minimale, për çdo vlerë të indeksit të tretë.



Vlerat e serive të pafundme

Në praktikë, vlerat numerike të funksioneve të ndryshme llogariten duke shfrytëzuar seri të pafundme. Por, meqë realisht nuk mund të shkohet në pafundësi, llogaritjet ndërpriten duke marrë parasyshë një numër të caktuar anëtarësh të serive, me çka fitohen vlerat e përafërta të funksioneve.

Shembull Llogaritja e vlerës së përafërt `s` të shumës:

$$\sum_{i=1}^{\infty} \frac{1}{i^{2i}}$$

duke marrë parasysh vetëm anëtarët të cilët janë më të mëdhenj se numri i dhënë ε.

a. Bllok-diagrami

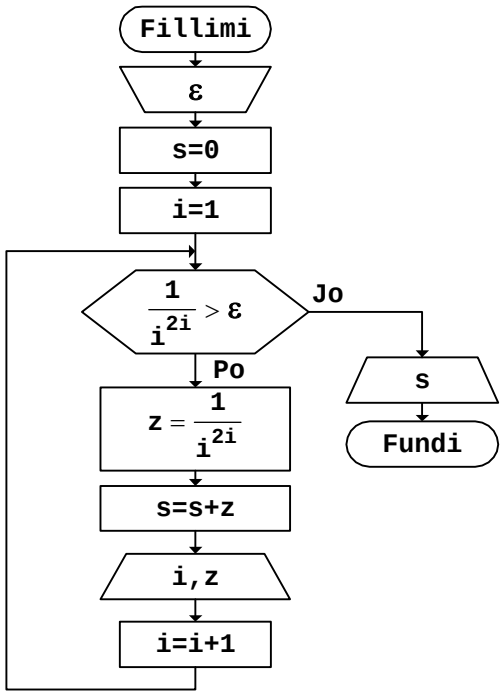


Fig.7.1

Algoritmet e tillë te të cilët paraprakisht nuk dihet se sa herë do të përsëritet një pjesë e caktuar e tyre, quhen edhe *algoritme iterative*.

b. Programi

```
// Programi Prg7_1
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      const double e=0.0000001;
      float i;
      double z,s;
      s=0;
      i=1;
      while ((1/pow(i,2*i)) > e)
      {
            z=1/pow(i,2*i);
            s=s+z;
            cout.precision(0);
            cout.width(5);
            cout <<i;
            cout.precision(7);
            cout.width(12);
            cout.setf(ios::fixed, ios::floatfield);
            cout <<z;
            cout << "\n";
            i=i+1;
      }
      cout << "Shuma s="
           << s
           <<"\n";
      return 0;
}
```

Në program, fuqizimi është llogaritur përmes funksionit `pow`, te i cili, argumenti i parë e paraqet madhësin që fuqizohet, kurse fuqia shënohet si argument i dytë.

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

```
1   1.0000000
2   0.0625000
3   0.0013717
4   0.0000153
5   0.0000001
```

`Shuma s=1.0638871`
ku në kolonën e dytë janë shtypur anëtarët e serisë së pafundme të cilët marrin pjesë në llogaritjen e vlerës së shumës.

Gjatë llogaritjes së vlerave përmes serive të pafundme, llogaritja mund të ndërpritet në një numër të caktuar anëtarësh të serisë, i cili zgjidhet lirisht.

Shembull Llogaritja e vlerës së përafërt `S` të shumës:

$$\sum_{i=1}^{\infty}(-1)^i \frac{x}{3i-2}$$

duke marrë parasysh `k`-anëtarët e parë të serisë.

a. Bllok-diagrami

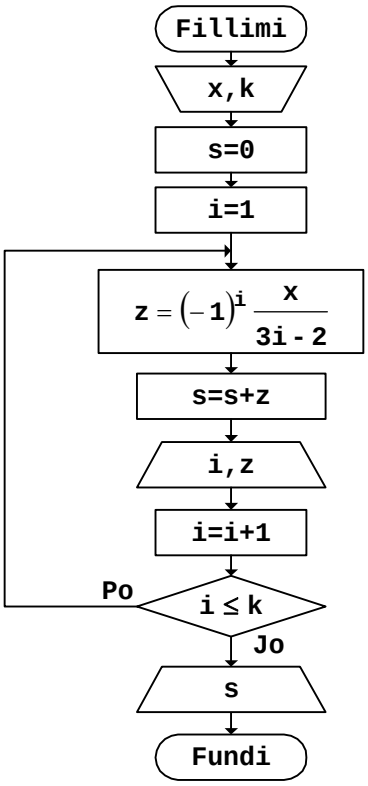


Fig.7.2

b. Programi

```
// Programi Prg7_2
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      int const x=2,k=5;
      int i;
      double z,s;
      s=0;
      for (i=1;i<=k;i++)
      {
            z=pow((-1.),i)*x/(3*i-2);
            s=s+z;
            cout.width(5);
            cout <<i;
            cout.precision(7);
            cout.width(13);
            cout.setf(ios::fixed, ios::floatfield);
            cout <<z;
            cout << "\n";
      }
      cout << "Shuma s="
           << s
           << "\n";
      return 0;
}
```

Në program, për llogaritje të fuqizimit është shfrytëzuar funksioni `pow`, përmes së cilit `(-1)` ngritet në fuqinë `i`. Rezultati që fitohet në ekran, për vlerat `k=5` dhe `x=2`, do të duket kështu:

```
    1   -2.0000000
    2    0.5000000
    3   -0.2857143
    4    0.2000000
    5   -0.1538462
Shuma s=-1.7395604
```

Kushti për ndërprerejen e shtimit të anëtarëve gjatë llogaritjeve përmes serive të pafundme mund të jetë edhe ndonjë vlerë e cila llogaritet nga vlera e anëtarit që shtohet.

Shembull Llogaritja e vlerës së përafërt `s` të shumës:

$$\sum_{i=1}^{\infty}\left(1+\frac{\pi}{5i^4}\right)$$

duke marrë parasysh vetëm anëtarët, logaritmi natyror i të cilëve është më i madh se numri i dhënë ε.

a. Bllok-diagrami

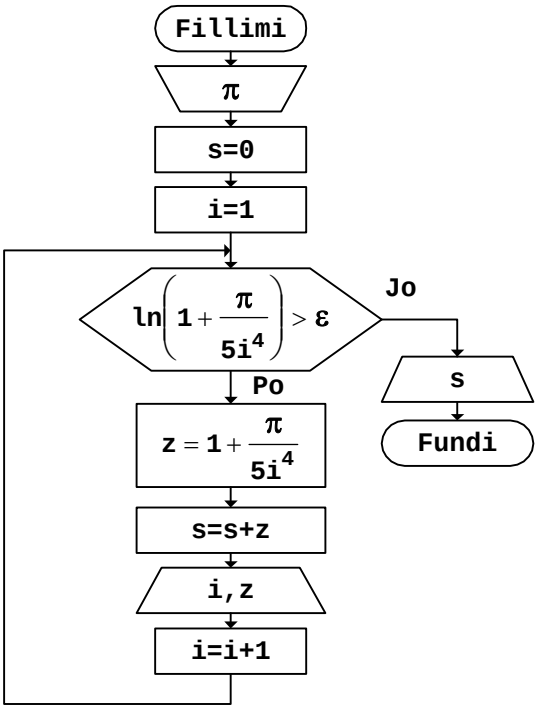


Fig.7.3

b. Programi

```
// Programi Prg7_3
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      const double e=0.001,pi=3.1415926,x=2;
      float i;
      double z,s;
      s=0;
      i=1;
      while (log(1+pi/(5*pow(i,4))) > e)
      {
            z=1+pi/(5*pow(i,4));
            s=s+z;
            cout.precision(0);
            cout.width(5);
            cout <<i;
            cout.precision(5);
            cout.width(10);
            cout.setf(ios::fixed, ios::floatfield);
            cout <<z;
            cout << "\n";
            i=i+1;
      }
      cout << "Shuma s="
           << s
           <<"\n";
      return 0;
}
```

Rezultati që shtypet në ekran, nëse ekzekutohet programi i dhënë, duket kështu:

```
    1   1.62832
    2   1.03927
    3   1.00776
    4   1.00245
    5   1.00101
Shuma s=5.67881
```

ku në kolonën e dytë janë shtypur anëtarët e serisë, të cilët i shtohen vlersës së shumës.

Seritë e pafundme shfrytëzohen gjatë llogaritjes së vlerave të funksioneve të ndryshme trigonometrike.

Shembull

Llogaritja e vlerës së përafërt `s` e funksionit `sin(x)`, përmes serisë së pafundme:

$$\sum_{i=0}^{\infty}(-1)^i \frac{x^{2i+1}}{(2i+1)!}$$

nëse gabimi relativ i lejuar është ε.

Gabimi relativ llogaritet përmes shprehjes:

$$r = \left|\frac{S_v - S_p}{S_v}\right|$$

ku janë:

S_p - shuma parciale paraprake
S_v - shuma parciale vijuese.

a. Bllok-diagrami

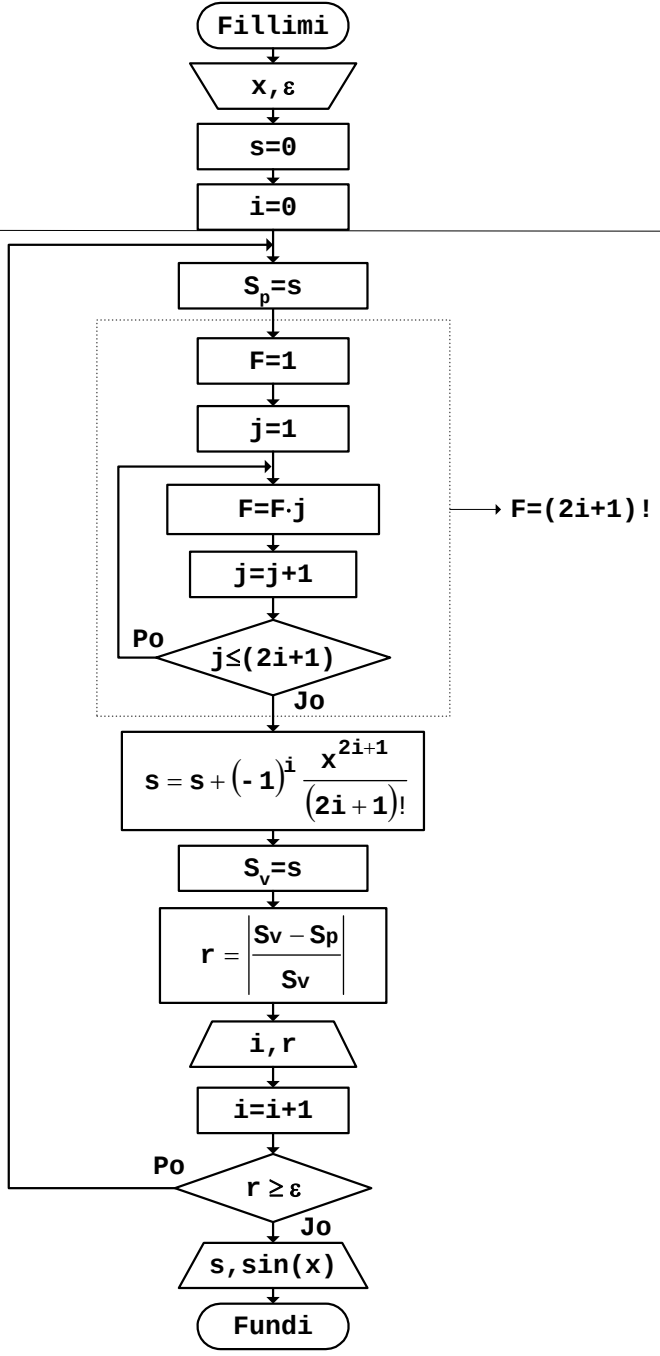


Fig.7.4

b. Programi

```
// Programi
Prg7_4
#include
<iostream>
#include
<cmath>
#include
<iomanip>
using namespace
std;
int main()
{
      const
double
e=0.0001,x=0.75
;
      int i,j;
      double
s,Sp,Sv,r,F;
      s=0;
      i=0;
      do
      {

      Sp=s;
            F=1;

            for
(j=1;j<=2*i+1;j
++)

F=F*j;

            s=s+pow((-1.),i)*pow(x,2*i+1)/F;
            Sv=s;
            r=fabs((Sv-Sp)/Sv);

            cout.width(4);
            cout <<i;
            cout.precision(6);
            cout.width(10);
            cout.setf(ios::fixed, ios::floatfield);
```

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

            i=i+1;
      }
      while (r>=e);

      cout << "Vlera e përafërt s="
           << s
           <<"\n"
           << "Vlera e saktë  sin="
           << sin(x)
           <<"\n";
      return 0;
}
```

Nëse ekzekutohet programi, rezultati që shtypet në ekran është:

```
   0  1.000000
   1  0.103448
   2  0.002901
   3  0.000039
Vlera e përafërt s=0.681639
Vlera e saktë  sin=0.681639
```

Në pjesën e parë të rezultatit janë shtypur vlerat e variablës `i` dhe gabimi absolut `r` për çdo `i`, i cili, siç shihet, shkon gjithnjë duke u zvogëluar. Kurse në dy rreshtat e fundit, së pari është shtypur shuma `s` e vlerës së përafërt të sinusit, dhe pastaj edhe vlera e funksionit `sin(x)`, e llogaritur përmes funksionit `sin`, i cili gjendet në kuadër të bibliotekës së funksioneve të gjuhës `C++`. Siç shihet, vlera e llogaritur përmes serisë së pafundme dhe ajo e llogaritur duke shfrytëzuar funksionin `sin`, për saktësinë e kërkuar, janë të barabarta.

Vlerat e serive të pafundme mund të llogariten edhe duke e pasur parasysh gabimin e lejuar relativ.

Shembull Llogaritja e vlerës së përafërt `s` e serisë së pafundme:

$$\sum_{i=1}^{\infty} \frac{x}{i^3}$$

nëse gabimi absolut i lejuar është ε.

Gabimi absolut llogaritet përmes shprehjes:

$$a = |S_v - S_p|$$

ku janë:

S_p - shuma parciale paraprake
S_v - shuma parciale vijuese.

a. Bllok-diagrami

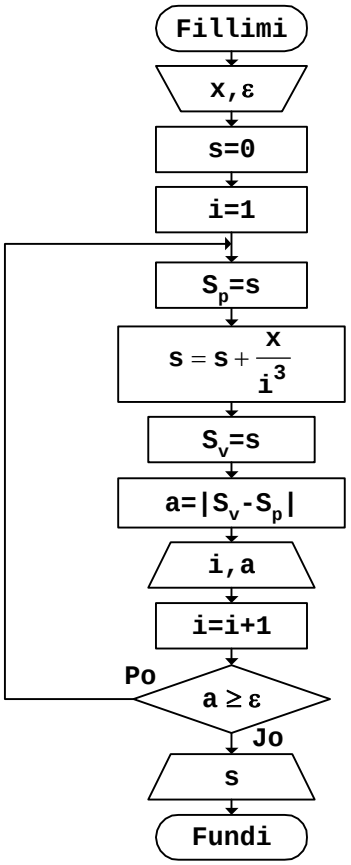


Fig.7.5

b. Programi

```
// Programi Prg7_5
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      const double x=2,e=0.01;
      float i;
      double s,Sp,Sv,a;
      s=0;
      i=1;
      do
      {
```

```
            Sp=s;
            s=s+x/pow(i,3);
            Sv=s;
            a=fabs(Sv-Sp);
            cout.precision(0);
            cout.width(4);
            cout <<i;
            cout.precision(6);
            cout.width(10);
            cout.setf(ios::fixed, ios::floatfield);
            cout <<a;
            cout << "\n";
            i=i+1;
      }
      while (a>=e);
      cout << "Vlera e llogaritur s="
           << s
           <<"\n";
      return 0;
}
```

Rezultati që fitohet në ekran pas ekzekutimit të programit të dhënë është:

```
   1  2.000000
   2  0.250000
   3  0.074074
   4  0.031250
   5  0.016000
   6  0.009259
Vlera e llogaritur s=2.380583
```

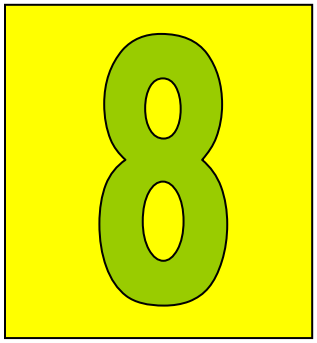
Në kolonën e dytë të rezultateve të dhëna më sipër janë shtypur vlerat e gabimit absolut `a` në iteracionet e veçanta, përkatësisht për çdo vlerë të variablës `i`.

Detyrë

Të llogaritet vlera e përafërt `z` e funksionit e^x, përmes serisë së pafundme:

$$\sum_{i=1}^{\infty} \frac{x}{i^3}$$

nëse gabimi relativ i lejuar është ε.



Numrat kompleksë

Gjatë llogaritjeve të ndryshme, kur kemi të bëjmë me vlera komplekse, operohet veç me pjesën reale dhe veç me atë imagjinare.

Shembull Llogaritja e shumës së anëtarëve kompleksë të vektorit `Z(n)`, nëse anëtarët e pjesës reale dhe të pjesës imagjinare janë dhënë përmes vektorëve `X(n)` dhe `Y(n)`.

$$
\begin{aligned}
s &= \sum_{i=1}^{n} z_i \\
&= \sum_{i=1}^{n} (x_i + j y_i) \\
&= \sum_{i=1}^{n} x_i + j \sum_{i=1}^{n} y_i \quad = a + j\,b
\end{aligned}
$$

a. Bllok-diagrami

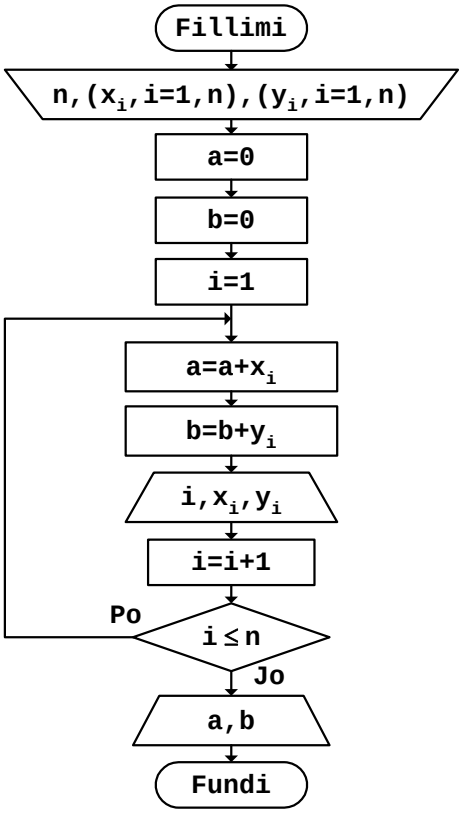


Fig.8.1

b. Programi

```
// Programi Prg8_1
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
      int const n=5;
      int a,b,i;
      int X[n]={3,7,4,2,-8};
      int Y[n]={6,-9,5,8,1};
      a=0;
      b=0;
      cout << "   i   X   Y"
```

```
            << "\n";
      cout << "-------------"
            << "\n";
      for (i=0;i<n;i++)
      {
          a=a+X[i];
          b=b+Y[i];
          cout << setw(4) << i
               << setw(4) << setiosflags(ios::right)<<X[i]
               << setw(4) << setiosflags(ios::right)<<Y[i]
               << "\n";
      }
      cout << "Shuma reale .... a="
            << a
            << "\n"
            << "Shuma imagjinare b="
            << b
            << "\n";
      return 0;
}
```

Për vlerat e vektorit kompleks `Z(n)`, përkatësisht komponentes reale `X(n)` dhe komponentes imagjinare `Y(n)`, të cilat janë marrë si shembull, pas ekzekutimit të programit në ekran do të shtypen këto rezultate:

```
   i   X   Y
-------------
   1   3   6
   2   7  -9
   3   4   5
   4   2   8
   5  -8   1
Shuma reale .... a=8
Shuma imagjinare b=11
```

Për llogaritjen e prodhimit të anëtarëve të një vektori kompleks duhet shfrytëzuar formën eksponenciale të shprehjes së anëtarëve përkatës të vektorit.

Shembull Llogaritja e prodhimit të anëtarëve kompleksë të vektorit `Z(n)`, nëse anëtarët e pjesës reale dhe të pjesës imagjinare janë dhënë përmes vektorëve `X(n)` dhe `Y(n)`.

$$
\begin{aligned}
P &= \prod_{i=1}^{n}(x_i + jy_i) \\
&= \prod_{i=1}^{n}\left(\rho_i \cdot e^{j\varphi_i}\right) \\
&= \left(\rho_1 \cdot e^{j\varphi_1}\right)\cdot\left(\rho_2 \cdot e^{j\varphi_2}\right)\cdot\ldots\cdot\left(\rho_n \cdot e^{j\varphi_n}\right) \\
&= \prod_{i=1}^{n}\rho i \cdot e^{j\sum_{i=1}^{n}\varphi_i} = \alpha \cdot e^{j\beta} \\
&= \alpha \cdot \cos\beta + j\alpha \cdot \sin\beta \\
&= a + jb
\end{aligned}
$$

ku janë:

$$\alpha = \prod_{i=1}^{n}\rho_i$$

$$\beta = \sum_{i=1}^{n}\varphi_i$$

$$a = \alpha \cdot \cos\beta\,,$$

$$b = \alpha \cdot \sin\beta$$

$$\rho_i = \sqrt{x_i^2 + y_i^2}$$

$$\varphi_i = \operatorname{arctg}\frac{y_i}{x_i}$$

a. Bllok-diagrami

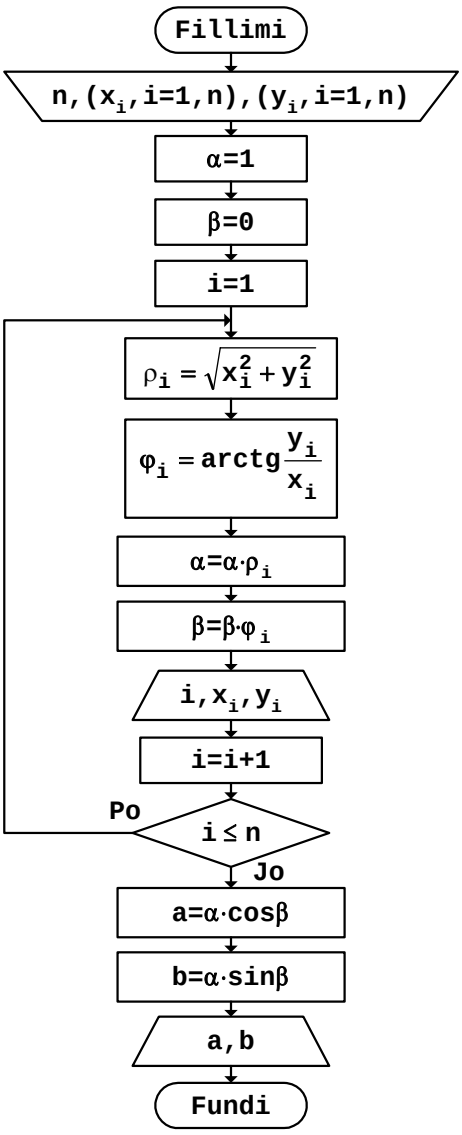


Fig.8.2

b. Programi

```
// Programi Prg8_2
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      int const n=5;
      int i;
      double X[n]={3,7,4,2,-8};
      double Y[n]={6,-9,5,8,1};
      double a,b,Alfa,Beta,Ro[n],Fi[n];
      Alfa=1;
      Beta=0;
      cout << "   i   X   Y"
           << "\n";
      cout << "-------------"
           << "\n";
      for (i=0;i<n;i++)
      {
         Ro[i]=sqrt(pow(X[i],2)+pow(Y[i],2));
         Fi[i]=atan(Y[i]/X[i]);
         Alfa=Alfa*Ro[i];
         Beta=Beta+Fi[i];
         cout << setw(4) << i
              << setw(4) << setiosflags(ios::right)<< X[i]
              << setw(4) << setiosflags(ios::right)<< Y[i]
              << "\n";
      }
      a=Alfa*cos(Beta);
      b=Alfa*sin(Beta);
      cout << "Pjesa reale .... a="
           << a
           << "\n"
           << "Pjesa imagjinare b="
           << b
           << "\n";
      return 0;
}
```

Nëse ekzekutohet programi i dhënë, rezultati që shtypet në ekran do të duket si në vijim.

```
   i    X    Y
-------------
   0    3    6
   1    7   -9
   2    4    5
   3    2    8
   4   -8    1
Pjesa reale .... a=-21570
Pjesa imagjinare b=24390
```

Për llogaritjen e shumës ose të prodhimit të funksioneve trigonometrike, ose edhe të funksioneve të tjera, duhet gjetur rrugë që shprehjet të zbërthehen në pjesën reale dhe në pjesën imagjinare të tyre.

Shembull Llogaritja e shumës:

$$s = \sum_{i=1}^{n} \sin z_i$$

ku z_i janë anëtarët kompleksë të vektorit `Z(n)`, kurse anëtarët e pjesës reale dhe të pjesës imagjinare të tij janë dhënë përmes vektorëve `X(n)` dhe `Y(n)`.

$$\begin{aligned} s &= \sum_{i=1}^{n} \sin z_i \\ &= \sum_{i=1}^{n} \sin(x_i + jy_i) \\ &= \sum_{i=1}^{n} \frac{e^{j(x_i+jy_i)} - e^{-j(x_i+jy_i)}}{2j} \\ &= \sum_{i=1}^{n} \frac{e^{y_i} + e^{-y_i}}{2} \sin x_i + j \sum_{i=1}^{n} \frac{e^{y_i} - e^{-y_i}}{2} \cos x_i \\ &= a + jb \end{aligned}$$

ku janë:

$$a = \sum_{i=1}^{n} \frac{e^{y_i} + e^{-y_i}}{2} \sin x_i$$

$$b = \sum_{i=1}^{n} \frac{e^{y_i} - e^{-y_i}}{2} \cos x_i$$

a. Bllok-diagrami

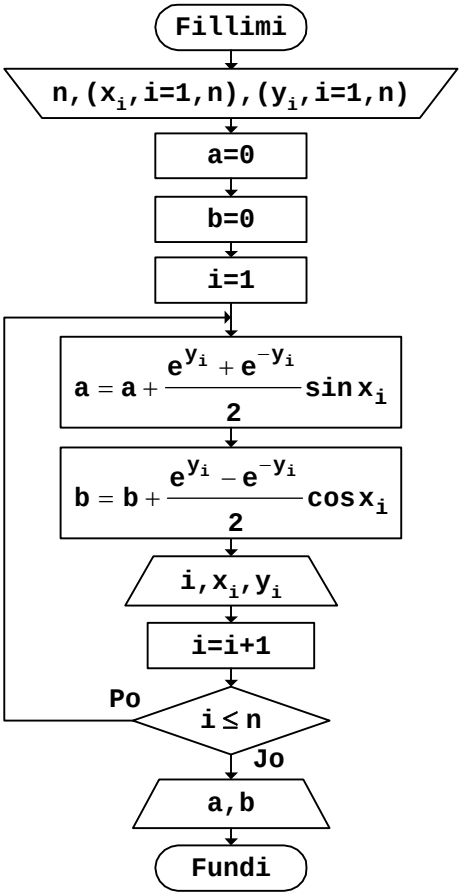


Fig.8.3

b. Programi

```
// Programi Prg8_3
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      int const n=5;
      int i;
      double X[n]={3,7,4,2,-8};
      double Y[n]={6,-9,5,8,1};
```

```
      double a=0,b=0;
      cout << "   i   X   Y"
           << "\n";
      cout << "-------------"
           << "\n";
      for (i=0;i<n;i++)
      {
         a=a+(exp(Y[i])+exp(-Y[i]))/2*sin(X[i]);
         b=b+(exp(Y[i])-exp(-Y[i]))/2*cos(X[i]);
         cout << setw(4) << i
              << setw(4) << setiosflags(ios::right)<<X[i]
              << setw(4) << setiosflags(ios::right)<<Y[i]
              << "\n";
      }
      cout << "Pjesa reale .... a="
           << a
           << "\n"
           << "Pjesa imagjinare b="
           << b
           << "\n";
      return 0;
}
```

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

```
   i   X   Y
-------------
   1   3   6
   2   7  -9
   3   4   5
   4   2   8
   5  -8   1
Pjesa reale .... a=3987.87
Pjesa imagjinare b=-3923.09
```

Detyra

Nëse anëtarët e vektorit `Z(n)` janë numra kompleksë dhe pjesët reale dhe imagjinare të tyre jepen përmes vektorëve `X(n)` dhe `Y(n)`, të llogariten vlerat:

$$p = \prod_{i=1}^{n} \cos z_i$$

$$s = \sum_{i=1}^{n} \ln z_i$$

9

Tabelimi i funksioneve

Me tabelimin e vlerave numerike të funksioneve nënkuptohet llogaritja e vlerave të tyre, për vlera të ndryshme të variablave që paraqiten brenda shprehjeve përkatëse.

Shembull

Llogaritja e vlerave të funksionit:

$$y = \begin{cases} e^{2x+1} & \text{për } x \le 3 \\ \sum_{i=1}^{n} \left(\frac{x}{2} + 2i \right)^2 & \text{për } x > 3 \end{cases}$$

për vlera të ndryshme të variablës x mes a dhe b, duke ndryshuar atë me hapin h.

a. Bllok-diagrami

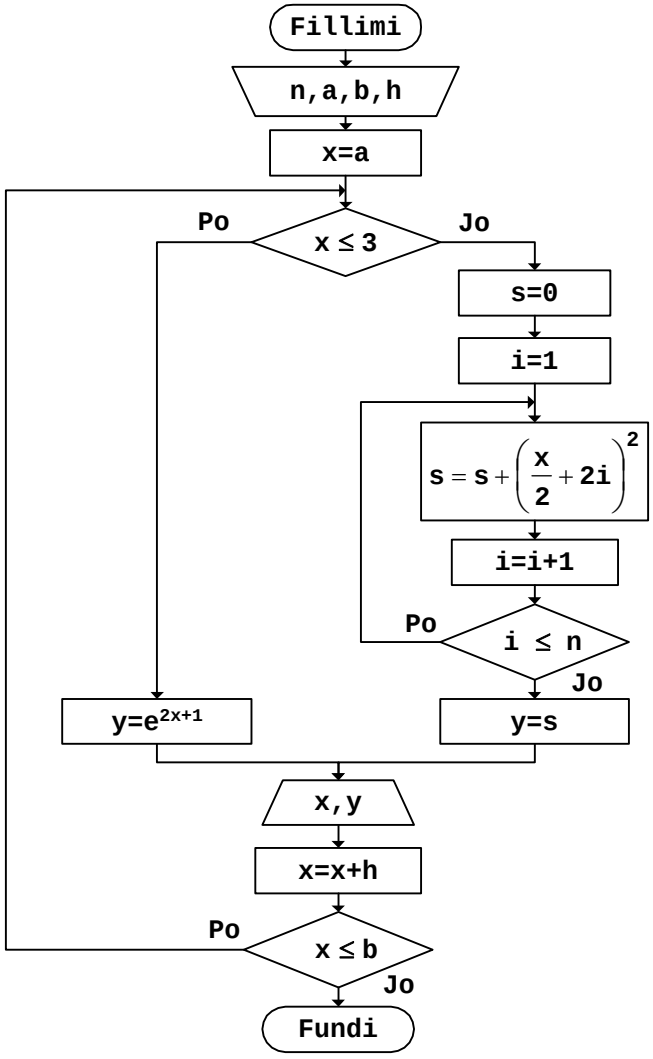


Fig.9.1

b. Programi

```
// Programi Prg9_1
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      const double a=2,b=5,h=0.5;
      int n=3,i;
      double x,y,s;
      cout << "Tabela e vlerave"
           << "\n";
      cout << "   x          y"
           << "\n";
      x=a;
      do
      {
            if (x<=3)
               y=exp(2*x+1);
            else
            {
               s=0;
               for (i=1;i<=n;i++)
                   s=s+pow(x/2+2*i,2);
               y=s;
            }
            cout.setf(ios::fixed);
            cout.precision(2);
            cout.width(6);
            cout << x;

            cout.precision(4);
            cout.width(12);
            cout << y
                 << "\n";
            x=x+h;
      }
      while (x<=b);
  return 0;
}
```

Rezultati që shtypet në ekran do të duket:

```
Tabela e vlerave
   x       y
  2.00  148.4132
  2.50  403.4288
```

```
3.00 1096.6332
3.50  107.1875
4.00  116.0000
4.50  125.1875
5.00  134.7500
```

Tabela mund t'i përmbajë vlerat numerike të disa funksionve njëkohësisht.

Shembull Llogaritja e vlerave numerike të funksioneve:

$$y = 3x^2 + 2x - 1$$

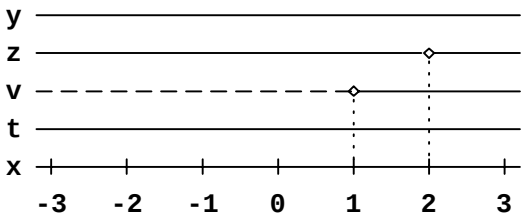
$$z = \frac{1}{2 - x}$$

$$v = 3\ln(x - 1)$$

$$t = e^x + 2\sin x$$

për vlera të ndryshme të variablës x mes a dhe b, duke ndryshuar vlerën e saj me hapin h.

Meqë për të gjitha vlerat e variablës x mund të llogariten vetëm funksionet y dhe t, shprehjet e tyre janë vendosur menjëherë në fillim të bllok-diagramit. Për funksionet e tjera, para se të llogariten vlerat e tyre, duhet kontrolluar vlerën e variablës x, sepse funksioni v nuk mund të llogaritet, nëse $x \le 1$, kurse vlera e funksionit z është e padefinuar për $x=2$, gjë që shihet edhe në paraqitjen vijuese:



a. Bllok-diagrami

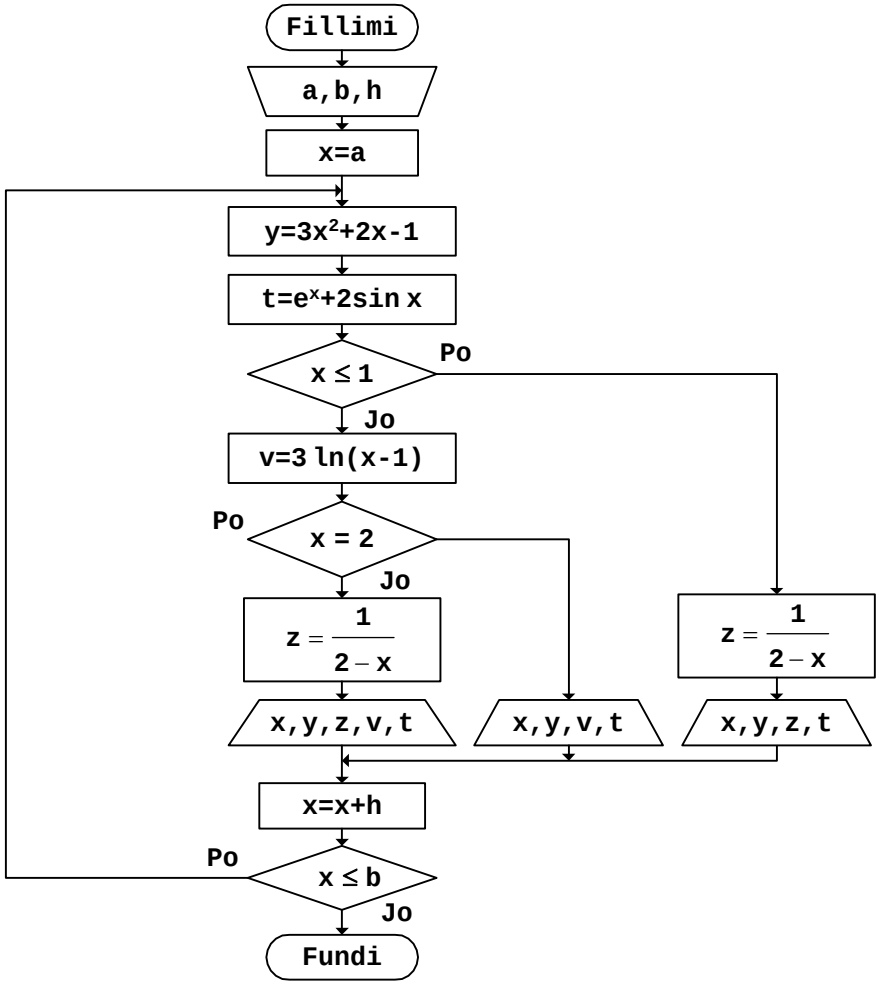


Fig.9.2

b. Programi

```
// Programi Prg9_1
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main()
{
      const double a=-3,b=3,h=0.5;
      double x,y,z,v,t;
      cout << "  x       y     z      v       t"
           << "\n"
           << "---------------------------------"
           << "\n";
      x=a;
      cout.setf(ios::fixed);
      cout.precision(2);
      do
      {
            y=3*pow(x,2)+2*x-1;
            t=exp(x)+2*sin(x);
            if (x<=1)
            {
               z=1/(2-x);
               cout << setw(5)
                    << x
                    << setw(8)
                    << y
                    << setw(6)
                    << z
                    << "  Mungon"
                    << setw(7)
                    << t
                    << "\n";
            }
            else
            {
               v=3*log(x-1);
               if (x==2)
               {
                  cout << setw(5)
                       << x
                       << setw(8)
                       << y
                       << "  ----"
                       << setw(8)
                       << v
                       << setw(7)
```

```
                        << t
                        << "\n";
                }
                else
                {
                   z=1/(2-x);
                   cout << setw(5)
                        << x
                        << setw(8)
                        << y
                        << setw(6)
                        << z
                        << setw(8)
                        << v
                        << setw(7)
                        << t
                        << "\n";
                }
             }
             x=x+0.5;
      }
      while (x<=b);
  return 0;
}
```

Tabela e vlerave numerike, e cila fitohet në ekran pas ekzekutimit të programit të dhënë, do të duket kështu:

```
  x        y      z       v        t
-----------------------------------
-3.00   20.00  0.20  Mungon  -0.23
-2.50   12.75  0.22  Mungon  -1.11
-2.00    7.00  0.25  Mungon  -1.68
-1.50    2.75  0.29  Mungon  -1.77
-1.00    0.00  0.33  Mungon  -1.32
-0.50   -1.25  0.40  Mungon  -0.35
 0.00   -1.00  0.50  Mungon   1.00
 0.50    0.75  0.67  Mungon   2.61
 1.00    4.00  1.00  Mungon   4.40
 1.50    8.75  2.00   -2.08   6.48
 2.00   15.00  ----    0.00   9.21
 2.50   22.75 -2.00    1.22  13.38
 3.00   32.00 -1.00    2.08  20.37
```

10

Mesataret dhe devijimet

Në praktikë, përveç *mesatares aritmetikore*, përdoret edhe *mesatarja gjeometrike*, si dhe *mesatarja harmonike*.

Shembull Llogaritja e mesatares aritmetikore, mesatares gjeometrike dhe e mesatares harmonike, për anëtarët e vektorit `X(n)`.

Për llogaritjen e mesatareve në fjalë, përdoren shprehjet:

- *Mesatarja aritmetikore*

$$a = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- *Mesatarja gjeometrike*

$$g = \sqrt[n]{\prod_{i=1}^{n} x_i}$$

- *Mesatarja harmonike*

$$h = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

a. Bllok-diagrami

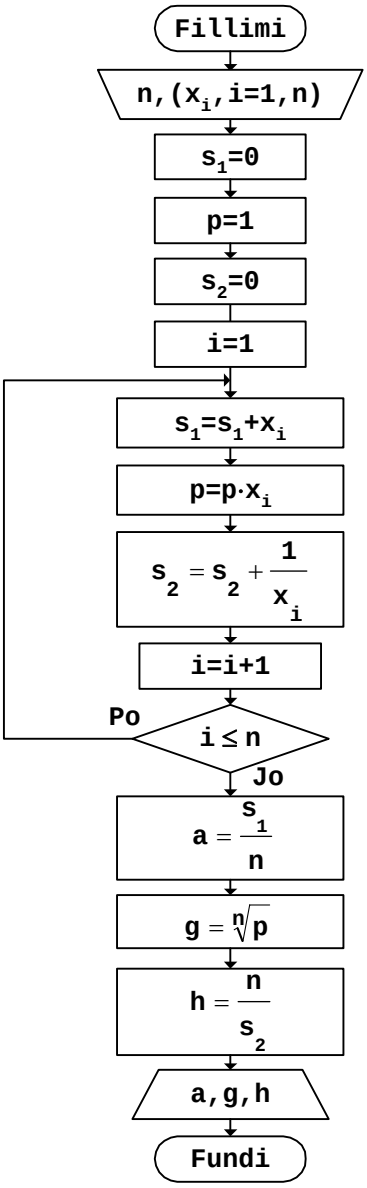


Fig.10.1

b. Programi

```
// Programi Prg10_1
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      const int n=5;
      int X[n]={7,4,2,8,5},i;
      double s1=0,p=1,s2=0,a,g,h;
      for (i=0;i<n;i++)
      {
            s1=s1+X[i];
            p=p*X[i];
            s2=s2+1./X[i];
      }
      a=s1/n;
      cout << "Mesatarja aritmetikore a="
           << a;
      g=pow(p,1./n);
      cout << "\nMesatarja gjeometrike  g="
           << g;
      h=n/s2;
      cout << "\nMesatarja harmonike    h="
           << h
           << "\n";
  return 0;
}
```

Për vlerat e anëtarëve të vektorit që është marra si shembull në program, pas ekzekutimit të programit, si rezultat në ekran do të shtypet:

```
Mesatarja aritmetikore a=5.2
Mesatarja gjeometrike  g=4.67789
Mesatarja harmonike    h=4.10557
```

Duke shfrytëzuar vlerat e llogaritura të mesatareve, përmes shprehjeve vijuese mund të gjenden edhe devijimet standarde përkatëse:

- *Devijimi aritmetikor*

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - a)^2}$$

- *Devijimi gjeometrik*

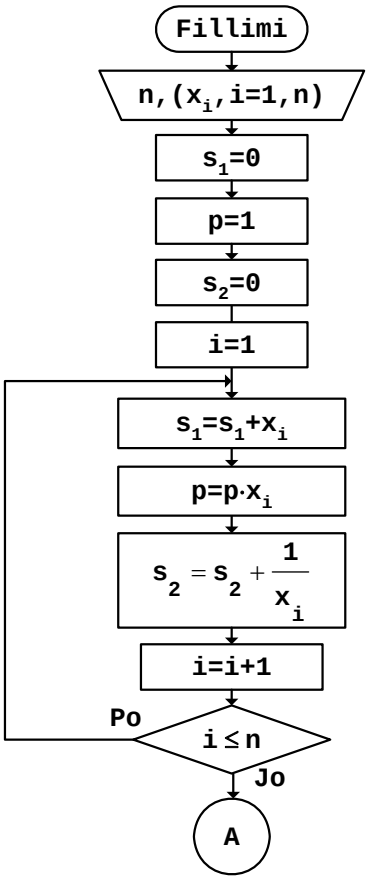
$$\rho = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - g)^2}$$

- *Devijmi absolut*

$$\delta = \frac{1}{n}\sum_{i=1}^{n}|x_i - a|$$

Shembull Llogaritja e devijimit standard aritmetikor, gjeometrik dhe harmonik, për anëtarët e vektorit `X(n)`.

a. Bllok-diagrami



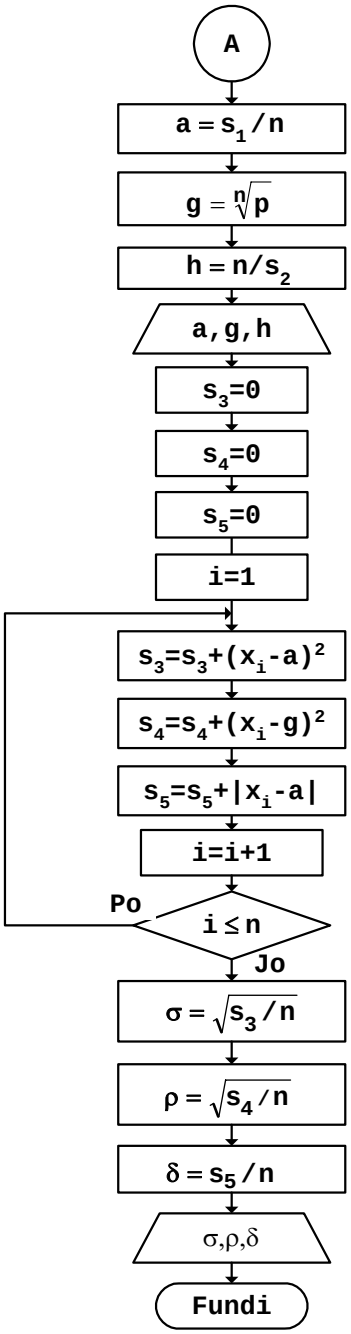

A
a = s1 / n
g = n√p
h = n/s2
a,g,h
s3=0
s4=0
s5=0
i=1
s3=s3+(xi-a)2
s4=s4+(xi-g)2
s5=s5+|xi-a|
i=i+1
Po
i ≤ n
Jo
σ = √s3 / n
ρ = √s4 / n
δ = s5 / n
σ,ρ,δ
Fundi

Fig.10.2

b. Programi

```
// Programi Prg10_2
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
      const int n=5;
      int X[n]={7,4,2,8,5},i;
      double s1=0,p=1,s2=0,a,g,h;
      double s3=0,s4=0,s5=0,Sigma,Ro,Delta;
      for (i=0;i<n;i++)
      {
            s1=s1+X[i];
            p=p*X[i];
            s2=s2+1./X[i];
      }
      a=s1/n;
      cout << "Mesatarja aritmetikore a="
           << a;
      g=pow(p,1./n);
      cout << "\nMesatarja gjeometrike  g="
           << g;
      h=n/s2;
      cout << "\nMesatarja harmonike    h="
           << h;
      for (i=0;i<n;i++)
      {
            s3=s3+pow(X[i]-a,2);
            s4=s4+pow(X[i]-g,2);
            s5=s5+fabs(X[i]-a);
      }
      Sigma=sqrt(s3/n);
      cout << "\nDevijimi aritmetikor Sigma="
           << Sigma;
      Ro=sqrt(s4/n);
      cout << "\nDevijimi gjeometrik     Ro="
           << Ro;
      Delta=s5/n;
      cout << "\nDevijimi absolut     Delta="
           << Delta
           << "\n";
  return 0;
}
```

Pas ekzekutimit të programit të dhënë, rezultatet në ekran do të shtypen kështu:

```
Mesatarja aritmetikore a=5.2
Mesatarja gjeometrike  g=4.67789
Mesatarja harmonike    h=4.10557
Devijimi aritmetikor Sigma=2.13542
Devijimi gjeometrik     Ro=2.19832
Devijimi absolut     Delta=1.84
```

11

Integrimi numerik

Për llogaritjen e vlerës së integralit të caktuar, përdoret *Metoda e Trapezit*, ose *Metoda e Simpsonit*.

Integralet e njëfishta

Vlera e integralit të njëfishtë:

$$\int_a^b f(x)dx$$

sipas *Metodës së Trapezit* llogaritet me shprehjen:

$$T = h \cdot \left[\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(a + i \cdot h) \right]$$

ku

$$h = \frac{b - a}{n}$$

është hapi i integrimit, nëse zona e integrimit ndahet në n-pjesë.

Shembull

Llogaritja e vlerës së integralit:

$$\int_1^4 (x^2 + 1)dx$$

me metodën numerike të *Trapezit*.

a. Bllok-diagrami

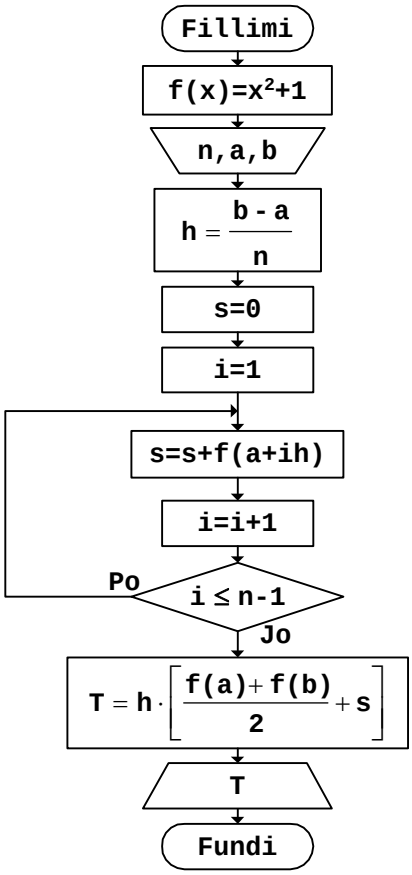


Fig.11.1

b. Programi

```
// Programi Prg11_1
#include <iostream>
using namespace std;
double f(double x);
int main()
{
      const int n=10;
      const double a=1,b=4;
      int i;
      double T,s,h;
      h=(b-a)/n;
      s=0;
      for (i=1;i<=n-1;i++)
```

```
          s=s+f(a+i*h);
      T=h*((f(a)+f(b))/2+s);
      cout << "Vlera e integralit T="
           << T
           << "\n";
  return 0;
}

// Nënprogrami
double f(double x)
{
  return x*x+1;
}
```

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

```
Vlera e integralit T=24.045
```

Vlera e integralit llogaritet edhe përmes *Metodës së Simpsonit*, duke shfrytëzuar shprehjen:

$$S = \frac{h}{3} \cdot \left\{ f(a) + f(b) + 4\sum_{i=1}^{n} f[a + (2i - 1)h] + 2\sum_{i=1}^{n-1} f[a + 2ih] \right\}$$

ku hapi i integrimit llogaritet kështu:

$$h = \frac{b - a}{2n}$$

nëse zona e integrimit ndahet në n-pjesë.

Shembull Llogaritja e vlerës së integralit:

$$\int_{1}^{4} (x^2 + 1)dx$$

me metodën numerike të *Simpsonit*.

a. Bllok-diagrami

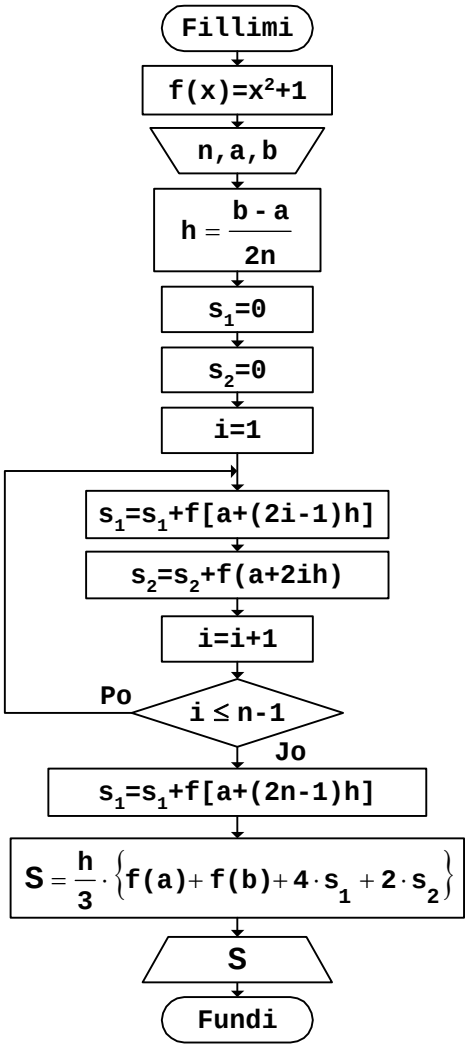


Fig.11.2

b. Programi

```
// Programi Prg11_2
#include <iostream>
using namespace std;
double f(double x);
int main()
{
      const int n=10;
      const double a=1,b=4;
      int i;
      double S,s1,s2,h;
      h=(b-a)/(2*n);
      s1=0;
      s2=0;
      for (i=1;i<=n-1;i++)
      {
          s1=s1+f(a+(2*i-1)*h);
          s2=s2+f(a+2*i*h);
      }
      s1=s1+f(a+(2*n-1)*h);
      S=h/3*(f(a)+f(b)+4*s1+2*s2);
      cout << "Vlera e integralit S="
           << S
           << "\n";
  return 0;
}

// Nënprogrami
double f(double x)
{
  return x*x+1;
}
```

Pas ekzekutimit të programit të dhënë, rezulati në ekran do të shtypet kështu:

```
Vlera e integralit S=24
```

Integralet e dyfishta

Për llogaritjen e vlerës së integralit të dyfishtë:

$$\int_a^b \int_c^d f(x, y)\,dx\,dy$$

sipas *Metodës së Trapezit*, shfrytëzohet shprehja:

$$T = h \cdot g \cdot \left\{ \frac{1}{4}[f(a,c) + f(a,d) + f(b,c) + f(b,d)] + \frac{1}{2}\left[\sum_{i=1}^{n-1} f(x_i,c) + f(x_i,d) + \sum_{j=1}^{m-1} f(a,y_j) + f(b,y_j) \right] + \sum_{j=1}^{m-1} \sum_{i=1}^{n-1} f(x_i,y_j) \right\}$$

ku janë:

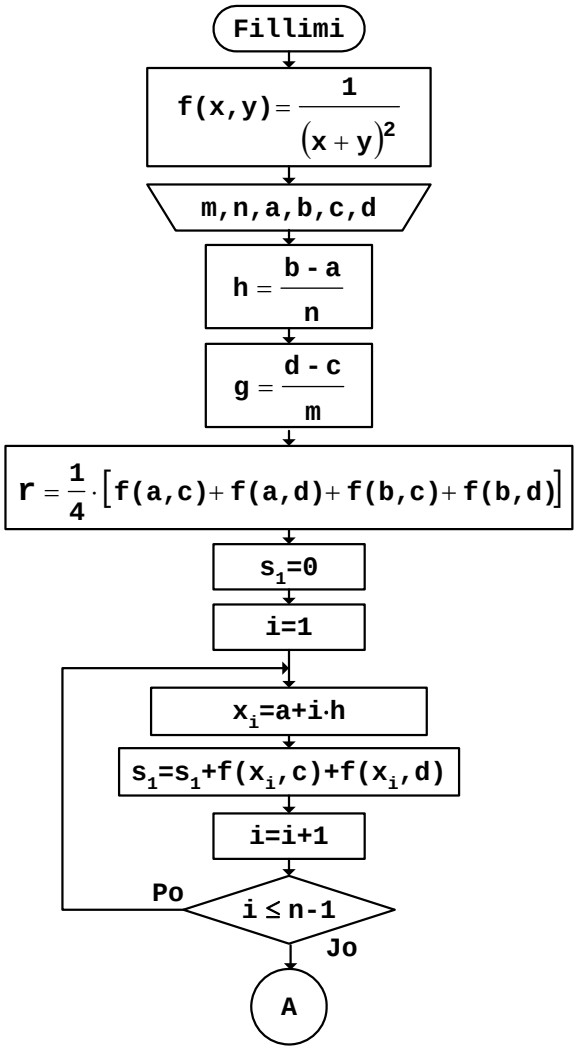
$$x_i = a + ih$$
$$y_j = c + jd$$
$$h = \frac{b - a}{n}$$
$$g = \frac{d - c}{m}$$

Shembull Llogaritja e vlerës së integralit të dyfishtë:

$$\int_1^2 \int_3^4 \frac{1}{(x + y)^2}\,dx\,dy$$

me metodën numerike të *Trapezit*.

a. Bllok-diagrami



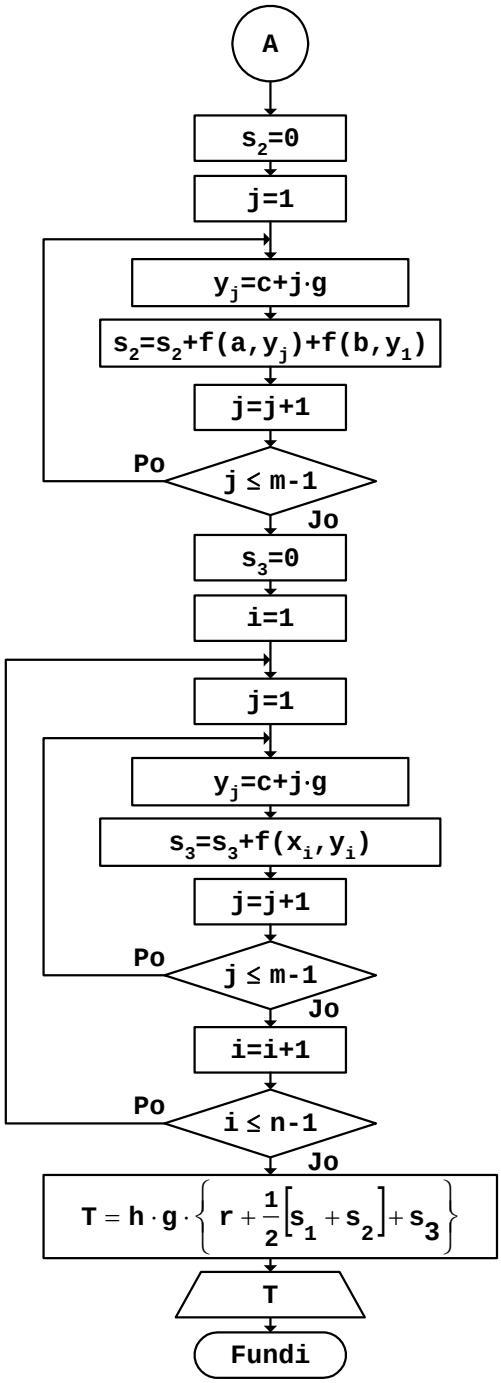

A
$s_2=0$
j=1
$y_j=c+j\cdot g$
$s_2=s_2+f(a,y_j)+f(b,y_1)$
j=j+1
Po
$j \le m-1$
Jo
$s_3=0$
i=1
j=1
$y_j=c+j\cdot g$
$s_3=s_3+f(x_i,y_i)$
j=j+1
Po
$j \le m-1$
Jo
i=i+1
Po
$i \le n-1$
Jo
$T = h \cdot g \cdot \left\{ r + \frac{1}{2}\left[s_1 + s_2\right] + s_3 \right\}$
T
Fundi

Fig.11.3

b. Programi

```
// Programi Prg11_3
#include <iostream>
#include <cmath>
using namespace std;
double f(double x,double y);
int main()
{
      const int m=6,n=4;
      const double a=1,b=2,c=3,d=4;
      int i,j;
      double T,s1,s2,s3,r,h,g,xi,yj;
      h=(b-a)/n;
      g=(d-c)/m;
      r=(f(a,c)+f(a,d)+f(b,c)+f(b,d))/4;

            s1=0;
            for (i=1;i<=n-1;i++)
            {
                  xi=a+i*h;
                  s1=s1+f(xi,c)+f(xi,d);
            }

      s2=0;
      for (j=1;j<=m-1;j++)
      {
            yj=c+j*g;
            s2=s2+f(a,yj)+f(b,yj);
      }

            s3=0;
            for (i=1;i<=n-1;i++)
            {
                  xi=a+i*h;
                  for (j=1;j<=m-1;j++)
                  {
                        yj=c+j*g;
                        s3=s3+f(xi,yj);
                  }
            }

      T=h*g*(r+(s1+s2)/2+s3);

      cout << "Vlera e integralit T="
             << T
             << "\n";
  return 0;
}
```

```
// Nënprogrami
double f(double x,double y)
{
  return 1/pow(x+y,2);
}
```

Rezultati që do të shtypet në ekran, nëse ekzekutohet programi i dhënë, është:

```
Vlera e integralit T=0.0408994
```

12

Ekuacionet diferenciale

Për zgjidhjen e ekuacioneve diferenciale lineare përdoren metoda të ndryshme numerike. Këtu do të jepen metodat *Runge-Kutta* dhe *Kutta-Merson*.

Metoda Runge-Kutta

Sipas metodës *Runge-Kutta* të rendit të katërt, për pikën (x_i, y_i), fillimisht llogariten vlerat e koeficientëve:

$$r_1 = f(x_i, y_i) \cdot h$$
$$r_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{r_1}{2}\right) \cdot h$$
$$r_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{r_2}{2}\right) \cdot h$$
$$r_4 = f(x_i + h, y_i + r_3) \cdot h$$

përmes së cilëve pastaj gjendet pika vijuese (x_{i+1}, y_{i+1}), kështu:

$$x_{i+1} = x_i + h$$
$$y_{i+1} = y_i + \frac{1}{6}\{r_1 + 2r_2 + 2r_3 + r_4\}$$

ku hapi i diferencimit llogaritet me shprehjen:

$$h = \frac{x_p - x_0}{n}$$

nëse kushtet fillestare merren (x_0, y_0) dhe zona e diferencimit $[x_0, x_p]$ ndahet në n-pjesë.

Shembull Zgjidhja e ekuacionit diferencial:

$$\frac{dy}{dx} = x^2 + y$$

përmes metodës *Runge-Kutta*, për kushtet fillestare $x_0=1$ dhe $y_0=1$.

a. Bllok-diagrami

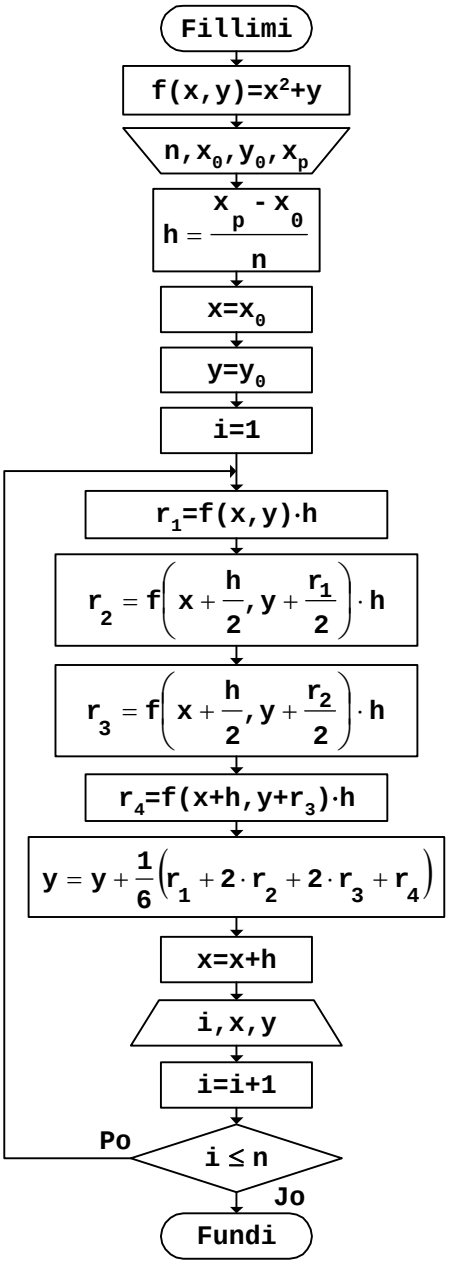


Fig.12.1

b. Programi

```
// Programi Prg12_1
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;
double f(double x,double y);
int main()
{
      const int n=10;
      const double x0=1,y0=1,xp=2;
      int i;
      double x,y,h,r1,r2,r3,r4;
      h=(xp-x0)/n;
      x=x0;y=y0;
      cout << "   i     x          y"
           << "\n-----------------------"
           << "\n";
      cout.setf(ios::fixed);
      for (i=1;i<=n;i++)
      {
            r1=f(x,y)*h;
            r2=f(x+h/2,y+r1/2)*h;
            r3=f(x+h/2,y+r2/2)*h;
            r4=f(x+h,y+r3)*h;
            y=y+(r1+2*r2+2*r3+r4)/6;
            x=x+h;
            cout.width(4);
            cout << i;
            cout.precision(2);
            cout.width(8);
            cout << x;
            cout.precision(5);
            cout.width(11);
            cout << y
                 << "\n";
      }
  return 0;
}

// Nënprogrami
double f(double x,double y)
{
  return pow(x,2)+y;
}
```

Pas ekzekutimit të programit, rezultatet që shtypen në ekran duken:

```
   i      x           y
------------------------
   1    1.10    1.22103
   2    1.20    1.48842
   3    1.30    1.80915
   4    1.40    2.19095
   5    1.50    2.64233
   6    1.60    3.17271
   7    1.70    3.79251
   8    1.80    4.51324
   9    1.90    5.34761
  10    2.00    6.30968
```

Metoda Kutta-Merson

Gjatë zgjidhjes së ekuacioneve diferenciale sipas metodës Kutta-Merson, vlera e pikës vijuese të funksionit y_{i+1} caktohet duke shfrytëzuar vlerën e pikës paraprake y_i:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_4 + k_5)$$

ku koeficientët e veçantë llogariten përmes shprehjeve:

$$k_1 = f(x_i, y_i) \cdot h$$

$$k_2 = f\left(x_i + \frac{h}{3}, y_i + \frac{k_1}{3}\right) \cdot h$$

$$k_3 = f\left(x_i + \frac{h}{3}, y_i + \frac{k_1 + k_2}{6}\right) \cdot h$$

$$k_4 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{8} + \frac{3}{8}k_3\right) \cdot h$$

$$k_5 = f\left(x_i + h, y_i + \frac{k_1}{2} - \frac{3}{2}k_3 + 2k_4\right) \cdot h$$

Nëse kushtet fillestare merren (x_0, y_0) dhe zona e diferencimit $[x_0, x_p]$ ndahet në n-pjesë, hapi përkatës i diferencimit është:

$$h = \frac{x_p - x_0}{n}$$

Shembull

Zgjidhja e ekuacionit diferencial:

$$\frac{dy}{dx} = \frac{1}{2}xy$$

përmes metodës *Kutta-Merson*, për kushtet fillestare $x_0=1$ e $y_0=1$, si dhe $x_p=1$ e $n=10$.

a. Bllok-diagrami

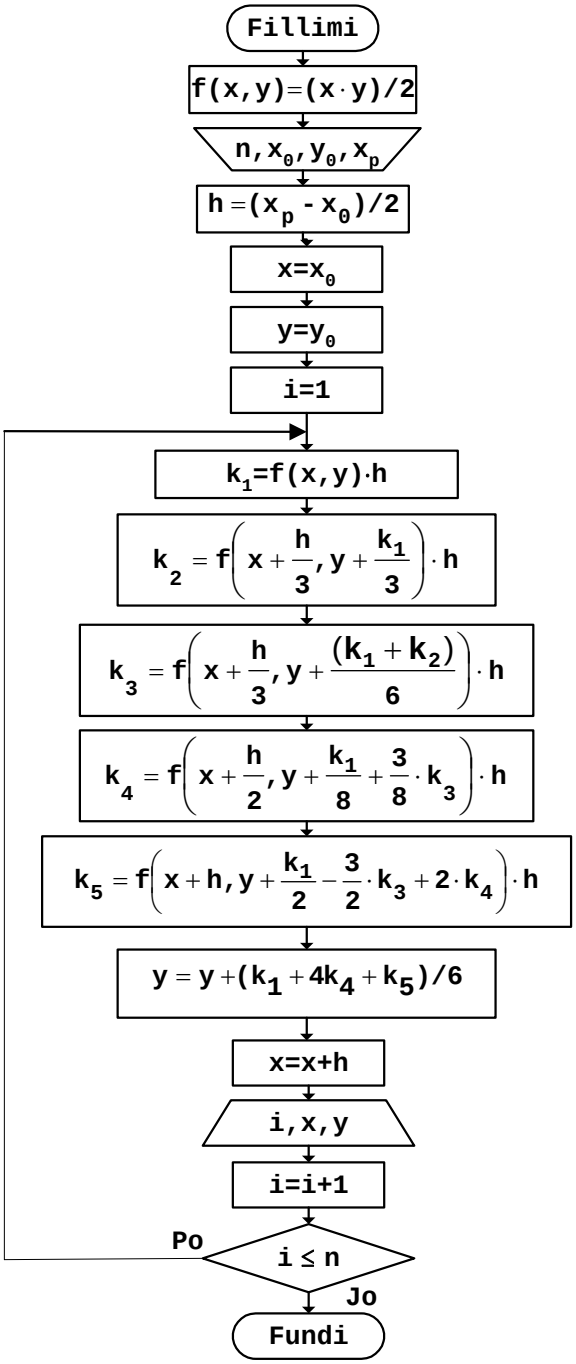


Fig.12.2

b. Programi

```
// Programi Prg12_2
#include <iostream>
#include <iomanip>
using namespace std;
double f(double x,double y);
int main()
{     const int n=10;
      const double x0=0,y0=1,xp=1;
      int i; double x,y,h,k1,k2,k3,k4,k5;
      h=(xp-x0)/n; x=x0;y=y0;
      cout << "   i     x          y"
           << "\n-----------------------" << "\n";
      cout.setf(ios::fixed);
      for (i=1;i<=n;i++)
      {     k1=f(x,y)*h;
            k2=f(x+h/3,y+k1/3)*h;
            k3=f(x+h/3,y+(k1+k2)/6)*h;
            k4=f(x+h/2,y+k1/8+(3*k3)/8)*h;
            k5=f(x+h,y+k1/2-(3*k3)/2+2*k4)*h;
            y=y+(k1+4*k4+k5)/6;
            x=x+h;
            cout.width(4); cout << i;
            cout.precision(2); cout.width(8);
            cout << x;
            cout.precision(5); cout.width(11);
            cout << y << "\n";
      }
  return 0;
}
// Nënprogrami
double f(double x,double y)
{
  return (x*y)/2;
}
```

Pas ekzekutimit të programit, vlerat numerike të cilat fitohen si zgjidhje e ekuacionit diferencial, në ekran do të shtypen kështu:

```
   i     x          y
------------------------
   1    0.10    1.00250
   2    0.20    1.01005
   3    0.30    1.02276
   4    0.40    1.04081
   5    0.50    1.06449
   6    0.60    1.09417
   7    0.70    1.13032
   8    0.80    1.17351
   9    0.90    1.22446
```

```
  10     1.00     1.28403
```

Agni H. Dika
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike
Prishtinë

ALGORITMET

njohuri themelore
me programe në gjuhën C++

Lektor
Dr. Ilaz Metaj

Kopertina
AfiDesign

Shtypi
Adea
Prishtinë