

# Gjuhë Programuese C++ Ushtrime

Avni Rexhepi

Avni Rexhepi



# Ushtrime në C++

Prishtinë, 2010

# Hyrje në programim

- Në kompjuter, të dhënat ruhen dhe përpunohen si vargje të shifrave binare: 1 dhe 0, të koduara në një **kod binar** (ang. binary code).

- Programet dhe Programimi

Programi paraqet një listë të urdhërave (instruksioneve, komandave) të cilat i tregojnë kompjuterit që të kryejë veprime të ndryshme.

- Gjuhët Programuese:

- Gjuhët e ulëta programuse (kodi i makinës) – 0, 1
- Asembleri – mnemonic (ADD, SUB, MOV...)

- Gjuhët e larta programuese

- Fortran, Basic, Pascal, C++, etj.

- Idea --> përshkrimi --> programi --> testimi --> përdorimi

## Interpreterët dhe kompajlerët.

**Interpreteri** – interpreton me radhë urdhërat e programit, sa herë që ekzekutohet programi.

**Kompajler** (ang. compiler) – verifikon tërë programin (sintaksën e gjuhës) dhe e “përkthen” kodin burimor në kod objektiv.

**Program burimor** (ang. source program, **source code**) - *fajll burimor* (ang. source file).

**Program objektiv** (ang. object program, **object code**) - *fajll objektiv* (ang. objekt file).

**Linker** --> i lidhë pjeset (fajllat) e programit dhe krijon *programin ekzekutiv* (ang. executable program, **exe file**) - *fajll ekzekutiv* (ang. exe file).

Termet themelore që i hasim:

Identifikatorët

Variablat

Deklarimi

Inicializimi

Konstantet

Literalet

Operatorët

Komentet

# Hyrje në C++

*Dikush ka thënë: "Mënyra më e mirë për ta mësuar programimin është programimi", pra përmes shembujve dhe provave të ndryshme.*

## ISO C++ Standard

Gjuha programuese C++ dhe Libraria Standarde e C++ janë përshkruar në ISO Standardin e publikuar në vitin 1998: ISO/IEC 14882:1998(E)

Standardi është një dokument i madh, i cili përshkruan gjuhën dhe libraritë deri në detaje (gjë që për shumë kënd është e mërzitshme dhe shumica e njerëzve nuk kanë nevojë për të). Megjithatë, ata që synojnë të mësojnë standardin, së paku duhet të kenë një libër reference, si p.sh. libri i Bjarne Stroustrup – The C++ Programming Language (B. Stroustrup – njihet si krijues i gjuhës C++).

C++ është gjuhë e kompajluar. Kompajleri është një program i cili e lexon kodin burimor, të shkruar nga programeri dhe nga ai e krijon një fajll ekzekutiv ose binar në formatin në të cilin mund të lexohet dhe të ekzekutohet nga kompjuteri. Fajlli burimor është një fajll që përmbanë kodin e shkruar në C++. Fajlli ekzekutiv përbëhet prej kodit të makinës, vargut të 1-sheve dhe 0-ve të cilat nuk janë të parapara që të jenë të përdorshme prej shfrytëzuesit, por janë të kuptueshme vetëm nga kompjuterët.

Le të fillojmë pra me një shembull të thjeshtë:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Programimi ne C++ ";
    return 0;
}
```

# - simboli për direktivat preprocesorike (urdhëresa, para fillimit të procesimit të programit)  
# **include** (përfshijë, përmbajë) – C++ i ka libraritë e urdhërave dhe fajllat e hederit (header files – angl. head – koka, kreu), të cilat i përfshinë në programe, përmes direktivës #include.

## Header fajlat – Fajlat e kreut

ISO C++ Standardi ka përcaktuar mënyrën e përfshirjes së fajllave të kreut (Header Files) në programe. Sipas standardit, përdoret mënyra: <iostream> dhe <vector> në vend të formës së vjetër: <iostream.h> ose <vector.h>

Gjithashtu: <cstdlib> dhe <stdlib> në vend të atyre: <stdio.h> dhe <stdlib.h>

Nëpër literaturë mund të gjenden të dy format, por preferohet që të përdoren metodat e reja, sepse me kohë, kompajlerët e rinj në treg nuk do t'i përkrahin më fajlat e vjetër.

Si pjesë e kompajllimit, kompajleri i C++-it e ekzekuton një program të quajtur preprocesor i C++.

Preprocesori është në gjendje që të shtojë dhe të largojë kod nga fajlli burimor. Direktiva #include i tregon preprocesorit që të përfshijë kodin nga fajli <iostream>. Ky fajll i përmbanë deklaratimet për funksionet që i nevojiten programit.

**using namespace std;** - definimi i përdorimit të objekteve dhe funksioneve nga libraria standarde, pa pasur nevojë shkruarjen e kualifikatorëve të veçantë (eksplisit).

C++ përkrahë konceptin e hapësirave emërtuese (name spaces). Në esencë, kjo i lejon variablat që të lokalizohen përbrenda kodit. Komanda using namespace std; i lejon të gjitha objektet dhe funksionet nga libraria standarde që të përdoren në program, pa kualifikator eksplisit. Namespace është një koncept i avansuar, kështu që për fillim kryesorja është që të dihet se duhet të shkruhet ky rresht, për të lejuar përdorimin e thjeshtë të librarisë standarde.

**iostream** – Input Output Stream (Rrjedha hyrëse, dalëse) – përmbanë urdhërat për hyrje/dalje.

**main ( )** – funksioni themelor, kryesor ose si thuhet me shpesh, programi kryesor.

{ - kllapa e madhe e hapur, tregon fillimin e programit kryesor

**cin** – operatori për lexim.

**cout** – operatori për shtypje.

**return** (kthe, kthimi) – kthimi i rezultatit të funksionit.

} – kllapa e madhe e mbullur tregon fundin e programit

Çdo urdhër i programit përfundon me simbolin “;” (pikëpresje).

#### Shikimi i rezultatit në dalje (në ekran)

Ka disa ambiente programuese, si p.sh., consola dalëse nën Microsoft Windows, në të cilat dritarja e rezultateve në dalje do të zhduket menjëherë pas përfundimit të programit, duke pamundësuar shikimin e rezultateve dalëse. Për të evituar këtë problem, në fund të programit mund të shtoni një urdhër, si

```
cin.get();
```

ose në disa raste duhet dy rreshtat vijues:

```
cin.ignore(1000, '\n');
```

```
cin.get();
```

Kjo bën që dalja e programit të jetë e dukshme deri sa të shtypet ndonjë tast ose tasti Enter.

`cin.ignore(1000, '\n');` - Përdoret edhe për të pastruar rrjedhën e të dhënave paraprake, ashtu që kur të lexohet ose shtypet vlera e re, baferi të jetë i zbrazët dhe i pastër, sepse mund të ndodhë që të shtypen të dhënat e vjetra të baferuara më herët. Kjo komandë bën që të injorohen 1000 karakteret paraprake ose e tërë rrjedha deri tek karakteri “\n” (pra kalimi në rreshtë të ri, gjegjësisht shtypja e ENTER në tastierë gjatë dhënies së vlerave me tasierë), cilado që paraqitet e para.

Funksioni `main()` është pika startuese e ekzekutimit të çdo programi në C++. Por, kjo nuk është tërësisht e vërtetë në të gjitha rastet. Nëse shkruhet programi që do të ekzekutohet në ambient të “pavarur”, nuk duhet patjetër funksioni `main()`. Me përjashtim të këtij rasti, programi patjetër duhet ta ketë funksionin `main()`. Standardi ISO tregon dy mënyrat vijuese të cilat duhet të pranohen nga cilido kompajler i C++:

```
int main()
```

```
int main(int argc, char *argv[])
```

Definicioni i parë është standard. Definicioni i dytë i merr dy parametra të funksionit. I pari përmbanë numrin e argumenteve të linjës komanduese që i përcillen programit nga procesi thirrës (ai që e thërret programin për ekzekutim), i dytë është një varg i parametrave të linjës komanduese.

Standardi i C++ lejon kompajlerin që të pranojë definicionet e `main()` me parametra shtesë, por kjo është e varur prej implementimit specifik dhe nuk është portabile në përgjithësi.

#### Kthimi nga funksioni `main()`

Urdhëri “return” për kthim të rezultatit nga funksioni `main()` është opcional (zakonisht mirret: return 0;). Mirëpo kompajlerët e vjetër dhe disa kompajlerë aktual nuk pranojnë programet `main()` që nuk kanë urdhërin return, prandaj në shumicën e rasteve ai shkruhet.

Gjëja e parë që vërehet, është se **main()** kthen një integer (deklarimi: `int main()`). Vlera e tipit integer reprezenton statusin e kthyer për procesin thirrës (sistemin operativ).

Mund të kthehet çfarëdo vlere, por vlerat e vetme portabile janë 0 dhe `EXIT_SUCCESS`, të cilat përfaqësojnë ndërprerjen (përfundimin) normal dhe `EXIT_FAILURE`, që përfaqëson përfundimin jonormal. `EXIT_SUCCESS` dhe `EXIT_FAILURE` janë të definuara në header-in `<cstdlib>`. Cilado vlerë tjetër e kthyer, ia kthen procesit thirrës një vlerë statusi të definuar prej implementimit.

Gjithashtu, disa kompajlerë do të pranojnë edhe definimin e `main()`, si: **void main()**. Megjithatë, kjo është thjeshtë jokorrekte dhe nuk duhet të përdoret.

Çka ndodhë nëse nuk përdoret urdhëri return? Nëse kllapa mbyllëse “}” e funksionit `main()` në fund të programit është e mbyllur pa urdhërin **return**, efekti është sikur të jetë dhënë urdhëri return 0, që do të thotë se return 0; është e nënkuptuar, duke treguar ndërprerjen (përfundimin) normal. Kështu, edhe programi vijues është valid, në C++:

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout << "Jungjatjeta" << endl;
```

```
    cin.get();
```

```
}
```

Megjithatë, disa kompajlerë (sidomos ata më të vjetër) nuk e pranojnë funksionin `main()` pa urdhër return në fund, prandaj ai do të duhet të shkruhet.

**Komentet**

Komentet, janë tekste për shfrytëzuesit, por që nuk kanë ndikim në program. Ato shkruhen sa për të dhënë ndonjë shpjegim ose përshkrim. Kompajlleri i injoron tërësisht.

// Komenti në një rresht

/\*Komenti në disa rreshta, apo  
komenti në bllok \*/

```
//Programi i parë
#include <iostream>
using namespace std;
int main()
{
    /* Permes operatorit për shtypje, cout,
    do të shtypet teksti në ekran */
    cout << "Programimi ne C++ ";
    return 0;
}
```

**Struktura e Programit:**

Kreu i programit  
Direktivat preprocesorike  
Funksionet  
Funksioni main()  
Deklarimi i variablave

Urdhërat e programit

Mbyllja e programit

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

int main()
{
    int a;

    a=5;
    cout << a;
    return 0;
}
```

Kreu i programit - Koment  
Direktivat preprocesorike

Funksioni main()

Deklarimi i variablave

Urdhërat e programit

Mbyllja e programit

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

int main()
{
    int a;

    cin >> a;    /leximi prej tastiere
```

```

    cout << a;    /shtypja në ekran
    return 0;
}

```

```

// Programi cin-cout - Llogaritja e vleres mesatare
#include <iostream>
using namespace std;
int main()
{
    int n1, n2, n3;
    float nm;

    cout<<"Jepni tri notat: " ;
    cin >> n1 >> n2 >> n3;

    nm=(n1+n2+n3)/3;

    cout <<"Nota mesatare="
        << nm
        << "\n";

    return 0;
}

```

Varësisht prej versionit të programit (kompajlerit), mund të paraqiten disa ndryshime në mënyrën e shkruarjes së kreut të programit, kështu që përfshirja e header fajllave mund të bëhet në formatin:

```
#include <iostream> ose #include <iostream.h>
```

me ç'rast, shkruhet emri i fajllit dhe prapashtesa ".h".

Në versionet më të reja, prapashtesa ".h" evitohet, duke shkruar vetëm <iostream>, por gjithsesi deklarohet përdorimi i "namespace-it" standard:

```
using namespace std;
```

Prandaj, të dy format vijuese japin rezultat të njëjtë:

```

#include <iostream.h>
int main()
{
    cout << "Programimi ne C++ ";
    return 0;
}

```

```

#include <iostream>
using namespace std;

int main()
{
    cout << "Programimi ne C++ ";
    return 0;
}

```

Edhe në versionin e Visual Studio 6.0 (Visual C++ 6.0), programi mund të shkruhet në formën e dytë, duke thirrur “namespace-in” standard (std).

### Namespace std

Si u tha edhe më herët, kur i përfshijmë fajlat e kreut (header files) prej librarisë standarde të C++, përmbajtjet ndodhen në “namespace std”. Ekzistojnë tri mënyra për të kualifikuar përdorimin e tyre ashtu që të mund të përdoren në programe:

(a) Duke përdorur direktivën: **using namespace std;** - Kjo është metoda më e thjeshtë, mirëpo i bën të gjithë identifikatorët e namespace std globalisht të disponueshëm.

(b) Duke përdorur deklarin: **using std::cout;** - Kjo është metodë më e kufizuar dhe duhet të përsëritet për gjithçka që dëshirojmë të kualifikojmë për përdorim në program, kështu që programi do të duhet të shkruhet si në vijim:

```
#include <iostream>
using std::cout;
using std::endl;
using std::cin;
int main()
{
    cout << "Jungjatjeta" << endl;
    cin.get();
    return 0;
}
```

(c) Identifikatori eksplicit: **std::cin.get();** - Në këtë rast, përdoret identifikatori eksplicit si prefiks (parashtesë) për secilin përdorim. Programi do të duhej të shkruhet, si:

```
#include <iostream>
int main()
{
    std::cout << "Jungjatjeta" << std::endl;
    std::cin.get();
    return 0;
}
```

### Header-ët e C++

Nëpër programe, shpeshherë nevojite përfshirja e header fajlave (fajlave të kreut) ashtu që të përdoren urdhërat, operatorët, manipulatorët dhe funksionet e gatshme të cilat i ofrojnë libraritë e C++-it.

Kjo do të thotë se duhet të dijmë se çka përmbajë header fajlat, ashtu që të dijmë se çka mund të përdorim nëpër programe. Nëse dështon përfshirja e header-ave të duhet, programi nuk do të kompajlohet.

Libraria e C++ përfshinë header-at prej librarisë standarde të gjuhës C. Ato kanë emrat, si:

**<stdlib.h>, <string.h>, <time.h>**

Ju mund të përdorini këta header-a në programt në C++ por duhet të keni kujdes, pasi që përmbajtjet e tyre nuk ndodhen në namespace std, por ato ndodhen në namespace-in global. Përdorimi i këtyre header-ave të vjetër nuk preferohet në C++, sepse në të ardhmen ata mund të largohen nga standardi.

C++ ofron versionet e reja të këtyre header-ave, përmbajtja e të cilëve ndodhet në namespace std. Header-ët ekuivalent për këta të mësipërm janë:

**<cstdlib>, <cstring>, <ctime>**

Pra, siç mund të shihet, ata nuk e përdorin prapashtesën .h dhe e kanë parashtesën “c”. Ju duhet të përdorni versionet e reja nëpër programe, përveq nëse jeni duke mirëmbajtur ndonjë kod të vjetër të shkruar në C.

C++ gjithashtu prezanton disa header fajla të ri, të cilët nuk kanë ekzistuar në C, si:

**<iostream>, <vector>, <algorithm>**

Si edhe header-at e tjerë të C++, ata nuk e përdorin prapashtesën “.h”, por nuk kanë ndonjë prefiks.

Nëpër literaturë do të hasen versionet e vjetra të header-ave të C++, të përdorur para standardit. Ata kanë emrat si: **<iostream.h>, <fstream.h>** në vend të: **<iostream>, <fstream>**

Këto versione të vjetra nuk janë adresuar prej standardit të C++ fare. Ata nuk përjashtohen, por nuk janë header të C-së e as nuk janë pjesë e standardit të C++. Ata thjeshtë janë header-ë të vjetër, të cilët janë përdorur për një kohë të gjatë, mirëpo preferohet që tash të përdoren versionet e reja, edhe pse kompajlerët sigurisht do të vazhdojnë të përkrahin akoma versionet e vjetra, për pajtueshmëri prapavepruese.

Konfuzion shkakton edhe çështja e përdorimit të të header-it për string. Ekzistojnë tre header-a për string: **<string.h>, <cstring>** të cilët janë të stilit të C-së (vargjet e karaktereve të përfunduara me “null” –



null-terminated character arrays) dhe përmbajnë funksionet për string si: strcpy(), strcat() and strlen() si dhe <string> i cili përdoret për std::string class-at në C++.

Ajo çka është me rëndësi, është se duhet të jeni në gjendje të shkruani kode të cilat e respektojnë C++ standardin, gjë që do të thotë se mund t'i kompajloni në cilindo kompajler modern të C++-it.

## Variablat, tipet e tyre

- Të dhënat (ang. Data)

Tipi	Rangu i vlerave	madhësia në bajta
unsigned short <a href="#">int</a>	0..65535	2
short <a href="#">int</a>	-32768..32767	2
unsigned <a href="#">long int</a>	0..4294967295	4
<a href="#">long int</a>	-2147483648..2147483647	4
<a href="#">int</a> - 16 bit	-32768..32767	2
<a href="#">int</a> - 32 bit	-2147483648..2147483647	4
unsigned <a href="#">int</a> - 16 bit	0..65535	2
unsigned <a href="#">int</a> - 32 bit	0..4294967295	4
float	1.2E-38..3.4E38	4
<a href="#">double</a>	2.2E-308..1.8E308	8
<a href="#">char</a>	256 simbole	1

## Deklarimi

Variabla deklarohet, duke deklaruar tipin dhe emrin:

**tipi****Variablës** **emriVariablës**, si p.sh.:

```
int a;
double x;
float z;
short int koha;
char g;
char Emri[20];
/* Ne C++ nuk ka variabel tekstuale - String, keshtu qe Stringu
krijohet si varg i karaktereve te vecanta*/
```

```
int x,y;
long int dita,e,f3;           //Deklarimi i perbashket

double x,h; int g; float a,p; //Deklarimet ne nje rresht

int A[8];                     //Vektori

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

**Indekset fillojnë me vlerën zero**

```
int A[6]; //6 anëtar - A[0], D[1], ..., D[5].
```

## Deklarimi dhe inicializimi

Nëse gjatë deklarimit, variablës i ndahet edhe vlera fillestare (inicializuese), atëherë thuhet se kemi bërë deklarimin dhe inicializimin njëkohësisht.

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

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

char Z[7]={'d','4','*','a','G','$'}; //Vektori me karaktere

const double PI=3.1415926;

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

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

## Tipet, identifikatorët dhe fjalët e rezervuara

### Tipi

Kur dëshironi të ruani të dhëna në një program në C++, si numri i plotë ose ndonjë karakter, duhet t'i tregojmë kompajlerit se cilin tip të të dhënave dëshirojmë ta ruajmë. Tipi do të ketë karakteristikat, si rangun e vlerave të cilat mund të ruhen dhe operacionet të cilat mund të kryhen në variablat e atij tipi.

### Tipet themelore

C++ ofron tipet themelore të brendshme (built-in): Boolean, character, integer dhe floating-point (pikë e lëvizshme). Gjithashtu mundëson krijimin e tipeve të shfrytëzuesit duke përdorur enumeration (grupimin) dhe class (klasa).

Për secilin prej tipeve themelore rangun e vlerave dhe operacioneve që mund të kryhen në to është i përcaktuar prej kompajlerit. Secili kompajler duhet të ofrojë operacionet e njëjta për tipin e veçantë por rangun e vlerave mund të ndryshojë ndërmjet kompajlerëve të ndryshëm.

### Bool

Vlera e tipit Bool-eane (të Bultit) mund të kenë vlerën true (e saktë) ose false (e pasaktë). P.sh.,

```
bool tek = false;
bool VlerauGjet = true;
```

Nëse vlera buleane konvertohet në numër të plotë, atëherë true bëhet 1 kurse false bëhet 0.

Nëse një vlerë numerike e plotë konvertohet në vlerë Bool-eane, atëherë vlera 0 bëhet false kurse çfarëdo vlere jo-zero bëhet true (e saktë).

### Character

Tipi karakter përdoret për të ruajtur shkronjat, shifrat dhe simbolet (zakonisht ASCII karakteret, por jo gjithmonë). P.sh.,

```
char menyja = 'q';
char vleraHyrese = '3';
char SimboliPerqind = '%';
```

Vëreni se si karakteri futet në thonjëza të njëfishta (apostrofe). Variablat e tipit karakter mund të ju ndahen edhe vlerat numerike:

```
char chNumri = 26;
```

Mund të deklarohen karakteret me shenjë (signed) ose pa shenjë (unsigned), ku karakteret me shenjë mund të kenë vlerë pozitive, negative ose zero, kurse karakteret pa shenjë mund të kenë vetëm vlerë pozitive ose zero.

```
signed char VleraIme = 100;
signed char VleraRe = -43;
unsigned char Karakteri2 = 200;
```

Nëse përdoret tipi char i thjeshtë, as me shenjë, as pa shenjë:

```
char Vlera = 27;
```

a jo mund të ndryshojë ndërmjet kompajlerëve, ashtu që mund të ketë sjellje si të karakterit me shenjë ose pa shenjë. Në disa kompajlerë ai mund të pranojë vlerë pozitive, negative ose zero kurse në disa mund të pranojë vetëm vlera pozitive dhe zero. Madhësia e tipit char është gjithmonë një bajt, që është e garantuar të jetë së paku 8 bita. C++ gjithashtu ofron edhe tipin wchar\_t, tip i zgjeruar karakter, zakonisht i përdorur për bashkësi të zgjeruara (të mëdha) të karaktereve.

Një varg i karaktereve mund të përdoret që të përmbajë string të stilit të C-së në C++, p.sh.:

```
char aString[] = "Ky eshte string i C-stilit";
```

C++ gjithashtu ofron edhe klasën string që ka përparësi ndaj vargjeve të karaktereve.

### Integer

Tipi integer përdoret për ruajtjen e numrave të plotë. Mund të përdoren tipet signed (me shenjë), unsigned (pa shenjë) ose i thjeshtë:

```
signed int indeksi = 4198;
signed int temperatura = -32;
unsigned int numri = 0;
int lartesia = 100;
int vlera = -67;
```

Sikur në rastin e karaktereve, numrat me shenjë mund të përmbajnë vlera negative, pozitive ose zero kurse ata pa shenjë vetëm vlera pozitive ose zero. Sidoqoftë, numrat e thjeshtë mund të përmbajnë cilat do vlera, pasi që gjithmonë e kanë shenjën. Numrat integer mund të deklarohen në formë të shkurtër, pa fjalën e rezervuar int:

```
signed indeksi = 4198;
unsigned numri = 0;
```

Vlerat Integer paraqiten në tri madhësi: plain int (int i thjeshtë), short int (int i shkurtër) dhe long int (int i gjatë).

```
int normal = 1000;
short int vleraVogel = 100;
long int vleraMadhe = 10000;
```

Rangu i vlerave të këtyre tipeve është i definuar prej kompajlerit. Zakonisht, tipi i thjeshtë int ka rang më të gjerë se short int, kurse long int më të gjatë se ai i thjeshtë, edhe pse nuk ndodhë kështu gjithmonë.

Ajo që është e sigurtë, është se tipi i thjeshtë do të jetë së paku i madh sa short int ose më i madh dhe long int do të jetë së paku i madh sa i thjeshti ose më i madh. Tipi short int është së paku 16 bita, kurse long int së paku 32 bita.

Vlerat short int dhe long int mund të definohen edhe në formë të shkurtër, pa fjalën e rezervuar int:

```
short vleraVogel = 100;
long vleraMadhe = 10000;
```

Mund të kemi vlera long int dhe short int me shenjë ose pa shenjë:

```
unsigned long vleraPozitive = 12345;
signed short = -7;
```

### Floating-Point

Tipet e numrave me pikë të lëvizshme mund të përmbajnë numra decimal, si p.sh: 1.23, -.087. Ekzistojnë tri madhësi: float (single-precision – me precizitet të njëfishtë), double (double precision – me precizitet të dyfishtë) dhe long double (extended-precision – precizitet i zgjeruar). P.sh.:

```
float celsius = 37.623;
double fahrenheit = 98.415;
long double xhirollogaria = 1897.23;
```

Rangu i vlerave që mund të ruhen në secilën prej tyre është i definuar prej kompajlerit. Zakonisht tipet double do të ruajnë vlera më të mëdha se rangu i float dhe long double do të ruajnë vlera më të mëdha se double (edhe pse kjo nuk është gjithmonë e vërtetë). Sidoqoftë, është e sigurtë se double do të jetë së paku i madh sa float por mund të jetë më i madh, dhe long double do të jetë i madh së paku sa long por mund të jetë edhe më imadh.

### Enumeration (numeracioni - grupi)

Një numeracion (grup) është tip i definuar prej shfrytëzuesit që i mundëson shfrytëzuesit që të definojë rangun e vlerave për tipin e dëshiruar. Konstantet e emëruara (anëtarët e grupit) përdoren për të paraqitur vlerat e një numeracioni (grupi), si p.sh.:

```
enum ditaeJaves {eHene,eMarte,eMerkure,eEnjte,ePremte,eShtune,eDiele};
ditaeJaves ditaeSotme = eMerkure;
if(ditaeSotme == eMarte)
{
// urdherat...
}
```

Vlerat e nënkuptuara të përcaktuara për konstantet e numeracionit janë zero-based (me bazë zero), kështu që, për shembullin paraprak:

```
eHene == 0, eMarte == 1, etj.
```

Shfrytëzuesi mund të përcaktojë vlera të tjera për ndonjërin anëtarë të grupit, dhe vlerat vijuese do të kenë vlerë për një më të madhe. P.sh.

```
enum frutet {molla=3, bananja=7, portokalli, dardha, kiwi};
keshtu, që portokalli do të ketë vlerën 8, dardha 9, kiwi 10, etj.
```

### Class - Klasat

Klasa i mundëson shfrytëzuesit krijimin e tipeve më të sofistikuar të të dhënave, për të modeluar objektet dhe veprimet nga jeta e përditshme. Pra, klasa është një konstruksion softverik që mund të përdoret për simuluar ndonjë objekt të jetës reale. P.sh., modeli softverik i një veture, klasa “vetura”, mund të përmbajë të dhëna për tipin e veturës, si: “tipi”, “ngjyra”, “harxhimi” etj., dhe aftësitë (aftësi vepruese, operative) si “përsheptimi”, “ngadalësimi”, “frenimi”, etj. Klasa është tip i definuar i shfrytëzuesit që i ka të dhënat, anëtarët (members) dhe aftësitë (metodat). Objekti është një instancë (shembull, rast) e klasës. P.sh., për të krijuar klasën katrori e cila ka të dhënat (vlerat) për madhësi dhe ofron operacionet vizato dhe ndryshomadhësinë:

```
class katrori {
public:
katrori();
~katrori();
void vizato();
bool ndryshoMadhesine(int madhesiaeRe);
private:
int madhesia;
};
```

Diskutimi për klasat, i tejkalon kufinj të asaj çka mësohet në këtë semestër dhe nuk është përfshirë në këtë tekst.

## Identifikatorët dhe fjalët e rezervuara të C++

Në C++ ne i krijojmë emrat për entitetet që i krijojmë: variablat, funksionet dhe tipet e deklaruara. Këta emra ose si quhen Identifikatorë, duhet t'i respektojnë disa rregulla. Një identifikatorë duhet të fillojë me shkronjë dhe të përbëhet nga një varg i pandërprerë i shkronjave dhe numrave. Megjithatë, lejohet përdorimi i “nënvizës” (underscore) “\_” si lidhëse, e cila në këtë kontekst konsiderohet si “shkronjë”. Nuk ka kufizime sa i përket gjatësisë së identifikatorit.

Identifikatorët që fillojnë me nënvizë të pasuar me shkronjë të madhe janë të rezervuar nga implementimi, ashtu siç janë edhe ata me dy nënviza, kështu që për të evituar problemet duhet të evitoni përdorimin e tyre. Gjithashtu, nuk mund të përdoren si identifikatorë fjalët e rezervuara në C++, të cilat janë:

`and, and_eq, asm, auto, bitand, bitor, bool, break, case, catch, char, class, compl, const, const_cast, continue, default, delete, do, double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace, new, not, not_eq, operator, or, or_eq, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, while, xor, xor_eq.`

Gjithashtu duhet të provoni të evitoni emrat nga libraritë e C++-it, si: `abs, swap, max` etj. C++ është “case sensitive” ( i ndjeshëm në madhësi të shkronjave, pra bën dallim mes shkronjës së madhe dhe asaj të vogël) kështu që shkronja e vogël dallon prej asaj të madhe. Kjo do të thotë, që emrat:

“Numri” dhe “numri” ose “ditaEjaves” dhe “ditaEjaves” dallojnë mes veti dhe do të trajtohen si dy identifikatorë të ndryshëm.

Shembuj të identifikatorëve të pranueshëm janë:

`llogaria, numri_i_pare, lexoVleren, kanali25, po`

Shembuj të identifikatorëve të papranueshëm:

`numri i pare, delete, 2shkronja, _TUNG_`

### Këshilla dhe udhëzime për Identifikatorët

- Përdorni emra me kuptim dhe përshkruar për ta bërë kodin lehtë të kuptueshëm, si p.sh.:  
`int lartësia` në vend të `int l, char menya` e jo, `char m, int mosha_ePacientit` e jo `int nr.`

- Evitoni emrat që dallojnë vetëm për nga madhësia e shkronjave

`lartesia, Lartesia`

- Evitoni përdorimin e fjalëve të rezervuara, duke i ndryshuar vetëm në shkronjën e parë të madhe

`Return, Continue.`

- Mundohuni të ruani stilin, si p.sh.:

- Shkronja e madhe për secilen pjesë, përveq asaj të parës: `lexoEmrin()`,  
`merrVlerenMaksimale()`.

- Përdorni nënvizën për të ndare fjalët: `lexo_Emrin()`, `mer_Vleren_Maksimale()`.

- Përdorni prefiksin për identifikim të tipit: `int nNumeriPlote`, `char stEmri[]`.

## Leximi dhe Shtypja

- **cout** - Për shtypje të teksteve në ekran, përdoret operatori **cout**, i cili është i definuar në header fajllin <iostream> (“c out” (lexohet “si aut”), si simbolikë që nga C++ shtypet jashtë).

- **cin** - Për marrjen (leximin) e vlerave prej tastierës, përdoret operatori **cin**, gjithashtu i definuar në header fajllin <iostream> (“c in” (lexohet “si in”), si simbolikë, që C++ merre brenda).

Për manipulim me tekstin, C++ përdorë buffer-in (baferin, regjistrin për tekst), i cili zbrazet me urdhërin **flush** (pastrim, zbrazje), që menjëherë e dërgon përmbajtjen e baferit në dalje.

Kursori kalon në rreshtin e ri me njërën nga format: **cout<<”\n”**; ose **cout<<endl**; . Me “\n”, nuk zbrazet baferi, kurse **endl** e zbrazë baferin dhe e kalon kursorin në rresht të ri.

- **cerr** – shërben për shtypje të pa-baferuar në pajisjen standarde për gabime (standar error device), që nënkuptohet të jetë ekрани. I pa-baferuar, do të thotë se çfarëdo mesazhi ose e dhëne do të shkruhet menjëherë në dalje, pa u ruajtur në buffer së pari. Me hyrjen e baferuar, të dhënat së pari ruhen në bafer nga sistemi operativ, në mënyrë transparente për programin. Kur baferi të mbushet, të gjitha të dhënat shkruhen “jashtë” në dalje. Kjo është më efikase, sepse çdo shkruarje kërkon një “overhead” nga sistemi operativ. Duke shkruar bafere të mëdha, kërkohet më pak “overhead” sesa për shkruarjen e mesazheve të vogla. Ana negative është se nëse programi “dështon” para se të shkruhet përmbajtja e baferit, asgjë nga ai nuk do të paraqitet në dalje. Paraqitja në dalje, përmes **cerr** është e pabaferuar, për të siguruar se mesazhet e gabimeve do të shkruhen në dalje.

- **clog** – mundëson paraqitjen e baferuar në pajisjen standarde për gabime (ekran).

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

## Rezervimi i hapësirës

Për shtypje të formatizuar, rezervimi i vendeve bëhet me **cout.width** (ang. width – gjerësia) **cout.width(k)**; // rezervon shtypjen në k-kolona

```
cout.width(8); // shtypja e rezultatit ne 8 kolona
cout << a; // p.sh., nese shtypet variabla a, qe ka vleren 15
// rezultati do te duket si ne vijim
```

```
□□□□□15 // ku □ paraqet zbrazëtiren, rreshtimi (plotesimi i
// vendeve te rezervuara) behet nga skaji i djathte
```

E njëjta mund të bëhet edhe duke përdorur manipulatorin **setw(x)** (angl. set width – përcakto gjerësinë), i cili ndodhet në librarinë <iomanip> (Input-Output manipulation).

```
....
#include <iomanip>
```

```
...
setw(8);
cout<<a:
....
```

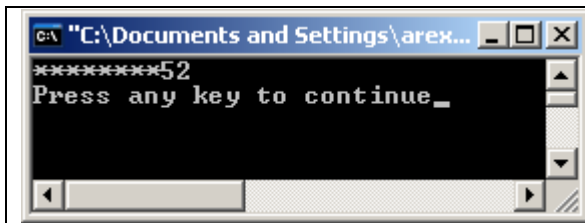
## Mbushja me mostër

Mbushja e pozitive të zbrazëta, mund të bëhet me ndonjë simbol, i cili duhet të jepet brenda thojzave të njëfishta (apostrofeve), në urdhërin cout.fill (ang. Fill – mbushe).

```
cout.fill('f');          // mbushja me simbolin 'f'
```

```
// Programi shtypja me mbushje
#include <iostream>
using namespace std;
int main()
{
    int x=52;
    cout.width(10);
    cout.fill('*');
    cout << x
        << "\n";
    return 0;
}
```

rezultati ne ekran do te duket si:



Për të dalluar me lehtë pjesën e “përshkruar” nga pjesa e shtypjes së vlerave që llogariten (dhe mirren) prej programit, shpeshherë rezultatet shtypen si në vijim:

```
cout << "Vlera e lexuar a="
    << a
    << "\n";
cout << "Vlera e lexuar a="
    << a
    << endl;
```

Pra, çkado që shkruhet brenda thonjzave përsëritet ashtu si është shkruar, kurse pjesa (variablat) jashtë thonjzave mirren prej programit. Ndarja e këtyre pjesëve bëhet me shenjën e ridrejtimt të rrjedhës: “<<”.

Për të arritur rezultatet në dalje, me format të caktuar, përdoren edhe karakteret speciale, të cilat e braktisin rrjedhën normale hyrëse/dalëse, prandaj edhe quhen Escape-Sequence characters (Karakteret për braktisje të sekuencës/rrjedhës).

**Escape-sequence characters (karakteret speciale)**

\a	Alarm
\b	Backspace
\f	Form Feed (për printer)
\n	New line (CR+LF) (rreshti i ri)
\r	CR
\t	Tab
\v	Vertical Tab
\\	\
\?	?
\'	'
\"	"
\000	Numër oktal
\xhh	Numër heksadecimal
\0	Null zero (zero binare)

\xhh, mund të përdoret për të shtypur në ekran karakteret të cilat nuk i kemi në tastierë, si p.sh. shkronja “ë” dhe “ç”, të gjuhës shqipe. ASCII kodi përkatës i karakterit, duhet të shkruhet në formën heksadecimale. P.sh. për “ë”, kodi është: “137” (Si e shtypim zakonisht në tastierë, ALT+137). Atëherë, 137 në bazën 16 (numër heksadecimal), është:  $89 (8 \cdot 16^1 + 9 \cdot 16^0 = 137)$ . Prandaj, për ta shtypur në dalje shkronjën “ë”, duhet të shkruhet: “\x89”. P.sh., pwr tw shtypur nw dalje:

“Sot është ditë e mirë” – në C++ duhet të shkruhet - “Sot \x89sht\x89 dit\x89 e mir\x89”

```
cout << "Sot \x89sht\x89 dit\x89 e mir\x89" ;
```

(Për ta shkruar më lehtë dhe më shpejtë, e shkruani tekstin me „w“ (ose ndonjw shenjw tjetwr) në vend të shkronjës „ë“ (si veprohet zakonisht), e pastaj me: *Edit – Replace*, e zëvendësoni: „w“ me „\x89“).

**Preciziteti**

Për të përcaktuar precizitetin e shtypjes për vlerat reale (me presje dhjetore), përdoret urdhëri për precizitet:

```
cout.precision(k); // shtypja me precizitet te caktuar, me k-shifra
```

ose manipulatori **setprecision(x)**; i cili ndodhet në <iomanip>.

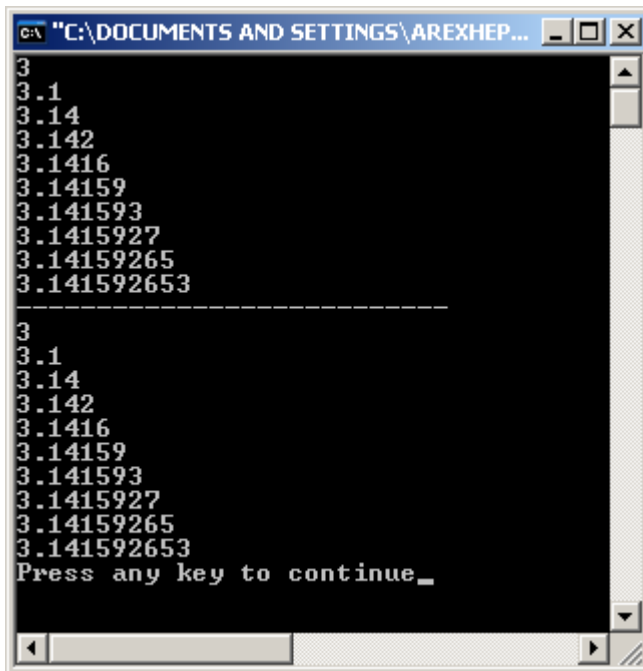
```
// Programi shtypja me precizitet
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
    int i;
    double x=3.1415926534;
    for (i=1;i<=10;i++)
    {
        cout.precision(i);
        cout.width(i);
        cout << x << "\n";
    }
    //ose
```



```

    cout<<"-----\n";
    for (i=1;i<=10;i++)
    {
        cout<< setprecision(i) << setw(i) << x << "\n";
    }
    return 0;
}

```



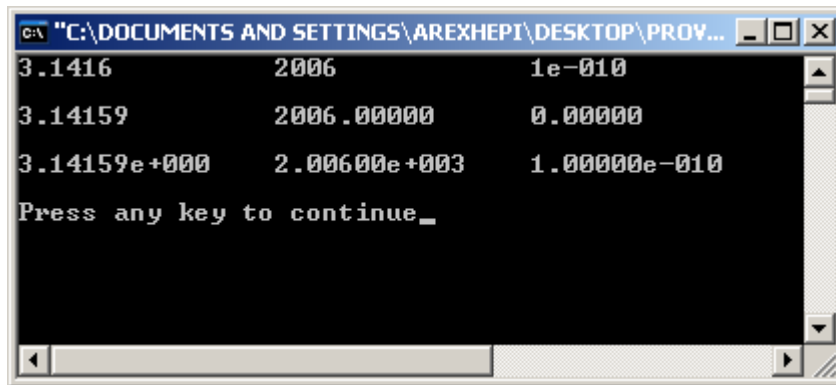
Përndryshe, në kombinim me shtypjen me precizitet mund të definohet formati për shtypjen e bazës së vlerës në formatin **fixed** (fiks), **scientific** (shkencor) ose e papërcaktuar:

```

// shtypja me format dhe precizitet
#include <iostream>
using namespace std;

int main () {
    double a,b,c;
    a = 3.1415926534;
    b = 2006.0;
    c = 1.0e-10;
    cout.precision(5);  \\preciziteti, shifrat pas presjes dhjetore
    cout << a << "\t\t" << b << "\t\t" << c << endl<<endl;
    cout << fixed      << a << "\t\t" << b << "\t" << c << endl<<endl;
    cout << scientific << a << "\t" << b << '\t' << c << endl<<endl;
    return 0;
}

```



```

C:\ "C:\DOCUMENTS AND SETTINGS\AREXHEPT\DESKTOP\PROV...
3.1416      2006      1e-010
3.14159     2006.000000  0.000000
3.14159e+000 2.00600e+003  1.000000e-010
Press any key to continue_

```

## Sistemi numerik

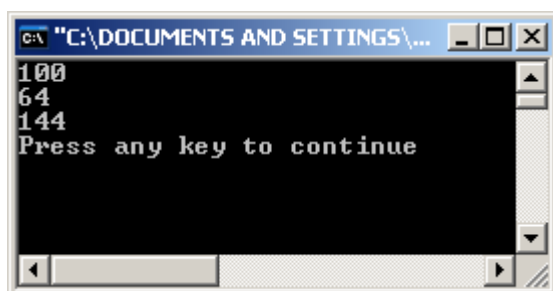
Rezultatet mund të shtypen në sistemet numerike me bazë të ndryshme, si decimal (baza 10), heksadecimal (baza 16), octal (baza 8), etj, duke përdorur para shtypjes së vlerës, urdhërin për përcaktimin e formatit përkatës: **dec**, **hex** apo **oct**.

```

// modifikimi i bazës së numrave
#include <iostream>
using namespace std;

int main ()
{
    int n;
    n=100;    //vlera decimale 100
    cout << dec << n << endl;
    cout << hex << n << endl;
    cout << oct << n << endl;
    return 0;
}

```



```

C:\ "C:\DOCUMENTS AND SETTINGS\...
100
64
144
Press any key to continue

```

Pra, kemi sistemet numerike me peshë:

$$N = a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \dots + a_1 b^1 + a_0 b^0$$

ku, a – shifra në pozitën përkatëse, b – baza

Dec:  $100 = 1 \cdot 10^2 + 0 \cdot 10^1 + 0 \cdot 10^0 = 1 \cdot 100 + 0 \cdot 10 + 0 \cdot 1 = 100 + 0 + 0 = 100$ ;

Hex:  $64 = 6 \cdot 16^1 + 4 \cdot 16^0 = 96 + 4 = 6 \cdot 16 + 4 \cdot 1 = 100$ ;

Oct:  $144 = 1 \cdot 8^2 + 4 \cdot 8^1 + 4 \cdot 8^0 = 1 \cdot 64 + 4 \cdot 8 + 4 \cdot 1 = 64 + 32 + 4 = 100$ ;

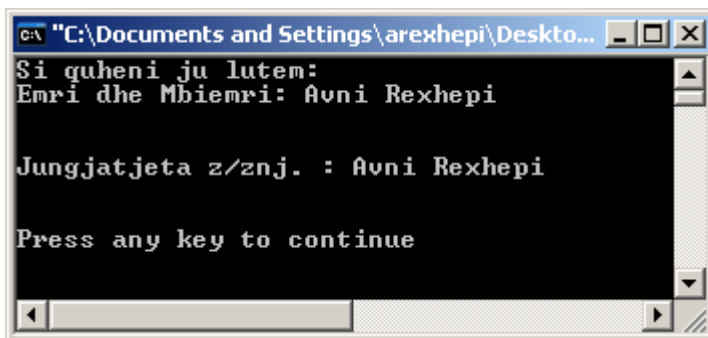
## Leximi i fjalëve dhe fjalive

```
// Programi Fjala1
#include <iostream>
using namespace std;
int main()
{
    char Emri[20], Mbiemri[20];

    cout<<"Si quheni ju lutem:\n";
    cout<<"Emri dhe Mbiemri: ";

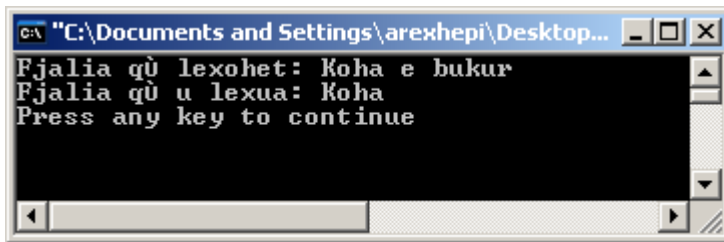
    cin >> Emri >> Mbiemri;

    cout << endl;
    cout << endl;
    cout << "Jungjatjeta z/znj. : "<<Emri<<" " <<Mbiemri<< "\n";
    cout << endl;
    cout << endl;
    return 0;
}
```



Tekstet ose fjalët deklarohen si variabla të tipit karakter me gjatësi të caktuar (varg/vektor i karaktereve). Duhet pasur kujdes, se gjatë leximit prej tastierës (përmes urdhërit „cin“) mirret vetëm teksti deri tek zbrazëtira e parë që haset gjatë leximit, kështu që lexohet vetëm fjala e parë e shtypur.

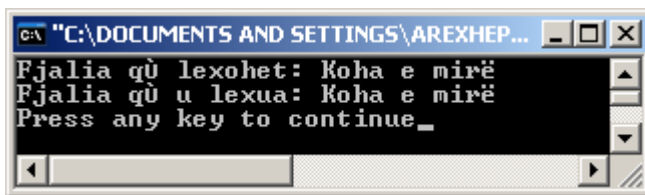
```
// Programi lexo-shtyp
#include <iostream>
using namespace std;
int main()
{
    char A[20];
    cout << "Fjala që lexohet: ";
    cin >> A;
    cout << "Fjala që u lexua: "
         << A
         << "\n";
    return 0;
}
```



Leximi i tekstit me gjatësi të caktuar, bëhet me urdhërin `cin.get` (ang. Get – marr, merr)

`cin.get(a,n)` // `n` – numri i karaktereve që lexohen (pranohen)

```
// Programi lexo6
#include <iostream>
using namespace std;
int main()
{
    const int m=20;
    char A[m];
    cout << "Fjalja që lexohet: ";
    cin.get(A,m);
    cout << "Fjalja që u lexua: "
        << A
        << "\n";
    return 0;
}
```



Leximi i tekstit mund të bëhet edhe duke lexuar komplet rreshtin:

`cin.getline` // leximi i rreshtit (ang. `getline` – merre rreshtin)

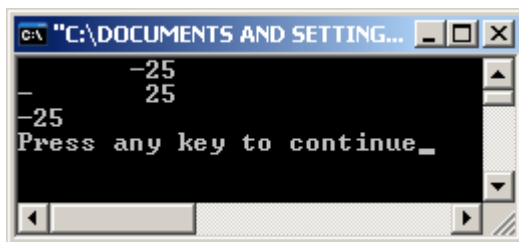
```
// Programi lexo7
#include <iostream>
using namespace std;
int main()
{
    const int m=20;
    char A[m];
    cout << "Fjalja që lexohet: ";
    cin.getline(A,m);
    cout << "Fjalja që u lexua: "
        << A
        << "\n";
    return 0;
}
```

## Rreshtimi i tekstit

Gjatë shtypjes, rezultati mund të rreshtohet djathtas, majtas apo në mes. Për të bërë rreshtimin e tekstit përdoren manipulorët: left (majtas), right (djathtas) dhe internal (për brenda).

```
// Rreshtimi
#include <iostream>
using namespace std;

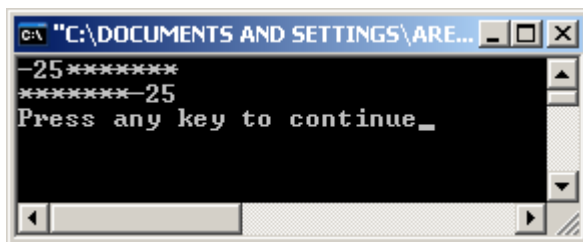
int main ()
{
    int n;
    n=-25;
    cout.width(10); cout << right << n << endl;
    cout.width(10); cout << internal << n << endl;
    cout.width(10); cout << left << n << endl;
    return 0;
}
```



Rreshtimi mund të bëhet edhe duke përdorur manipulorët: `setiosflags(ios::left)` dhe `setiosflags(ios::right)` të cilët ndodhen në header fajllin `<iomanip>`, prandaj duhet të përfshihen me direktivën `#include`.

```
// Rreshtimi
#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
    int n;
    n=-25;
    cout.width(10);
    cout.fill('*');
    cout << setiosflags(ios::left) << n << endl;
    cout.width(10);
    cout.fill('*');
    cout << setiosflags(ios::right) << n << endl;
    cout.width(15);
    return 0;
}
```



## Stringu

C++ ka dy metoda për paraqitjen e të dhënave tekstuale - stringjeve (variabla string), vargu i karaktereve në stil të C-së dhe klasa string. Vargu i karaktereve është nivel i ulët i përfaqësimit të të dhënave string, mirëpo akoma haset nëpër kode të ndryshme. Klasa string, e cila është pjesë e librarisë standarde të C++-it, ofron metoda për manipulim të lehtë me të dhënat tekstuale (string).

### Vargu i karaktereve

Siç tregon edhe vet emri, vargu i karaktereve në stil të C-së është paraqitje e të dhënave string që përdoret në gjuhën C. Në C, kjo është teknika e vetme për ruajtjen dhe manipulimin e të dhënave string. Stringjet ruhen si vargje të karaktereve të përfunduara me null - '\0'. Kjo paraqitje ka disa dobësi, të cilat duhet të evitohen në C++ duke përdorur klasën string.

- Vargu i karaktereve me madhësi të mjaftueshme duhet të definohet ose alokohet për të mbajtur së paku gjatësinë e stringut plus një, pasi që një bajt nevojitet për të ashtuquajturin “null terminator” (përfunduesin null (zero)). Është përgjegjësi e programerit që të jetë i sigurtë për gjatësinë e mjaftueshme të vargut. Kompajlimi nuk do të lajmëroj ndonjë gabim apo vërejtje nëse gjatësia është më e vogël se sa që duhet. Gabimet në madhësinë e vargut të karaktereve do të paraqiten gjatë ekzekutimit. Programi mund të bllokohet, të ketë sjellje të gabuar ose të paparashikuar.
- Ndojnëherë, është e nevojshme që “null terminatori” të shtohet në mënyrë eksplicite.
- Zakonisht nevojiten ose përdoren pointerët për të ju qasur dhe për të manipuluar me të dhënat string.
- Kur të kopjohen stringjet, programeri duhet të verifikojë vargun cak për gjatësi të nevojshme. Kur shtohen stringjet, përsëri gjatësia e stringut duhet të mirret në konsiderim.

Mirëpo, ka disa arsye të rëndësishme, për studimin dhe mësimin e stringjeve të stilit të C-së. Së pari, mirëmbajtja e ndonjë kodi të trashëguar, sepse disa programerë i përdorin akoma. Së dyti, klasa string nuk ishte pjesë e versioneve fillestare të C++-it. Programertë i përdorin të dy mënyrat. Në secilin rast, nevojitet njohja e C-stringjeve. Arsyeja e fundit është se akoma përdoren në C++ për të manipuluar me argumentet e linjës komanduese. Argumentet e linjës komanduese janë të dhëna ose urdhëra të cilët i përcillen programit kur ai fillon ekzekutimin. Këto argumente i përcillen programit të C++-it në stilin e C-stringjeve.

### Klasa String

Libraria standarde e C++-it ofron klasën string. Përdorimi i kësaj klase duhet të përfshihet në program me direktivën: `#include <string>`.

Për dallim prej C-stringjeve, përfaqësimi i brendshëm i të dhënave string është i “fshehur” përmes klasës string. Të dhënat janë një bashkësi, e qasur dhe e manipuluar përmes përdorimit të metodave të klasës string. Programeri nuk është i preokupuar si dhe ku ruhet stringu. Kjo është një shembull i “enkapsulimit”. Le të shohim krijimin dhe inicializimin e objekteve string përmes një shembulli:

```

#include <iostream>
#include <string>
using namespace std;

int main()
{
    string S1;
        // Konstruktori i zakonshem, pa inicializim
    string S2("Tung");
        // Inicializimi me C-Style string
    string S3("Tung nga Kosova");
        // Inicializimi me C-Style string
    string S4(S3);
        // Inicializimi me nje string objekt tjetër
    cout << "S2: " << S2 << endl;
    cout << "S3: " << S3 << endl;
    cout << "S4: " << S4 << endl;

    S1 = S2;
        // Ndarja e një stringu, stringut tjetër

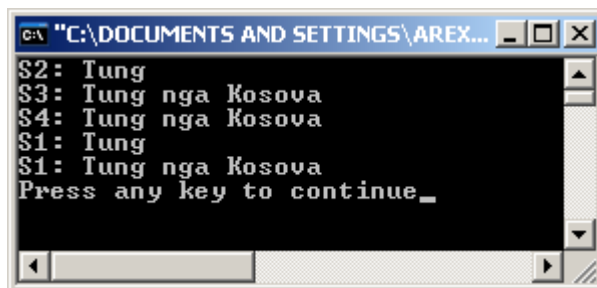
    cout << "S1: " << S1 << endl;

    S1 = S3;
        // S1 mori madhësinë e kërkuar.
        // Nuk nevojitet ndonjë urdhër i veçantë
    cout << "S1: " << S1 << endl;

    return 0;
}

```

Rezultati:



Pra, siç mund të shihet, ka disa konstruktorë për klasën string. Konstruktori standard krijon stringun e zbrazës. Format tjera të konstruktorit marrin karakteret e C-Stringut ose ndonjë tjetër objekt string si argument dhe inicializojnë stringun e ri me këtë vlerë. Gjithashtu, është e mundur që një stringu t'i "ndahet" vlera e stringut tjetër drejtpërdrejt. Me C-stringjet, një ndarje e tillë e vlerës do të duhej të bëhej përmes unazës ose ndonjë funksioni nga libraritë (si: strcpy).

### Veprimet me String

Klasa string përkrahë operatorët relacional si: <, <=, ==, !=, >= dhe > për krahasim të stringjeve. Operatori "+" përdoret për bashkimin e stringjeve. Operatori [] (subscript operator) mund të përdorte për të ju qasur ose për të caktuar ndonjë element individual të stringut. Klasa

string ka gjithashtu edhe metodat (funksionet për veprim), empty (zbrazë) për të testuar stringun e zbrazët dhe size (madhësia) për të llogaritur madhësinë (gjatësinë) e stringut.

**Shembull:**

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string S1;
    string S2(" nga ");
    string S3("Kosova");
    string S4;

    if (S1.empty())
    {
        cout << "S1 eshte i zbrazet" << endl;
        cout << "Ka gjatesine " << S1.size() << endl;
    }
    else
    {
        cout << "Jo, nuk eshte i zbrazet." << endl;
    }
    S1 = "Tung";
    cout << "S1 tani ka gjatesine " << S1.size() << endl;
    if (S3 < S1)    // Krahasoni relacionin
    {
        cout << "K eshte para T" << endl;
    }
    S4 = S1 + S2 + S3;    // bashkimi
    cout << "S4: " << S4 << endl;
    S1 += S2;    // bashkimi
    S1 += S3;

    cout << "S1: " << S1 << endl;
    //indeksimi (subscript)
    cout << "S1[0] " << S1[0] << endl;
    cout << "S1[1] " << S1[1] << endl;
    cout << "S1[2] " << S1[2] << endl;
    cout << "S1[3] " << S1[3] << endl;
    cout << "S1[4] " << S1[4] << endl;
    cout << "S1[5] " << S1[5] << endl;
    cout << "S1[6] " << S1[6] << endl;
    cout << "S1[7] " << S1[7] << endl;
    cout << "S1[8] " << S1[8] << endl;
    cout << "S1[9] " << S1[9] << endl;
    cout << "S1[10] " << S1[10] << endl;
    cout << "S1[11] " << S1[11] << endl;
    cout << "S1[12] " << S1[12] << endl;
    cout << "S1[13] " << S1[13] << endl;
    cout << "S1[14] " << S1[14] << endl;
    cout << "S1[15] " << S1[15] << endl;
    return 0;
}
```



Rezultati:

```

C:\> "C:\DOCUMENTS AND SETTINGS\A..."
S1 eshte i zbrazet
Ka gjatesine 0
S1 tani ka gjatesine 4
K eshte para T
S4: Tung nga Kosova
S1: Tung nga Kosova
S1[0] T
S1[1] u
S1[2] n
S1[3] g
S1[4]
S1[5] n
S1[6] g
S1[7] a
S1[8]
S1[9] K
S1[10] o
S1[11] s
S1[12] o
S1[13] v
S1[14] a
S1[15]
Press any key to continue

```

### Metodat e klasës string

Klasa string ka shumë metoda për manipulim të stringjeve, si: shtimi, insertimi, fshirja, kërkimi dhe zëvendësimi. Disa nga to janë përshkruar në tabelën vijuese:

Metoda	Përdorimi
append(char *pt); append(char *pt, size_t count); append(string &str, size_t offset, size_t count); append(string &str); append(size_t count, char ch); append(InputIterator Start, InputIterator End);	Shton karakteret prej C-stringut, char ose objekteve tjera string.
at(size_t offset);	Kthen referencën në karakterin në pozitën e specifikuar. Dallon prej operatorit të indeksimit [ ], pasi që verifikohen kufinjat.
begin();	Kthen një iterator në fillim të stringut.
*c_str();	Kthen pointerin në versionin e C-stringut të përmbajtjes së stringut.
clear();	Fshinë tërë stringun
copy(char *cstring, size_t count, size_t offset);	Kopjin "numrin-count" e karaktereve nga C-stringu duke filluar nga zhvendosja-offset.
empty();	Teston a është stringu i zbrazët.
end();	Kthen iteratorin në pozitën pas fundit të stringut (një pozitë më pas).
erase(iterator first, iterator last); erase(iterator it); erase(size_t pos, size_t count);	Fshinë karakteret nga pozita e specifikuar.
find(char ch, size_t offset = 0); find(char *pt, size_t offset = 0); find(string &str, size_t offset = 0);	Kthen indeksin e karakterit të parë të substringut kur ai gjendet. Përndryshe, kthehet vlera speciale "npos".

find_first_not_of();	Seti i njëjtë i argumenteve sikur "find". Gjenë indeksin e karakterit të parë që nuk është në stringun kërkues.
find_first_of();	Seti i njëjtë i argumenteve sikur "find". Gjenë indeksin e karakterit të parë që është në stringun kërkues.
find_last_not_of();	Seti i njëjtë i argumenteve sikur "find". Gjenë indeksin e karakterit të fundit që nuk është në stringun kërkues.
find_last_of();	Seti i njëjtë i argumenteve sikur "find". Gjenë indeksin e karakterit të fundit që është në stringun kërkues.
insert(size_t pos, char *ptr); insert(size_t pos, string &str); insert(size_t pos, size_t count, char ch); insert(iterator it, InputIterator start, InputIterator end);	Inserton karakteret në pozitën e specifikuar.
push_back(char ch);	Inserton karakteret në fund të stringut.
replace(size_t pos, size_t count, char *pt); replace(size_t pos, size_t count, string &str); replace(iterator first, iterator last, char *pt); replace(iterator first, iterator last, string &str);	Zëvendëson elementet në string me karakteret e specifikuar. Rangu mund të specifikohet përmes pozitës startuese dhe numrit të elementeve për zëvendësim ose duke përdorur iteratorët.
size();	Kthen numrin e elementeve në string.
swap(string &str);	Shkëmben dy stringje.

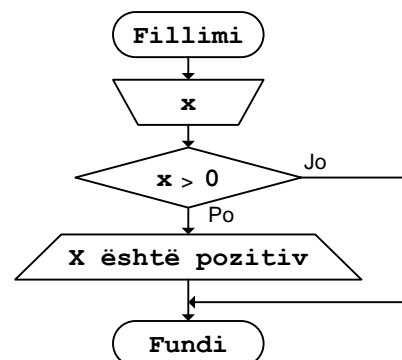
## Kushtet (Degëzimet)

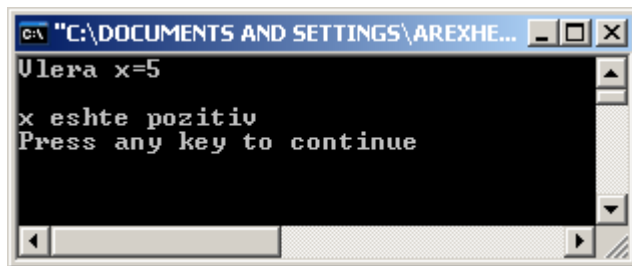
Gjatë shkruarjes së programeve për raste të ndryshme, paraqitet nevoja e kushtëzimit të ekzekutimit të urdhërave në bazë të ndonjë kushti të caktuar. Në C++ kemi disa forma të paraqitjes së kushteve: Rasti më i thjeshtë është kur për ndonjë kusht të caktuar ekzekutohet urdhëri (ose urdhërat), kurse nëse nuk plotësohet kushti, nuk ndodhë asgjë por programi vazhdon me urdhërin e ardhshëm (vijues), si në vazhdim:

```
if (kushti) urdhëri ;
ku:
    kushti - kushti për degëzim
    urdhëri - urdhëri që ekzekutohet nëse plotësohet kushti k
Zakonisht, në program kushti shkruhet në rresht të veçantë:
```

```
if (kushti)
    urdhëri ;
```

```
// Programi ifla
#include <iostream>
using namespace std;
int main()
{
    int x;
    cout << "Vlera x=";
    cin >> x;
    if (x > 0)
        cout << "\nx eshte pozitiv \n";
    return 0;
}
```





Gjatë shkruarjes së programeve, përdoren operatorët aritmetik dhe operatorët relacional:

Operatori në C++	Operacioni	Shembull	Rezultati
+	Mbledhja	3+2	5
-	Zbritja	3-2	1
*	Shumëzimi	3*2	6
/	Pjestimi	3/2	1.5
%	Moduli	3%2	1

\*Moduli, është mbetja nga pjesëtimi i plotë (5%2=1; 8%3=2).

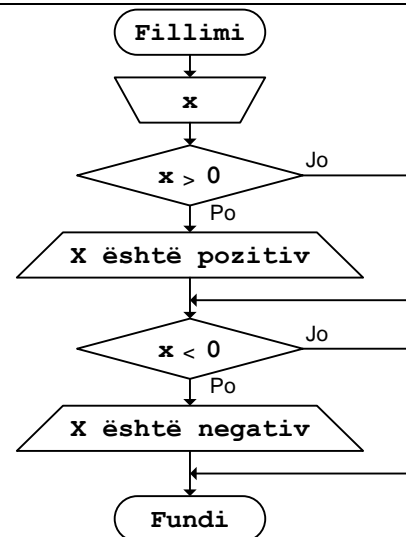
Operatori në C++	Domethënia	Shembull	Rezultati
<	Më i vogël se	2 < 3	true
<=	Më i vogël se, ose barazi me	3 <= 2	false
==	Barazi me	3 == 3	true
>	Më i madhë se	3 > 2	true
>=	Më i madhë se, ose barazi me	2 >= 3	false
!=	Jobarazi me	2 != 3	true

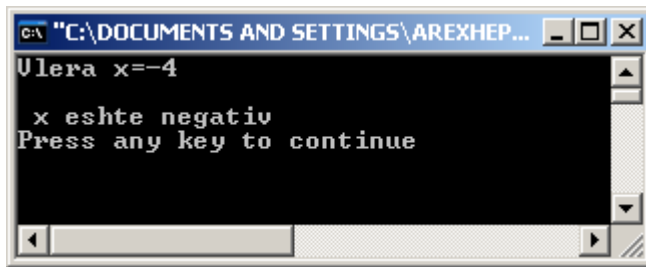
```
// Programi if1b
#include <iostream>
using namespace std;
int main()
{
    int x;

    cout << "Vlera x=";
    cin >> x;

    if (x > 0)
        cout << "\n x eshte pozitiv \n";
    if (x < 0)
        cout << "\n x eshte negativ \n";

    return 0;
}
```

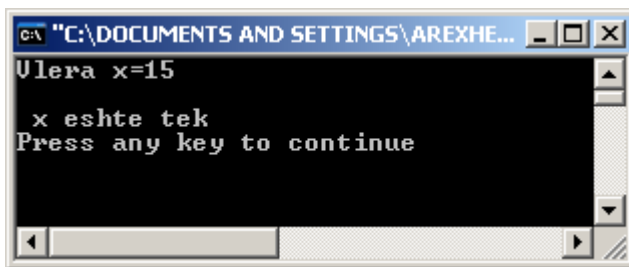




```
// Programi if2
#include <iostream>
using namespace std;
int main()
{
    int x;
    cout << "Vlera x=";
    cin >> x;

    if ((x%2) == 0)
        cout << "\n x eshte cift\n";

    if ((x%2) == 1)
        cout << "\n x eshte tek\n";
    return 0;
}
```



Kemi edhe rastin kur për ndonjë kusht të caktuar ekzekutohet urdhëri (ose urdhërat), kurse nëse nuk plotësohet kushti duhet të ekzekutohet ndonjë urdhër tjetër, si në vazhdim:

```
If (kushti)
    urdhëri1;
else
    urdhëri2;
```

**kushti** - kushti për degëzim.

**urdhëri1** - urdhëri që ekzekutohet nëse plotësohet kushti k.

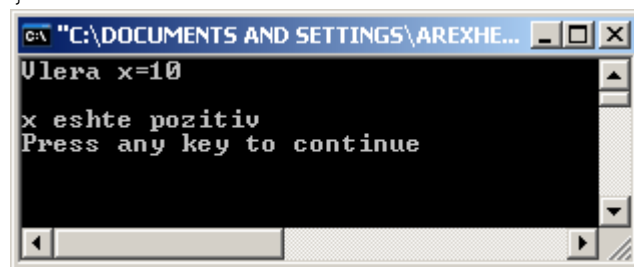
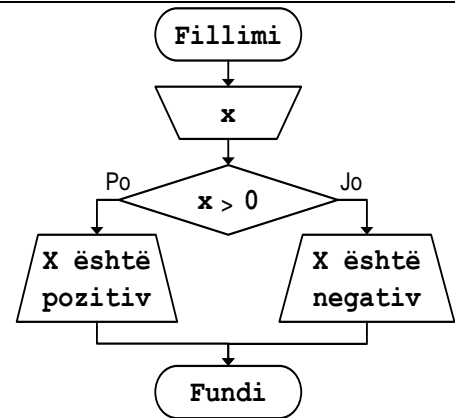
**urdhëri2** – urdhëri që ekzekutohet nëse nuk plotësohet kushti k.

```
// Programi if2a
#include <iostream>
using namespace std;
int main()
{
    int x;

    cout << "Vlera x=";
    cin >> x;

    if (x > 0)
        cout << "\nx eshte pozitiv \n";
    else
        cout << "\nx eshte negativ\n";

    return 0;
}
```



## Blloqet e urdhërave në degë

Nëse për një kusht duhet të ekzekutohen disa urdhëra (më shumë se një urdhër), atëherë urdhërat e tillë duhet të futen brenda kllapave të mëdha, duke krijuar një bllok të urdhërave.

```
if (kushti)
{
    urdhërat1;
}
else
{
    urdhërat2;
}
```

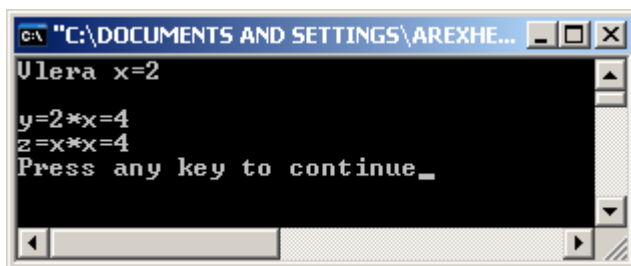
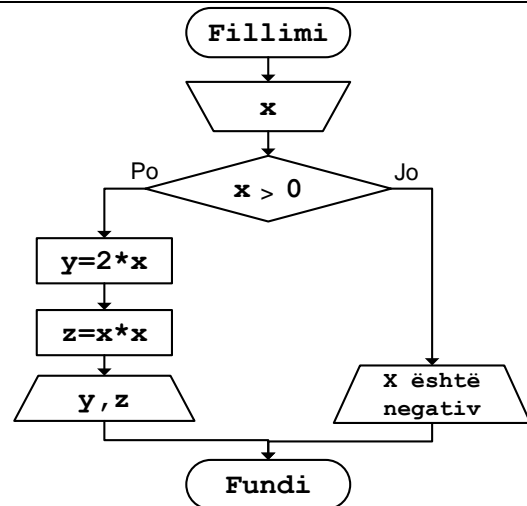
```
// Programi if3
#include <iostream>
using namespace std;
int main()
{
    int x,y,z;

    cout << "Vlera x=";
    cin >> x;

    if (x > 0)
    {
        y=2*x;
        z=x*x;

        cout << "\ny=2*x="<<y;
        cout << "\nz=x*x="<<z;
    }
    else
        cout << "\nx eshte negativ\n";

    cout<<endl;
    return 0;
}
```



## Kushtet e shumëfishta dhe kushtet ndërthurura

Kushtet e shumëfishta krijohen duke i kombinuar disa kushte përmes operatorëve logjik: AND (Dhe), OR (Ose) dhe negacionit NOT (Jo). Për AND duhet të plotësohen të dy kushtet. Për OR mjafton të plotësohet njëri prej kushteve.

Operatori	Operacioni	Shembull	Llogaritja	Rezultati
&&	Konjunksioni, AND	(2<3) && (4==4)	true && true	true
	Disjunksioni, OR	(2!=3)    (3>4)	true    false	true
!	Negacioni, NOT	!(5>4)	!true	false

Kushtet e ndërthurura paraqiten në rastet kur kemi kusht brenda kushtit (IF brenda IF-it), pra kur ndonjë urdhër duhet të ekzekutohet vetëm nëse janë plotësuar disa kushte të njëpasnjëshme.

```

if (kushti1)
    if (kushti2)
        urdhëri1;
    else
        urdhëri2;
else
    urdhëri3;

```

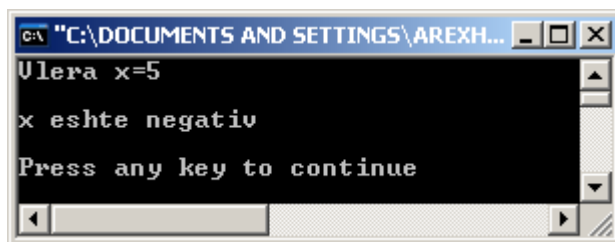
Pra, **urdhëri1** ekzekutohet vetëm kur plotësohen edhe **kushti1** edhe **kushti2**. **urdhëri2** ekzekutohet kur plotësohet **kushti1** por nuk plotësohet **kushti2**. **urdhëri3** ekzekutohet kur nuk plotësohet fare **kushti1** (në këtë rast, kushti2 nuk vjen në pyetje fare).

P.sh, në rastin vijues, edhe pse duket se llogaritja do të bëhet në rregull duke përdorur kombinim e kushteve përmes operatorit DHE, mund të shihet se për numrat pozitiv por më të vegjël se 10, rezultati del i gabuar.

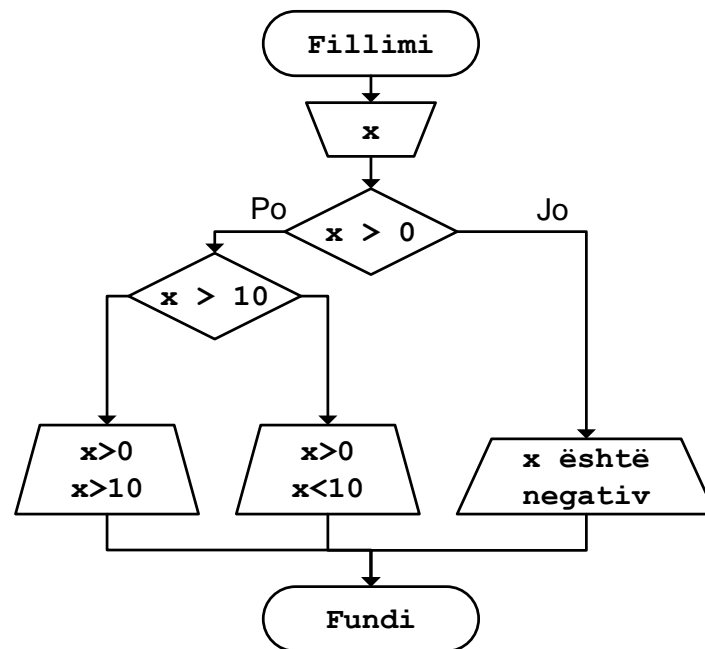
```

// Programi if4-llogaritja e gabuar
#include <iostream>
using namespace std;
int main()
{
    int x,y,z;
    cout << "Vlera x=";
    cin >> x;
    if ((x > 0) && (x>10))
    {
        cout << "\nx eshte pozitiv me i madh se 10\n";
    }
    else
        cout << "\nx eshte negativ\n";
    cout<<endl;
    return 0;
}

```



Duke përdorur kushtin e ndërthurur (IF brenda IF-it), mund të llogaritet në mënyrë korrekte.



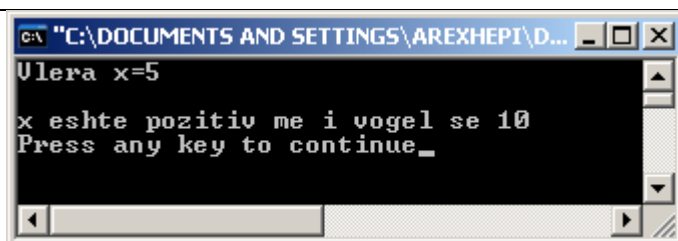
```

// Programi if4b - permiresimi i gabimit nga detyra paraprake
#include <iostream>
using namespace std;
int main()
{
    int x;

    cout << "Vlera x=";
    cin >> x;

    if (x > 0)
        if (x > 10)
            cout << "\nx eshte pozitiv me i madh se 10\n";
        else
            cout << "\nx eshte pozitiv me i vogel se 10\n";
    else
        cout << "\nx eshte negativ \n";
    return 0;
}

```



## Operatori “?”

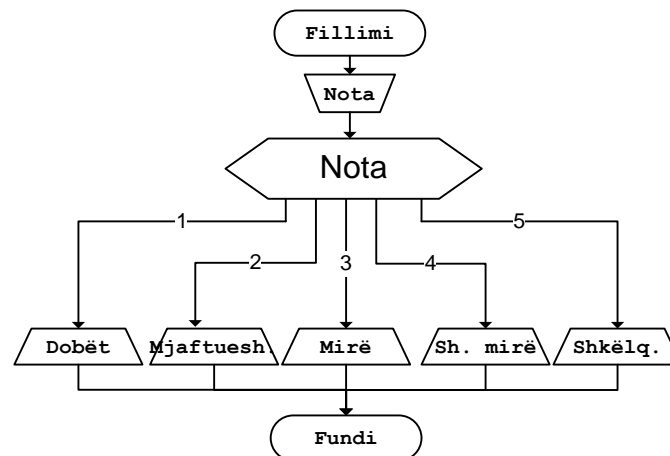
Operatori i kushtëzimit “?” funksionon si një “IF” me “else”, ashtu që nëse plotësohet kushti ekzekutohet “urdhëri1”, përndryshe ekzekutohet “urdhëri2”:

```
(Kushti) ? urdhëri1 : urdhëri2;
```



```
// Programi - operatori i kushtezimit
#include <iostream>
using namespace std;
int main()
{
    int a=2, b=8;
    int y;
    y=(a>b) ? a : b;
    cout << "Vlera e variablës y="
        << y
        << "\n";
    return 0;
}
```

## Switch ( )



Për degëzimet e shumëfishta, me “if” programi do të duhej të shkruhet si vijon:

```
// Programi case1
#include <iostream>
using namespace std;
int main()
{
    int Nota;

    cout<<"Jepni noten 1, 2, 3, 4 ose 5 :  Nota=" ;
    cin >> Nota;

    if (Nota == 1) cout<<"Dobët: ";
    else
    if (Nota == 2) cout<<"Mjaftueshem: ";
    else
    if (Nota == 3) cout<<"Mire: ";
    else
    if (Nota == 4) cout<<"Shume mire: ";
    else
    if (Nota == 5) cout<<"Shkelqyeshem: ";
    else cout << "Gabim - Nota duhet mes 1 dhe 5";
}
```

```

cout<<" Nota="
    << Nota
    << "\n";

    return 0;
}

```

Në vend të kësaj, përmes urdhërit/operatorit **Switch**, bëhet shpërndarja më e qartë, me mundësi më të lehta të përcjelljes, të shtimit të opcioneve të reja, etj. (angl. Switch – ndryshim, transferim, ndërprerës për kyçje/çkyçje, etj). Variabla që kontrollohet në Switch, duhet të jetë e tipit **Integer** ose **Char**.

```

// Programi case2
#include <iostream>
using namespace std;
int main()
{
    int Nota;

    cout<<"Jepni noten 1, 2, 3, 4 ose 5 :  Nota=" ;
    cin >> Nota;
    switch(Nota)
    {
        case 1: cout<<"Dobët: ";
                break;
        case 2: cout<<"Mjaftueshëm: ";
                break;
        case 3: cout<<"Mire: ";
                break;
        case 4: cout<<"Shume mire: ";
                break;
        case 5: cout<<"Shkelqyeshëm: ";
                break;
        default: cout <<"Gabim-Nota duhet te jete mes 1 dhe 5\n";
    }

    cout    <<" Nota="
           << Nota
           << "\n";

    return 0;
}

```

Për çdo “case” (angl. Case – rast, shembull, mundësi) duhet të vendoset edhe **break** (ndale, ndërpreje), për të ndërprerë ekzekutimin e mëtejme të kushteve kur të gjendet vlera e rastit të caktuar. Në fund, me “**default**” eliminohet mundësia e dhënies së vlerave të gabuara (që nuk janë përfshirë në asnjërin rast të paraparë (ose parashikuar)).

P.sh, pjesa në vijim mund të paraqitet në fillim të ndonjë programi më të gjatë:

```

// Programi case3
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{

```

```

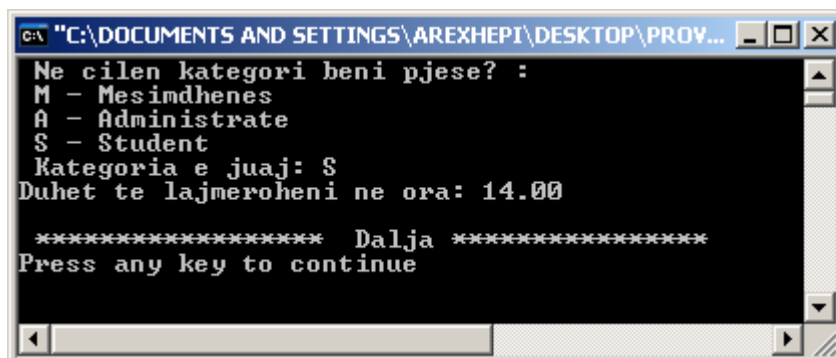
int i;
char zgjedhja;
cout<<" Ne cilen kategori beni pjese? :\n" ;
cout<<" M - Mesimdhenes\n" ;
cout<<" A - Administrative\n" ;
cout<<" S - Student\n" ;
cout<<" Kategoria e juaj: " ;
cin >> zgjedhja;

//if (zgjedhja>=97)
//zgjedhja=zgjedhja-32;
zgjedhja=toupper(zgjedhja);

switch(zgjedhja)
{
case ('M'): cout<<"Duhet te lajmeroheni ne ora: 10.00\n";
            break;
case ('A'): cout<<"Duhet te lajmeroheni ne ora: 12.00\n";
            break;
case ('S'): cout<<"Duhet te lajmeroheni ne ora: 14.00\n";
            break;
default: cout << "Gabim\n";
}
cout << "\n ***** Dalja *****\n";

return 0;
}

```



Për t'a kthyer shkronjën e dhënë në shkronjë të madhe, mund të përdoret mënyra përmes shndërrimit sipas vlerës së kodit ASCII, sepse shkronjat e mëdha e kanë kodin prej A – 65 gjere në Z – 90, kurse ato të vogla, për 32 më shumë: prej a – 97 gjere ne z – 122

```

//if (zgjedhja>=97)
//zgjedhja=zgjedhja-32;

```

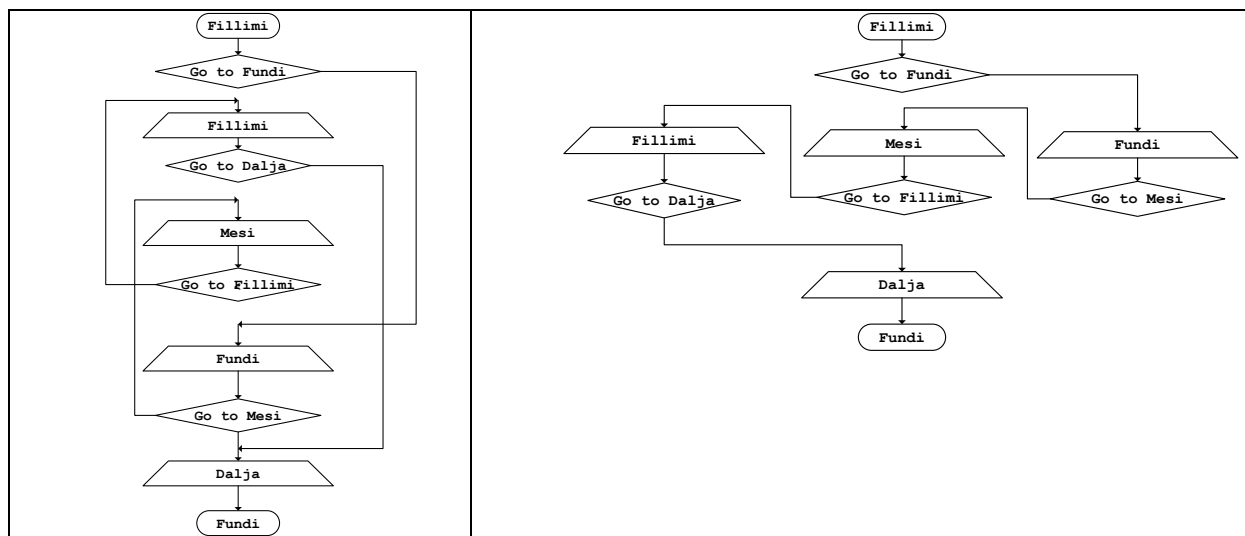
Shndërrimi në shkronjë të madhe mund të bëhet edhe përmes funksionit “toupper” (angl. to upper - në të madhe) i cili ndodhet në heder fajllin (librarinë) <cstdlib>. Ekziston edhe funksioni “tolower” (angl. to lower – në të vogël).

```

zgjedhja=toupper(zgjedhja); //shndërrimi në shkronjë të madhe

```

## Kapërcimi pa kusht: goto



**Go to** (angl. kalo tek, shko tek) bën kapërcimin e detyrueshëm në pjesën e caktuar të programit, e cila është e adresuar përmes Label-ave (labelave, etiketave, adresave) të cilat janë emra të çfarëdoshëm, që përfundojnë me dy pika (:).

Para “unazave” (të cilat do të shpjegohen në vazhdim të tekstit), për t’u kthyer në pjesën e urdhërave në program të cilët veq janë ekzekutuar më herët dhe për t’i ekzekutuar ata përsëri, mënyra e vetme ka qenë kjo përmes urdhërit “goto”.

```

// Programi goto1
#include <iostream>
using namespace std;
int main()
{
    goto Fundi;

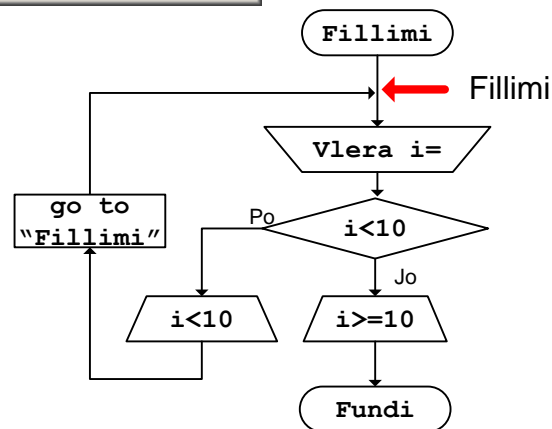
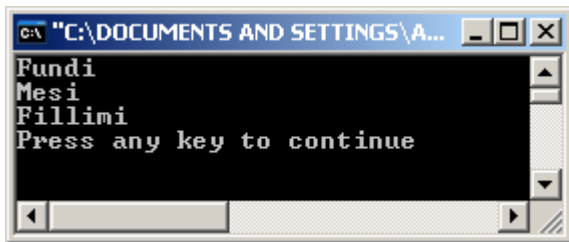
Fillimi:                                //Labela - etiketa,adresa "Fillimi"
    cout<< "Fillimi\n";
    goto Dalja;

Mesi:                                  //Labela - etiketa,adresa "Mesii"
    cout << "Mesi\n";
    goto Fillimi;

Fundi:
    cout << "Fundi\n";
    goto Mesi;

Dalja:
    return 0;
}

```



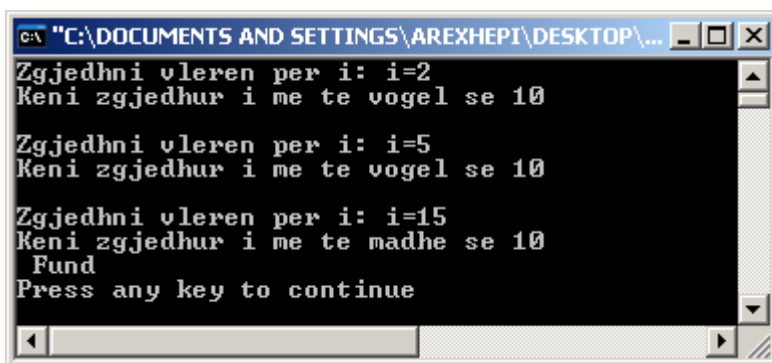
```

// Programi goto2
#include <iostream>
using namespace std;
int main()
{
    int i;
Fillimi: //Adresa Fillimi
    cout<< "Zgjedhni vleren per i: i=";
    cin>>i;

    if (i<10)
    {
        cout << "Keni zgjedhur i me te vogel se 10\n\n";
        goto Fillimi;
    }

    cout << "Keni zgjedhur i me te madhe se 10\n";
    cout <<" Fund \n";
    return 0;
}

```



Për t'i mundësuar shfrytëzuesit që pas ekzekutimit të programit, të zgjedhë nëse dëshiron të bëjë përsëri riekzekutimin e programit prej fillimit pa e ndërprerë atë (pa dalë prej tij), mund të përdoret forma si vijon, përmes urdhërit goto.

```
// Programi goto3
#include <iostream>
using namespace std;
int main()
{
    int i;
    char zgjedhja;

    Perseritje:    //Labela(etiketa-adresa) per kapercim pa kusht

    cout << "Zgjedhni vleren per i: i=";
    cin >> i;
    cout << endl;

    if (i<10)
        cout << "Keni zgjedhur i me te vogel se 10\n\n";
    else
        cout << "Keni zgjedhur i me te madh ose baraz me 10\n\n";

    cout << "Perserite testin? (P-Po, J-Jo) \a";
    cin >> zgjedhja;

    if ((zgjedhja=='P') || (zgjedhja == 'p'))
        goto Perseritje;

    cout << "\n *****          Dalja *****\n";
    return 0;
}
```

```
C:\DOCUMENTS AND SETTINGS\AREXHEPT\DESKTOP\PROVA C++\...
Zgjedhni vleren per i: i=5
Keni zgjedhur i me te vogel se 10
Perserite testin? (P-Po, J-Jo) P
Zgjedhni vleren per i: i=15
Keni zgjedhur i me te madh ose baraz me 10
Perserite testin? (P-Po, J-Jo) j
*****          Dalja *****
Press any key to continue
```

## Unazat (Loops - laqet)

**Loop** (angl Loop-laku, rrethi i mbyllur, apo ndryshe si term më i përshtatshëm në gjuhën shqipe, unaza) paraqet pjesën e programit, ku bëhet përsëritja e ekzekutimit të një apo më shumë urdhërave të programit. Unaza ka variablën e unazës, e cila përdoret si tregues se sa herë do të përsëriten urdhërat brenda unazës. Ajo fillon me vlerën fillestare dhe për çdo herë të ekzekutimit të urdhërave të unazës, e ndryshon vlerën për një hap të caktuar, që njihet si hapi i unazës. Në fund të pjesës së unazës, testohet se a është plotësuar kushti për dalje prej unazës. Kjo rritje e variablës së unazës, për hapin e caktuar, vazhdon deri sa të plotësohet kushti për dalje prej unazës. Në C++ përkrahen disa forma të unazave:

### Unaza **while** (gjersa)

Unaza **While** (gjersa), bën përsëritjen e urdhërave të unazës, gjersa vlenë kushti i caktuar. Pra, gjersa për kushtin e unazës **While** kemi rezultatin TRUE (E SAKTË, PO), ekzekutohen urdhërat e unazës dhe variabla e unazës rritet, për hapin e caktuar.

Unaza **While**, së pari e vlerëson kushtin dhe pastaj, nëse plotësohet kushti, ekzekuton urdhërat në unazë. Nëse kushti nuk plotësohet herën e parë që testohet kushti (për hapin e parë të variablës së unazës), pra për vlerën fillestare të variablës së unazës, atëherë urdhërat në unazë nuk do të ekzekutohen asnjëherë.

```
i=f;
while (kushti)
{
    urdhëri/at;
    i=i+h;
}
```

ku janë:

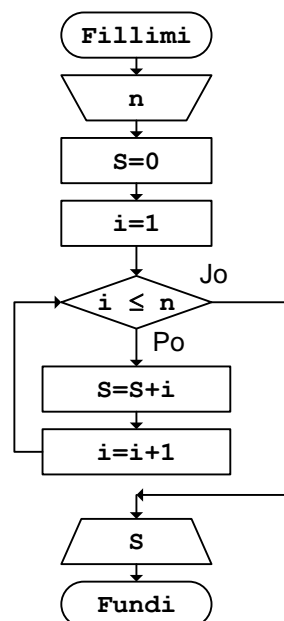
i - variabla e unazës.

f - vlera fillestare e variablës së unazës.

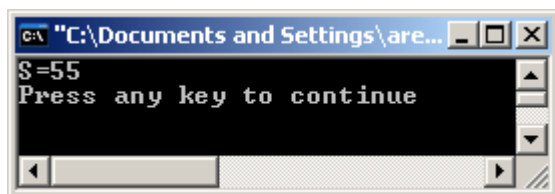
kushti - kushti i ekzekutimit të urdhërave që përfshihen në trupin e unazës.

h - hapi me të cilin ndryshohen vlerat e variablës i.

urdhëri/at – urdhërat që ekzekutohen brenda unazës.



```
// Program while1
#include <iostream>
using namespace std;
int main()
{
    double S;
    int i;
    S=0;
    i=1;
    while (i<=10)
    {
        S=S+i;
        i=i+1;
    }
    cout << "S="
         << S
         << "\n";
    return 0;
}
```



Nëse Unaza **While** për kushtin e unazës ka gjithmonë rezultatin TRUE, atëherë urdhërat e unazës do të përsëriten pafundësisht. P.sh, nwse vendoset kushti si nw vijim:

```
while (true)
{
    Urdhërat...
}
```

atëherë do të kemi unazë të pafundme, që përsëritet vazhdimisht. Për të mos ndodhur kjo, duhet që me ndonjë urdhër përbrenda unazës, të vendosim ndonjë kusht plotësues për ndërprerje të unazës.

## Unaza do – while

Unaza Do...**While** (Bëj...gjersa), bën përsëritjen e urdhërave të unazës, gjersa vlenë kushti i caktuar. Pra, gjersa për kushtin e unazës **While** kemi rezultatin TRUE (E SAKTË, PO), ekzekutohen urdhërat e unazës dhe variabla e unazës rritet, për hapin e caktuar.

Mirëpo, për dallim nga unaza **While**, unaza **Do...While**, së pari e i ekzekuton urdhërat e unazës, e pastaj e vlerëson kushtin për dalje prej unazës (apo për mbejtte në unazë) dhe pastaj, nëse plotësohet kushti, përsëri ekzekuton urdhërat në unazë. Edhe nëse kushti nuk plotësohet herën e parë që testohet (për hapin e parë të variablës së unazës), pra për vlerën fillestare të variablës së unazës, urdhërat në unazë veç janë ekzekutuar njëherë. Pra, së paku një herë ata do të ekzekutohen gjithsesi.



```

i=f;
do
{
    urdhëri/at;
    i=i+h;
}
while(kushti);

```

ku janë:

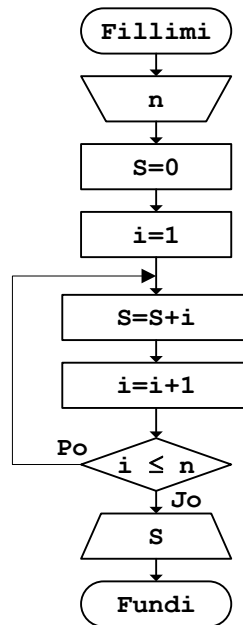
i - variabla e unazës.

f - vlera fillestare e variablës së unazës.

kushti - kushti i ekzekutimit të urdhërave që përfshihen në trupin e unazës.

h - hapi me të cilin ndryshohen vlerat e variablës i.

urdhëri/at – urdhëri/at e përfshira brenda unazës.



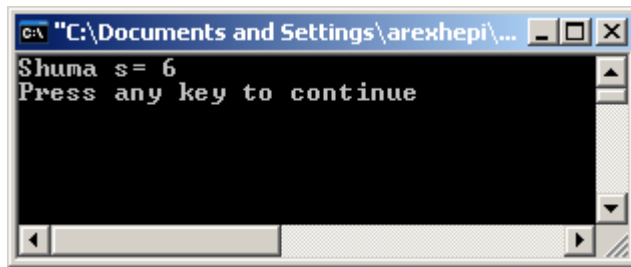
```

// Programi do-while1
#include <iostream>
using namespace std;
int main()
{
    const int m=3;
    int i;
    double s;

    s=0;
    i=1;
    do
    {
        s=s+i;
        i++;
    }
    while (i<=m);
    cout << "Shuma s= "
         << s
         << "\n";
    return 0;
}

```

//vlera fillestare  
 //(do - bëj, vepro, kryej)  
  
 //urdhëri  
 //rritja e haptit të unazës  
  
 //kushti për mbetje në unazë



## Unaza for

Unaza for (për), gjithashtu bën përsëritjen e urdhërave brenda unazës, duke përshkruar në fillim vlerën fillestare, kushtin dhe hapin e unazës. Ka formë më kompakte dhe përdoret në të shumtën e rasteve. Zakonisht, kur vlerat kufitare (vlera fillestare dhe vlera përfundimtare) të variablës së unazës janë të njohura paraprakisht, përdoret unaza for.

```
for (i=f; kushti; i=i+h)
{
    urdhëri/at;
}
```

ku:

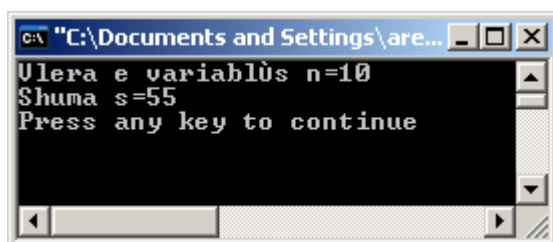
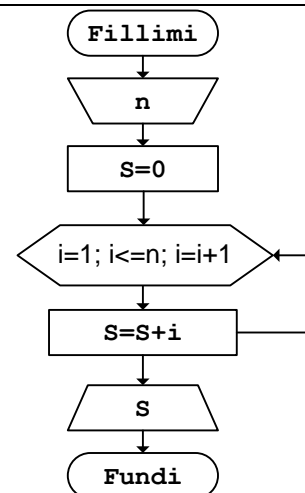
i - variabla e unazës.

kushti - kushti i mbetjes brenda unazës.

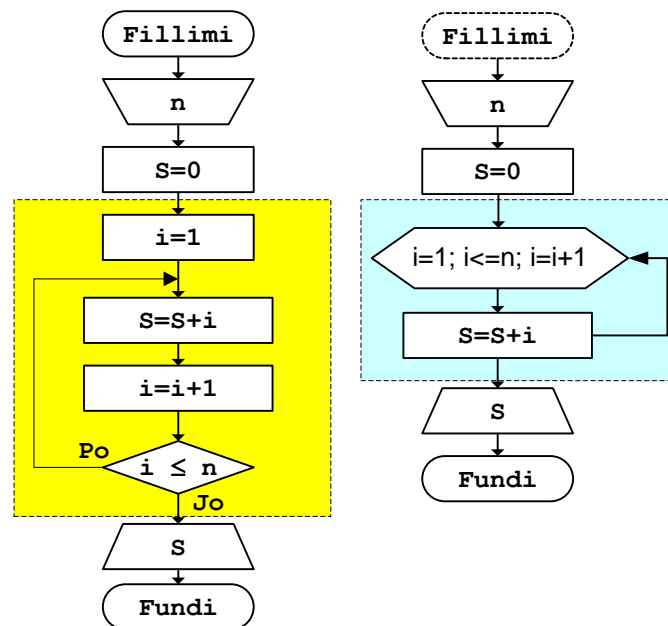
h - hapi me të cilin ndryshohen vlerat e variablës i.

urdhëri/at – urdhëri/at që ekzekutohen brenda unazës.

```
// Programi for1
#include <iostream>
using namespace std;
int main()
{
    int i,n;
    double s=0;
    cout << "Vlera e variablës n=";
    cin >> n;
    for (i=1;i<=n;i++)
        s=s+i;
    cout << "Shuma s="
        << s
        << "\n";
    return 0;
}
```



Nga skema në vijim mund të shihet se si pasqyrohet pjesa e unazës “do while” në atë “for”

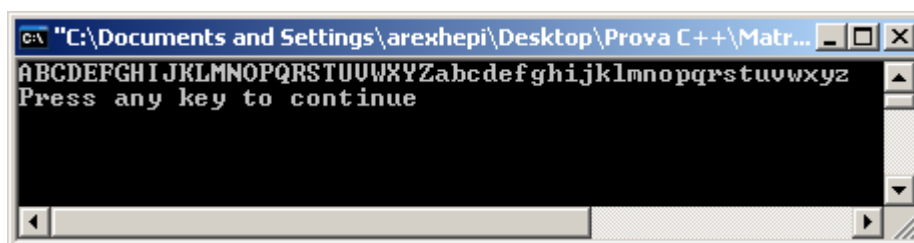


```

// Programi forAscii - shtypja e shkronjave te kodit ASCII
#include <iostream>
using namespace std;
int main()
{
    int i;

    for (i=65;i<=90;i++)
        cout << char(i);
    for (i=97;i<=122;i++)
        cout << char(i);
    cout << endl;
    return 0;
}

```



Per te shtypur te gjitha karakteret e kodit ASCII, behet unaza:  
 for(i=1; i<=255, i++)

Unaza for mund të përdorë edhe variabl të unazës që zvogëlohet për secilin hap të unazës:

```

#include <iostream>
using namespace std;
int main()
{
    int i;
    cout << "Numrimi ne zbritje" << endl;
    i=10;
}

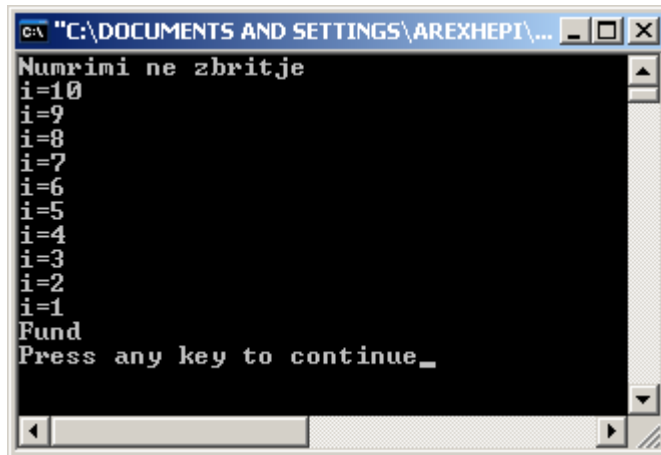
```

```

    for(i=10;i>0;i--)
    {
        cout<<"i="<<i<<"\n";
    }

cout << "Fund" << endl;
return 0;
}

```



### Format e unazës for

Unaza for mund të paraqitet edhe në trajtat e saj të zgjeruara, me më shumë varibala të fushave ose në trajta të shkurtëra, kur nuk i jepen të tri komponentet që përcaktojnë fillimin, fundin dhe hapin me të cilin rritet/zvogëlohet variabla e unazës.

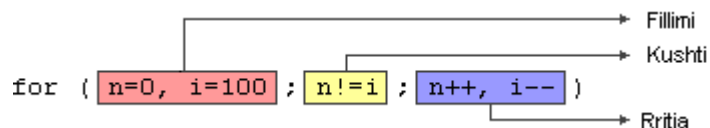
Duke përdorur operatorin presje “,” (operatori presje është një element ndarës i shprehjeve dhe shërben për ndarjen e dy e më shumë shprehjeve në rastet kur zakonisht pritët vetëm një shprehje), mund të përcaktohet më shumë se një shprehje në cilëndo fushë të unazës **for**, si në shembullin vijues, ku supozojmë se dëshirojmë të inicializojmë më shumë se një variabël të unazës:

```

for ( n=0, i=100 ; n!=i ; n++, i-- )
{
    // urdhërat tjerë...
}

```

Kjo unazë do të ekzekutohet për 50 herë (nëse as variabla “n” e as ajo “i” nuk modifikohen me ndonjë urdhër tjetër përbrenda unazës):



Vriabla “n” fillon me vlerën 0 kurse “i” me atë 100 dhe kushti i unazës është “n!=i” (n jobaraz me i). Pasi që “n” rritet për një kurse “i” zbritet për një, atëherë kushti i unazës do të bëhet “false” (jo i saktë) pas hapit të 50, kur n dhe i do të jenë të barabarta me 50.

Unaza for mund të përdoret edhe sa për të insertuar ndonjë vonesë kohore në një pjesë të programit, duke dhënë vetëm kreun e unazës por pa urdhëra përbrenda:

```
for ( n=0; n<1000000 ; n++);
```

Unaza for mund të paraqitet edhe në trajtën e shkurtër edhe pa ndonjërin ose në rastin ekstrem edhe pa asnjërin prej fushave të saj, si në vijim:

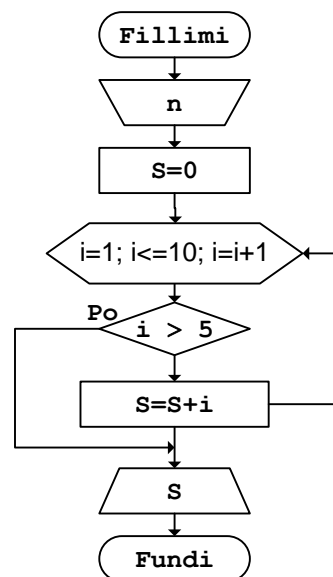
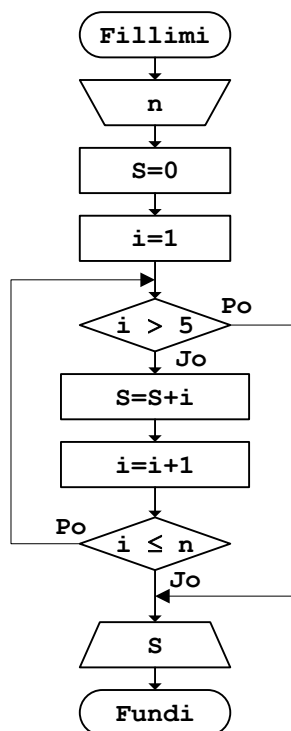
```
for(;;)
{
    cout << "Si quheni : ";
    cin >> Emri;
}
```

me ç'rast do të krijohej **unazë e pafundme**, si në rastin e unazës while me kushtin "true".

Për të mundësuar ndërprerjen e unazave të pafundme (ose edhe unazave të zakonshme për ndonjë kusht të caktuar) përdoret urdhëri "break" (angl break - ndërpreje, ndale).

## Ndërprerja e unazës (break)

Përmes urdhërit **break** (ndërpreje, ndale) mund të bëhet dalja e parakohshme prej unazës, gjegjësisht ndërprerja e saj. Kjo mund të bëhet, përmes testimit në ndonjë kushti shtesë, brenda trupit të unazës. P.sh., kemi unazën për shtypjen e numrave rendor, prej 1 deri në 10. Mirëpor, brenda unazës, vendosim kushtin, që nëse vlera e variablës së unazës është më e madhe se 5, të ndërpritet unaza.

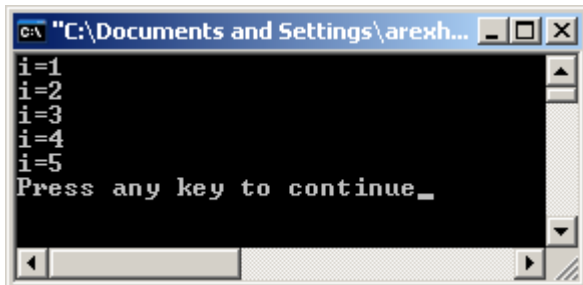


```
// Programi Break - ndërprerja e unazës, dalja prej unaze
#include <iostream>
using namespace std;
int main()
{
    int i;
    for (i=1; i<=10; i++)
```

```

{
    if (i>5) break;           //ndërprerja, dalja prej unaze
    cout << "i=" << i << "\n";
}
return 0;
}

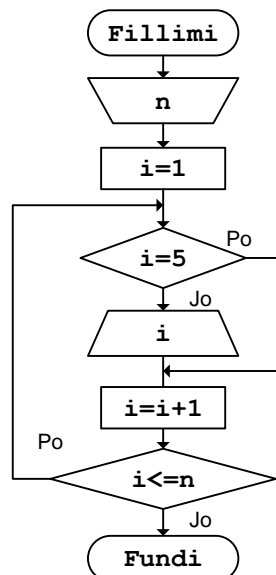
```



Sikur të mos ishte urdhëri “if”, do të shtypeshin vlerat deri në 10, mirëpo kështu porsa të kalohet vlera 5, bëhet dalja prej unaze (ndërprerja, ndalja e unazës).

### Kapërcimi i hapi të unazës (Continue)

Përmes urdhërit **Continue** (angl. Continue – vazhdo) në unazat “for” mund të bëhet kapërcimi i një hapi të unazës. Kjo mund të bëhet, përmes testimit në ndonjë kushti shtesë brenda trupit të unazës. P.sh., kemi unazën për shtypjen e numrave rendor, prej 1 deri në 10, duke e përjashtuar (anashkaluar) 5-shin. Brenda unazës, vendosim kushtin, që nëse vlera e variablës së unazës është  $i=5$ , të mos ekzekutohen urdhëri i unazës, por të rritet hapi i unazës, pra të vazhdohet në hapin tjetër të unazës ( $i=6$ ).



```

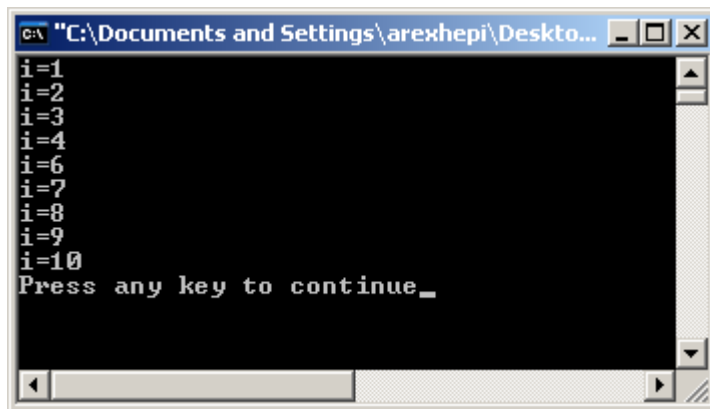
// Programi Continue - vazhdimi, kalimi i hapi të unazës
#include <iostream>
using namespace std;
int main()
{
    int i;
    for (i=1; i<=10; i++)

```

```

{
    if (i == 5) continue;    //kalo hapin, për i=5
        cout << "i=" << i << "\n";
    }
    return 0;
}

```



Rast i ngjashëm, është kur kërkohet të gjindet shuma e numrave prej 1 deri në n, por duke

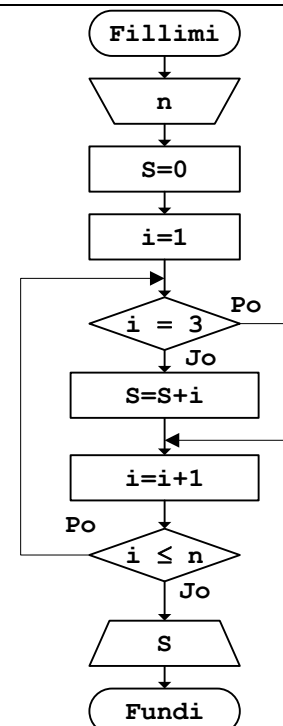
përfshiruar treshin (3), si në vijim 
$$S = \sum_{\substack{i=1 \\ i \neq 3}}^n i$$

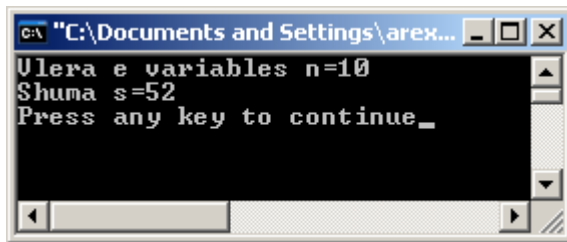
```

// Programi - Continue2
#include <iostream>
using namespace std;
int main()
{
    int i,n;
    double s=0;
    cout << "Vlera e variables n=";
    cin >> n;
    for (i=1;i<=n;i++)
    {
        if (i==3) continue;    //kaloje 3-shin
        s=s+i;
    }

    cout << "Shuma s="
        << s
        << "\n";
    return 0;
}

```





## Vargjet

Vargjet e numrave ose fushat numerike një-dimensionale (vektorët) dhe dy-dimensionale (matricat) në C++ definoohen si vijon:

```
int A[8];                /Vektori
double Z[3][5];          /Matrica
```

Indekset fillojnë me vlerën 0 (zero )

```
int A[6];    /6 anëtar - A[0], A[1], ..., A[5].
```

## Deklarimi dhe inicializimi i vargjeve

Vargjet mund t'i deklarojmë dhe aty për aty t'i inicializojmë me vlera, si në vijim:

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

char Z[7]={'d','4','*','a','G','$'}; /Vektori me karaktere

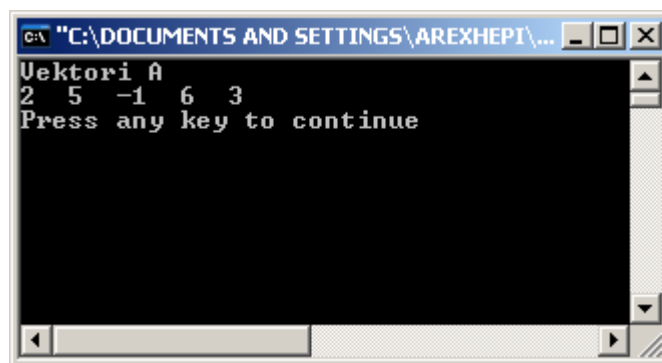
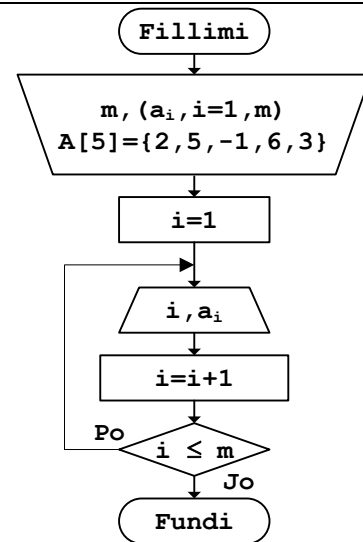
const int m=5;
int A[m]={7,2,4,1,3};

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

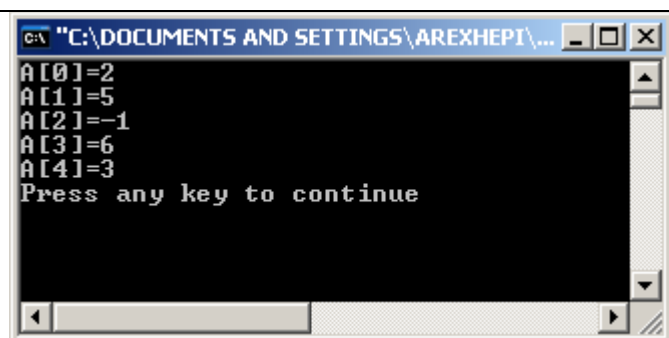


## Vektorët

```
// Programi vektor1
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i;
    //Deklarimi dhe inicializimi i vektorit
    int A[m]={2,5,-1,6,3};
    cout << "Vektori A"
        << "\n";
    for (i=0;i<m;i++)
        cout << A[i]
            << " ";
    cout << "\n";
    return 0;
}
```



```
// Programi vektor2
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i;
    int A[m]={2,5,-1,6,3};
    for (i=0;i<m;i++)
        cout << "A["
            << i
            << "]="
            << A[i]
            << "\n";
    return 0;
}
```



Vërejeni me kujdes pjesën për shtypje të anëtarëve të vektorit:

```
...
    cout << "A["
```

```

    << i
    << "]"=
    << A[i]
    << "\n";
    ...

```

e cila normalisht, mund të shkruhet në një rresht të vetëm, si:

```
cout << "A[" << i << "]"= << A[i] << "\n";
```

Pjesa e shkruar brenda thonjëzave “”, përsëritet, ashtu si edhe është dhënë, kurse pjesa jashtë thonjëzave është vlerë që mirret prej programit. Këto pjesë ndahen mes veti përmes shenjës/operatorit të ridrejtimit “<<”.

```

// Programi vektor3
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int i, A[m];

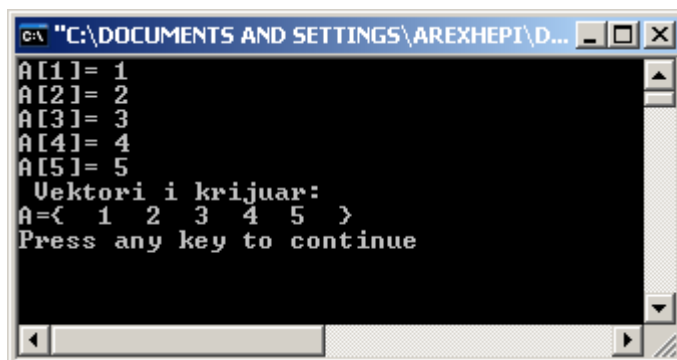
    //Krijimi i vektorit

    for (i=0;i<m;i++)
    {
        cout << "A["
            << i+1
            << "]"= ";
        cin >> A[i];
    }

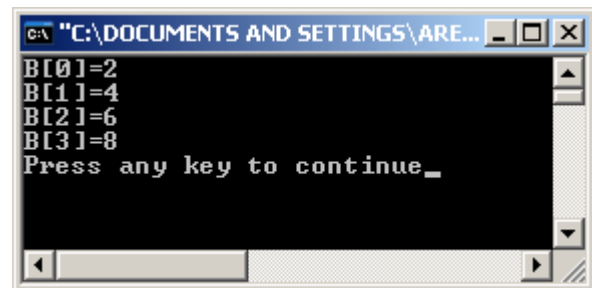
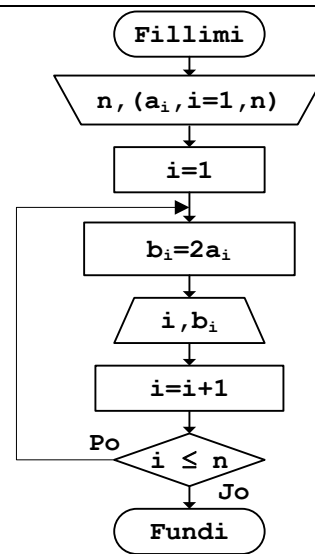
    // Shtypja e vektorit
    cout << " Vektori i krijuar: \n";
    cout << "A={ ";
    for (i=0;i<m;i++)
        cout << A[i]
            << " ";
    cout << "}\n";

    return 0;
}

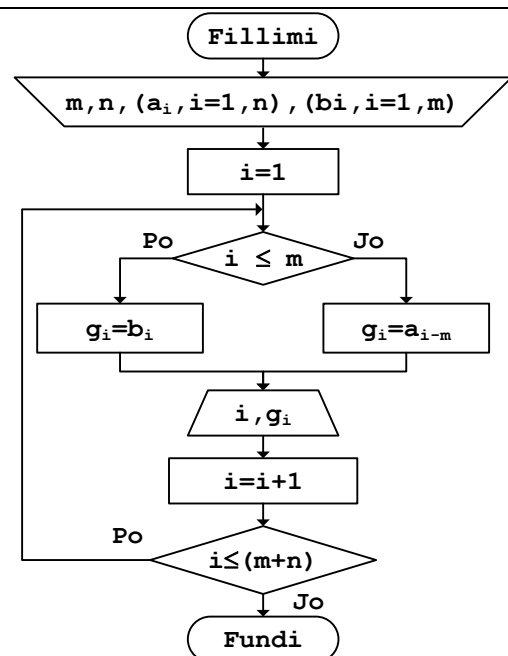
```

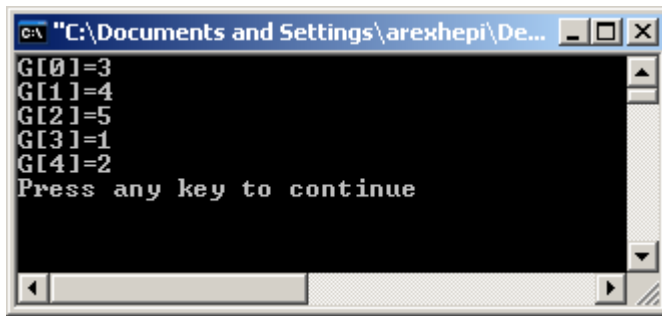


```
//Formimi i vektorit prej vektori
#include <iostream>
using namespace std;
int main()
{
    const int n=4;
    int i,A[n]={1,2,3,4},B[n];
    for (i=0;i<n;i++)
    {
        B[i]=2*A[i];
        cout << "B["
            << i
            << "]= "
            << B[i]
            << "\n";
    }
   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



```
// Bashkimi i dy vektoreve G=B & A
#include <iostream>
using namespace std;
int main()
{
    const int m=3,n=2;
    int A[n]={1,2},
        B[m]={3,4,5},
        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;
}
```

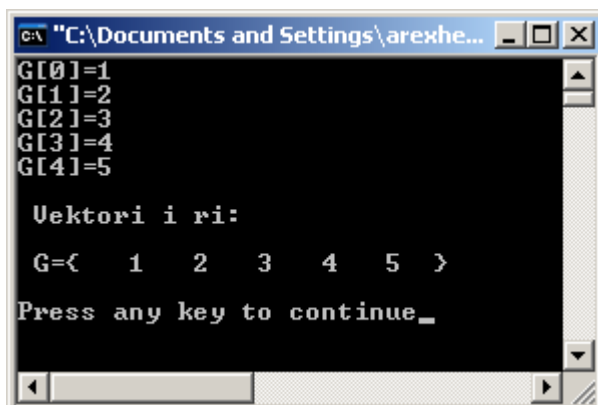




Pasi që përfundon vendosja e vektorit të parë (gjersa  $i < n$ ), për të vazhduar me vendosjen e vektorit të dytë, duhet kthyer indeksin në zero, kështu që zbritet numri i anëtarëve të vektorit të parë “m” ( $G[i]=A[i-m]$ ).

Nëse së pari vendoset vektori A, atëherë kushti do të ishte “if ( $i < n$ )” (sa numri i anëtarëve të vektorit A) dhe për kushtin e plotësuar:  $G[i]=A[i]$ , kurse kur i nuk është ( $i < n$ ), “ $G[i]=B[i-n]$ ”.

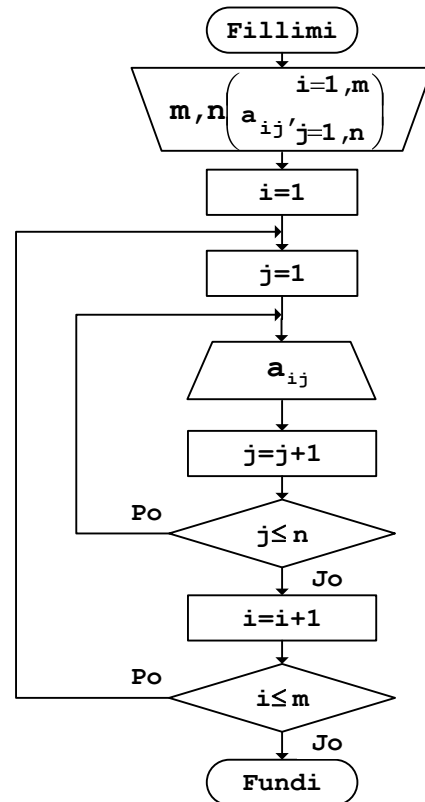
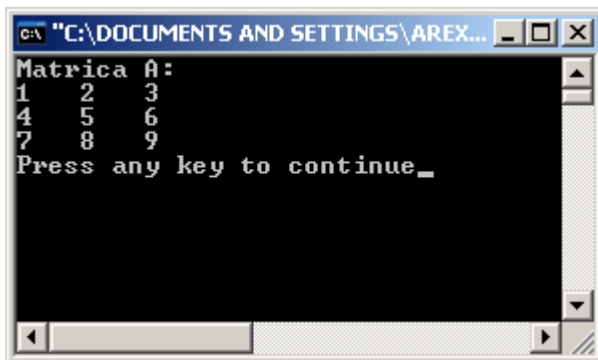
```
// Bashkimi i dy vektoreve G=A & B (Shtypja e vektorit ne fund)
#include <iostream>
using namespace std;
int main()
{
    const int m=3,n=2;
    int A[n]={1,2},
        B[m]={3,4,5},
        i,G[m+n];
    for (i=0;i<m+n;i++)
    {
        if (i<n)
            G[i]=A[i];
        else
            G[i]=B[i-n];
        cout << "G[" << i << "]= " << G[i] << "\n";
    }
    cout << "\n Vektori i ri:\n\n G=";
    for (i=0;i<m+n;i++)
    {
        cout.width(4);
        cout << G[i];
    }
    cout << "  }\n\n";
    return 0;
}
```



## Matricat

```
// Programi matricat1
#include <iostream>
using namespace std;
int main()
{
    const int m=3, n=3;
    int i,j;
    int A[m][n]={{1,2,3},
                 {4,5,6},
                 {7,8,9}};

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

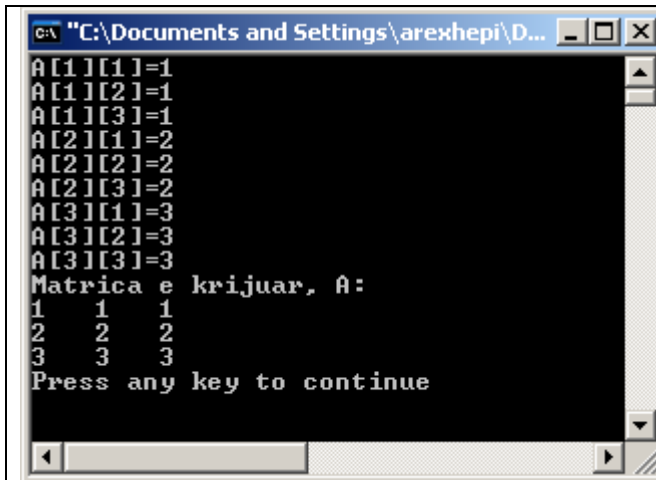


Për të krijuar matricën me vlera të çfarëdoshme, të cilat shfrytëzuesi i jep përmes tastierës, shtojmë pjesën përmes së cilës e krijojmë matricën:

```
// Programi matrica2
#include <iostream>
using namespace std;
int main()
{
    const int m=3, n=3;
    int A[m][n];
    int i,j;

    //Krijimi i matricës
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
        {
            cout << "A[" << i+1 << "][" << j+1 << "]=";
            cin >> A[i][j];
        }
    }
}
```

```
    }  
    }  
    //Shtypja e matricës  
    cout << "Matrica A: " << "\n";  
    for (i=0;i<m;i++)  
    {  
        for (j=0;j<n;j++)  
            cout << A[i][j]  
                << " ";  
        cout << "\n";  
    }  
    return 0;  
}
```



```
C:\Documents and Settings\arexhepi\D...  
A[1][1]=1  
A[1][2]=1  
A[1][3]=1  
A[2][1]=2  
A[2][2]=2  
A[2][3]=2  
A[3][1]=3  
A[3][2]=3  
A[3][3]=3  
Matrica e krijuar, A:  
1 1 1  
2 2 2  
3 3 3  
Press any key to continue
```

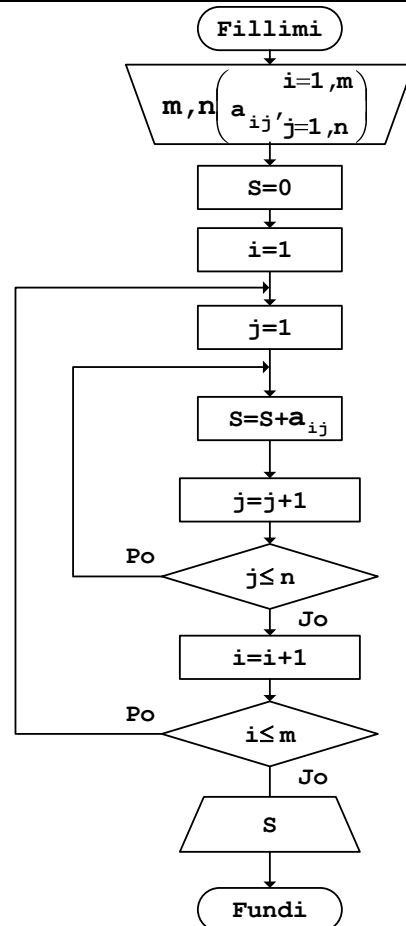
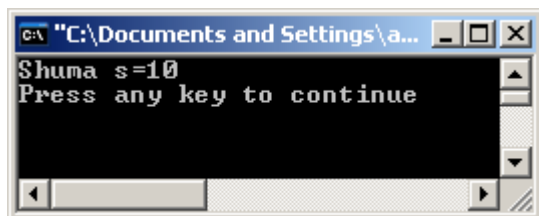
Siç shihet, për të paraqitur indeksin prej “1” e jo prej “0”, pra: A[1][1], si në matematikë, në unazat për krijimin e matricës, kemi rritur vlerat e i-së dhe j-së për “1” (i+1 dhe j+1).

## Veprimet me vektorë/matrica dhe gjetja e anëtarëve të caktuar të vektorit/matricës

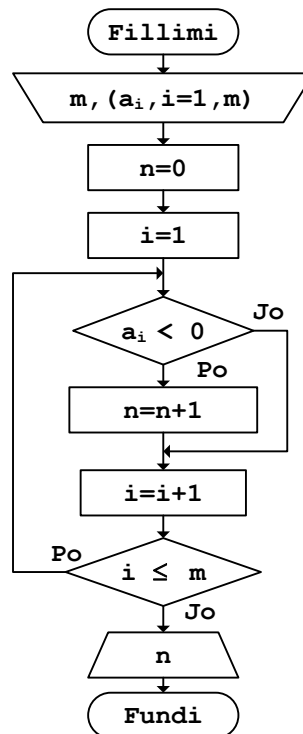
Shuma e anëtarëve të matricës:

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



Numrimi i anëtarëve bëhet duke marrë një numrator (p.sh. “n”) me vlerën fillestare  $n=0$ ; para unazës dhe pastaj në unazën e cila i përshkon të gjithë anëtarët e vargut, sa herë që gjendet një anëtarë që e plotëson kushtin e dhënë, numratori rritet për një ( $n=n+1$ ).



```

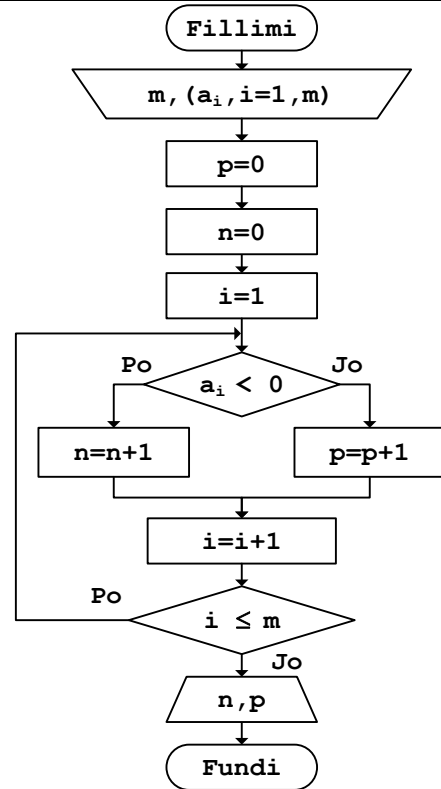
//Numri i antareve negativ te vektorit
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int A[m]={3,-2,7,-4,5},i,n;
    n=0; //numratori - fillimisht vlera zero
    for (i=0;i<m;i++) //unaza që përshkon vargun
        if (A[i]<0) //kushti për numrim
            n=n+1; //rritja e numratorit
    cout << "Anëtarë negativë n="
         << n
         << "\n";
    return 0;
}
  
```



```

//Numri i antareve negativ
//dhe pozitiv te vektorit
#include <iostream>
using namespace std;
#include <cmath>
int main()
{
    const int 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;
}

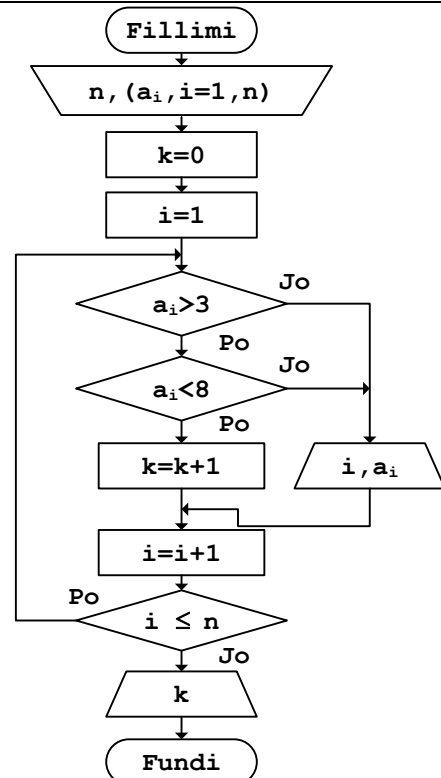
```



```

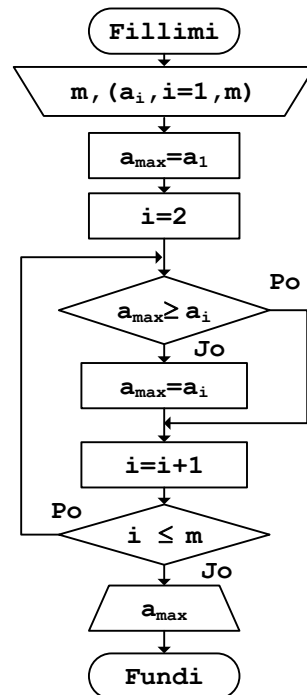
//Numri i anetareve mes 3 dhe 8
#include <iostream>
using namespace std;
#include <cmath>
int main()
{
    const int n=5;
    int A[n]={5,2,4,9,-7},i,k;
    k=0;
    for (i=0;i<n;i++)
    {
        if ((A[i]>3) && (A[i]<8))
            k=k+1;
        else
            cout << "i="
                << i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

```



## Anëtari më i madh (më i vogël)

Gjetja e anëtarit minimal/maksimal bëhet, duke e deklaruar fillimisht anëtarin e parë të vargut, si anëtari minimal/maksimal: p.sh.,  $A_{max}=A[0]$ ; dhe pastaj përmes unazës që i përshkon të gjithë anëtarët tjerë të vargut, e krahasojmë vlerën momentale të  $A_{max}$  me të gjithë anëtarët tjerë të vargut. Nëse gjindet ndonjë që është më i madh (për rastin minimal më i vogël) se anëtari momental maksimal, atëherë  $A_{max}$ , e merr vlerën e tij dhe krahasimi vazhdon me anëtarët vijues të vargut.



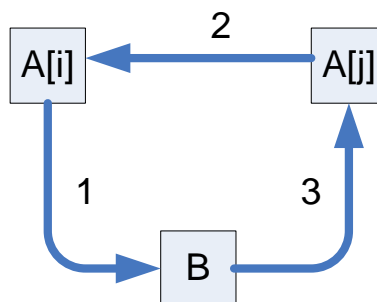
```

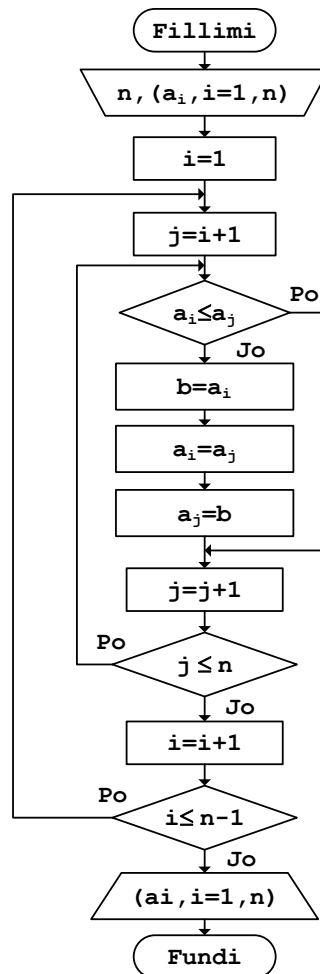
//Anetari me i madh i vektorit
#include <iostream>
using namespace std;
int main()
{
    const int m=4;
    int A[m]={3,7,5,9}, i, Amax;
    Amax=A[0];
    for (i=1; i<m; i++)
    {
        if (Amax < A[i]) //nëse ndonjë anëtar është më i madh
            Amax = A[i]; //Amax e merr vlerën e tij
    }
    cout << "Numri më i madhë Amax ="
         << Amax
         << "\n";
    return 0;
}
  
```

## Sortimi – radhitja sipas madhësisë

Një prej mënyrave të sortimit të anëtarëve të vargut realizohet kështu: fillohet me anëtarin në pozitën e parë dhe bëhet krahasimi i anëtarë nga pozita e parë, me të gjithë anëtarët e vargut. Pra: i pari-i dyti, i pari-i treti, ... i pari-i fundit. Sa herë që vlerat që krahasohen nuk janë të renditura si duhet, ua ndërrojmë vendet. Kështu, pasi të gjindet anëtar i më i vogël (ose për renditjen e kundërt, anëtar i më i madh) ai vendoset definitivisht në pozitën e parë dhe përcaktohet anëtar i më i vogël. Pastaj, vazhdon procedura e njëjtë, për të gjetur anëtarin e ardhshëm më të vogël, për pozitën e dytë. Pra, bëhet krahasimi i anëtarit në pozitën e dytë, me të gjithë anëtarët e ardhshëm: i dyti-i treti, i dyti-i katërti, ... i dyti – i fundit, dhe njësoj, sa herë ka nevojë, dy anëtarëve që krahasohen u ndërrohen vendet. Kur, përfundon gjetja e anëtarit për pozitën e dytë, procedura e njëjtë përsëritet për anëtarët vijues. Nëse vargu ka  $n$  anëtarë, atëherë duhet të bëhen  $n-1$  kalime të tilla (sepse krahasimi nuk bëhet me vet-vehten dhe në fund krahasohet anëtar i parafundit me atë të fundit).

Për krahasim, marrim dy numrator:  $i$  dhe  $j$ , dhe përdorën dy unaza (unazë brenda unazës), ku unaza për  $i$  fillon prej pozitës së parë dhe shkon deri tek pozita e parafundit, kurse  $j$  fillon prej pozitës së dytë ( $i+1$ ) dhe shkon deri tek pozita e fundit. Për secilin hap të unazës së jashtme,  $i$ , unaza e brendshme,  $j$ , fillon prej pozitës për një më të madhe: ( $j=i+1$ ). Në secilin hap, krahasohen  $A[i] > A[j]$ . Nëse ka nevojë që t'u ndërrohen vendet, atëherë ndërrimi i vendeve bëhet duke përdorur një variabël ndihmëse, (si në figurë) e cila së pari e merr vlerën e anëtarit të parë (që të mos humbet ajo  $B=A[i]$ ), pastaj anëtar i parë e merr vlerën e anëtarit të dytë ( $A[i]=A[j]$ ), dhe në fund, anëtar i dytë, e merr vlerën e anëtarit të parë prej variablës ndihmëse, e cila e ka ruajtur atë përkohësisht ( $A[j]=B$ ). Kjo është e ngjashme me procedurën e ndërrimit të përmbajtjes së dy enëve. Nuk mund të ndërrohen enët e tyre, pa e pasur së pari një enë të tretë ndihmëse, në të cilën e bartim përmbajtjen e enës së parë, pastaj atë prej së dytës e kalojmë në enën e parë, kurse në fund, prej enës ndihmëse, e kthejmë përbërjen e parë në enën e dytë.





```
//Sortimi i anetareve te vektorit
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    const int 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])
```

```
            {
```

```
                b=A[i];
```

```
                A[i]=A[j];
```

```
                A[j]=b;
```

```
            }
```

```
        }
```

```
    cout << "A=[  ";
```

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

```
    cout << A[i]
```

```
        << "  ";
```

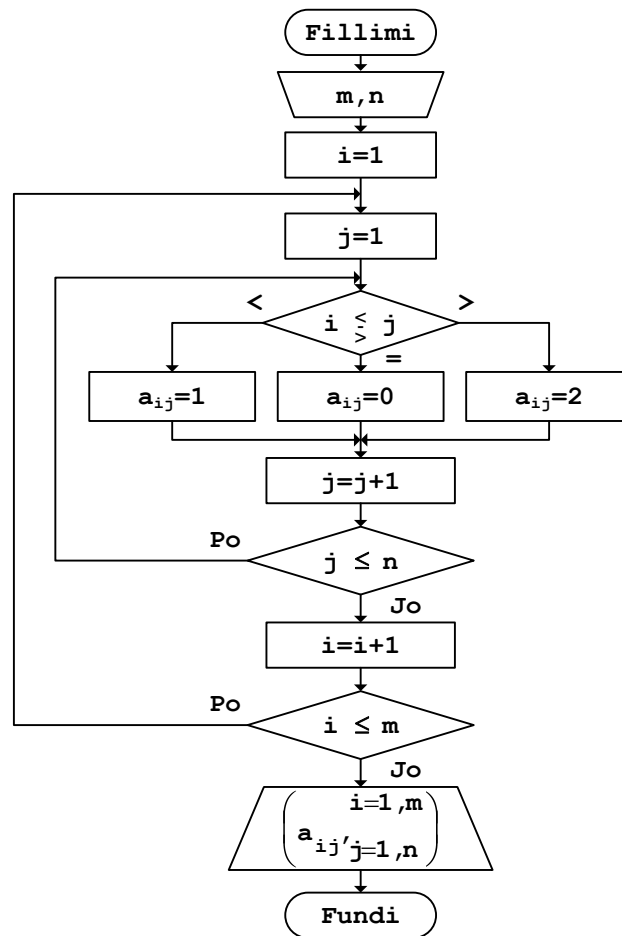
```
    cout << "]"
```

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

```
    return 0;
```

```
}
```

## Formimi i fushave numerike

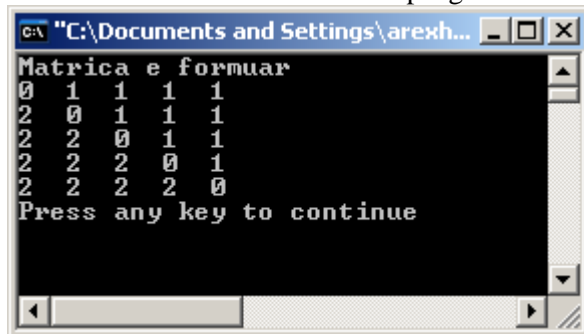


```

//Formimi i matrices
#include <iostream>
using namespace std;
#include <cmath>
int main()
{
    const int m=5,n=5;
    int i,j,A[m][n];
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
            if (i<j)
                A[i][j]=1;
            else
                if (i==j)
                    A[i][j]=0;
                else
                    A[i][j]=2;
    cout << "Matrica e formuar"
         << "\n";
}
  
```

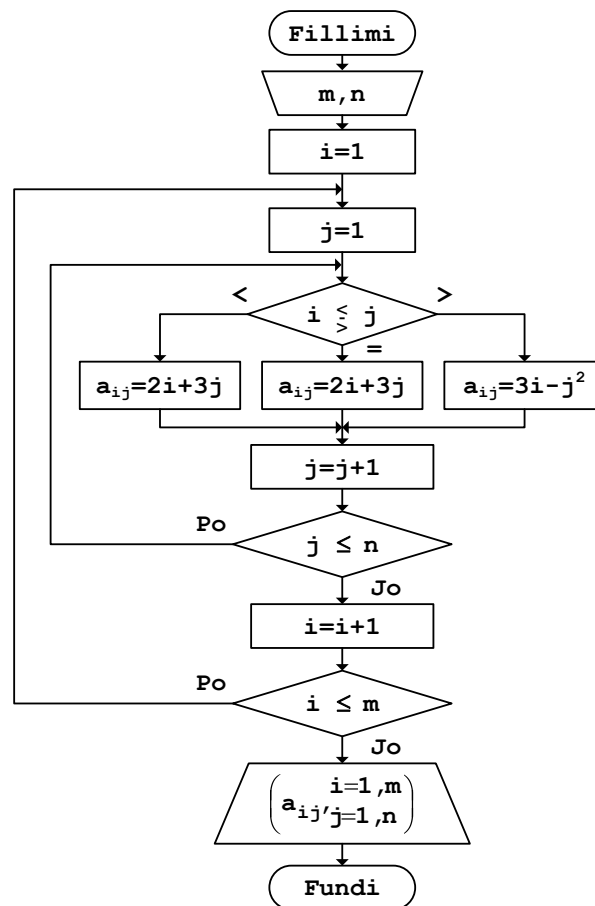
```
for (i=0; i<m; i++)  
{  
    for (j=0; j<n; j++)  
        cout << A[i][j]  
            << " ";  
    cout << "\n";  
}  
return 0;  
}
```

Rezultati i ekzekutimit te programit:



#### Shembull:

Keni kujdes, mos te “habiteni” nga teksti i detyres, si p.sh: Te formohet matrica e cila anëtarët e diagonales dhe ata mbi diagonale i ka sa “shuma e dy-fishit të rreshtit dhe trefishit të kolonës (shtyllës)”, kurse anëtarët nën diagonale i ka sa “diferenca e trefishit të rreshtit dhe kubit të kolonës (shtyllës)”. Në rastet e ngjashme, gjithmoë duhet të keni parasyshë, që për anëtarët  $A_{ij}$  (ose si shënohen në program:  $A[i][j]$ ) të matricës  $A[m][n]$  rreshtin e përfaqëson indeksi i parë “i”, kurse shtyllën indeksi “j”, prandaj do të kemi:

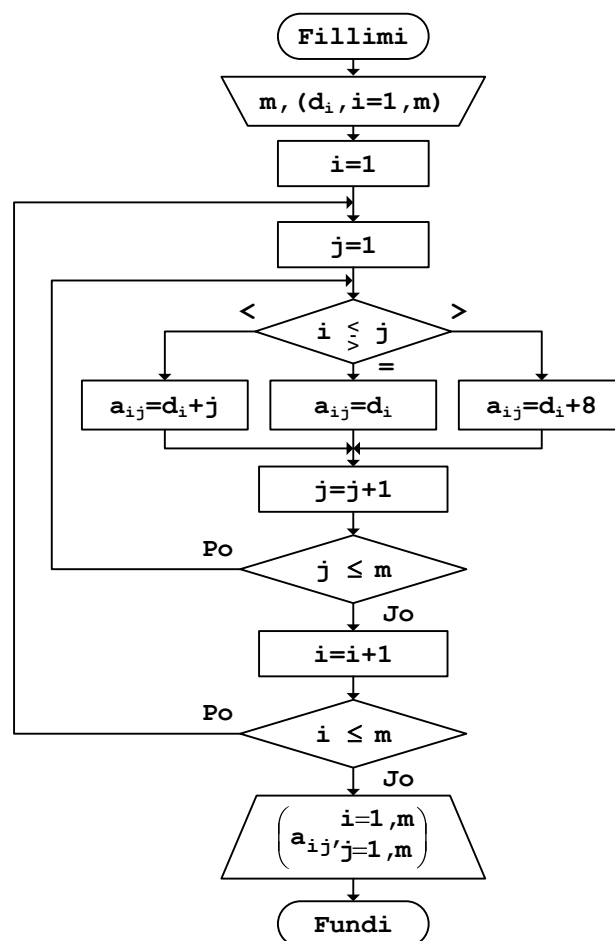


```

//Formimi i matrices
#include <iostream>
#include <math>
using namespace std;
int main()
{
    const int m=4,n=4;
    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;
}
  
```

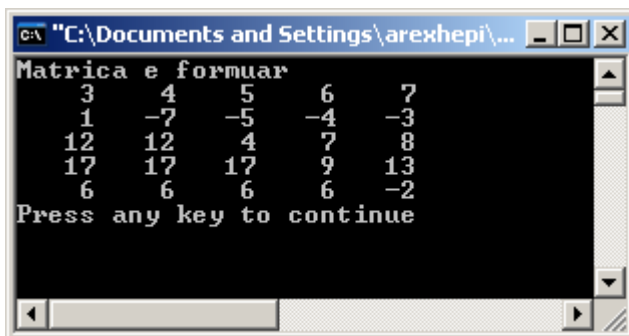
```

C:\Documents and Settings\arexhepi...
Matrica e formuar
0 3 6 9
3 2 8 11
6 5 4 13
9 8 5 6
Press any key to continue
  
```

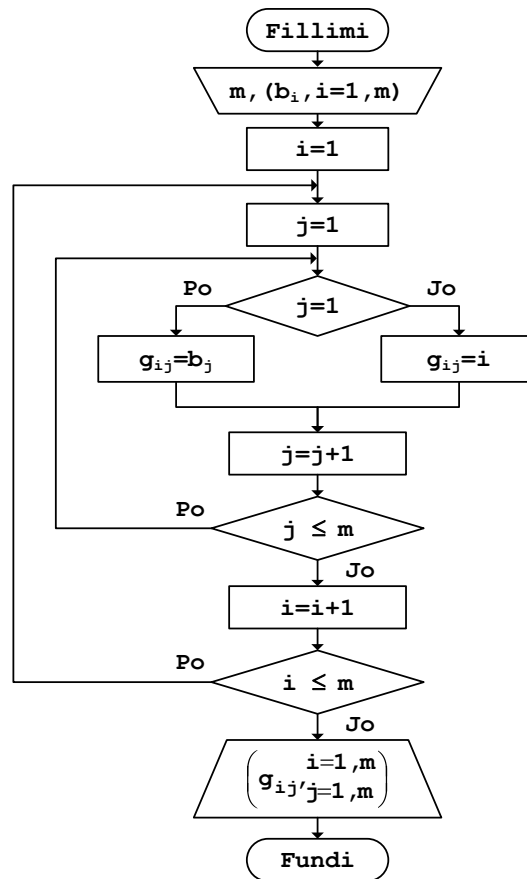




```
//Formimi i matrices prej vektorit
#include <iostream>
using namespace std;
int main()
{
    const int m=5;
    int D[m]={3,-7,4,9,-2};
    int i,j,A[m][m];
    for (i=0;i<m;i++)
        for (j=0;j<m;j++)
            if (i<j)
                A[i][j]=D[i]+j;
            else
                if (i==j)
                    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;
}
```



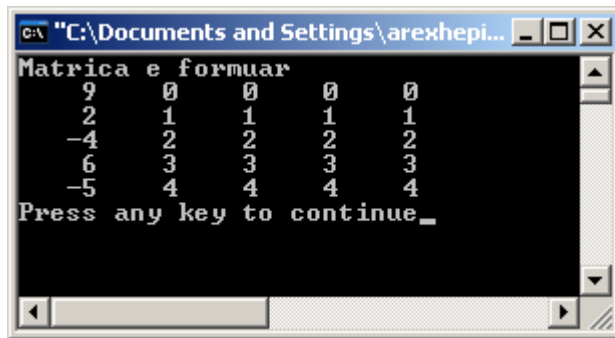
```
Matrica e formuar
 3  4  5  6  7
 1 -7 -5 -4 -3
12 12  4  7  8
17 17 17  9 13
 6  6  6  6 -2
Press any key to continue
```



//Formimi i matrices prej vektorit - Kolona e pare, vektorit

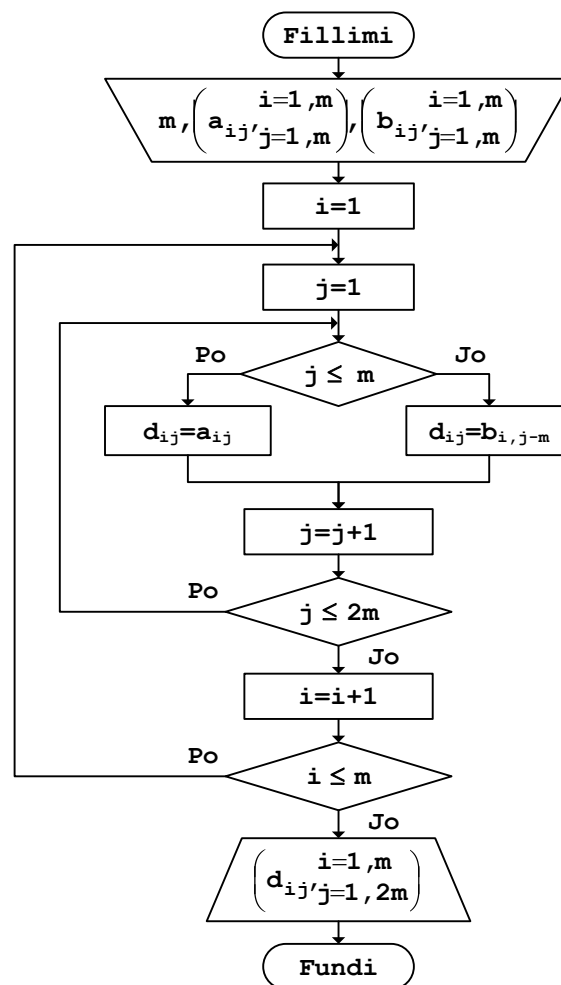
```

#include <iostream>
using namespace std;
int main()
{
    const int 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;
}
  
```

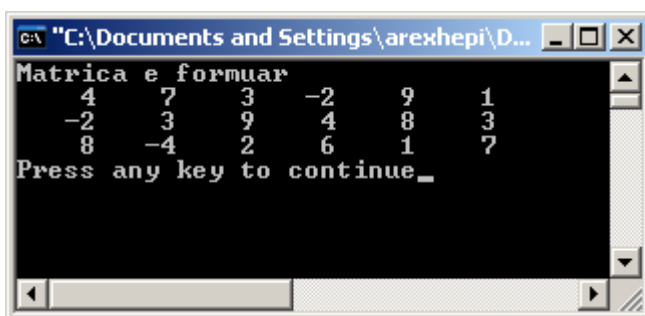


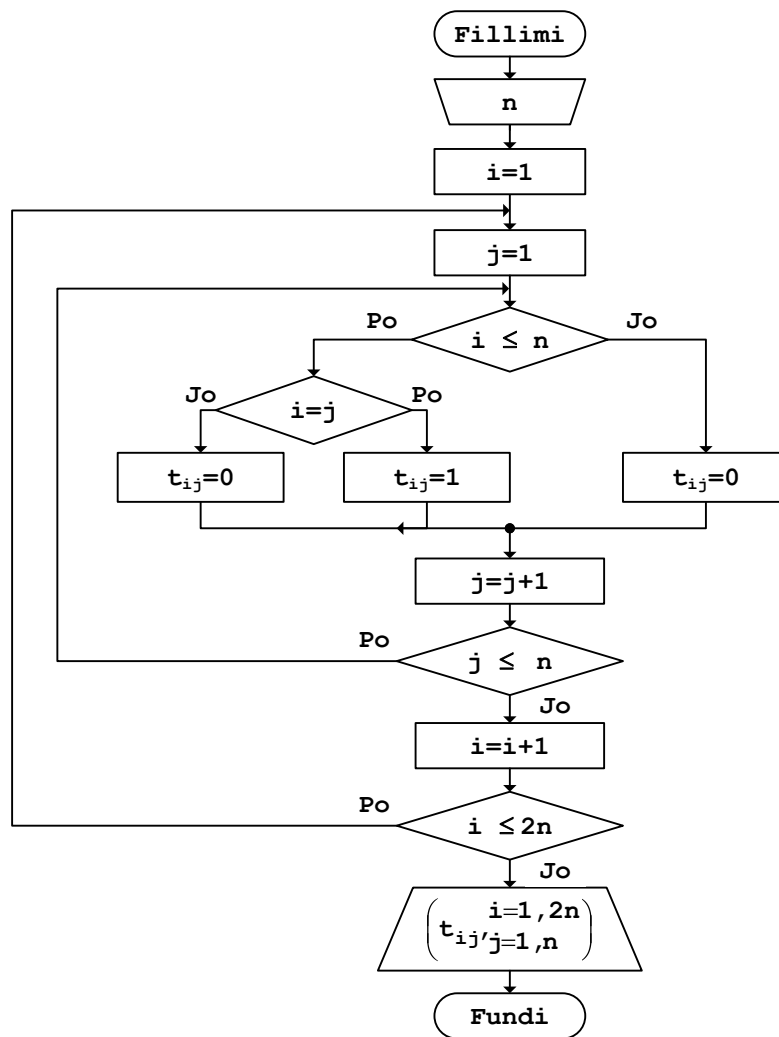
### Bashkimi i dy matricave

$a_{0,0}$	...	$a_{0,n-1}$	$b_{0,0}$	...	$b_{m-1,n-1}$
...	...	...	...	...	...
$a_{m-1,0}$	...	$a_{m-1,n-1}$	$b_{m-1,0}$	...	$b_{m-1,n-1}$



```
//Bashkimi i dy matricave
#include <iostream>
using namespace std;
int main()
{
    const int 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(5);
            cout << D[i][j];
        }
        cout << "\n";
    }
    return 0;
}
```





```
//Matrica e formuar prej dy matricave
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    const int 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"
```

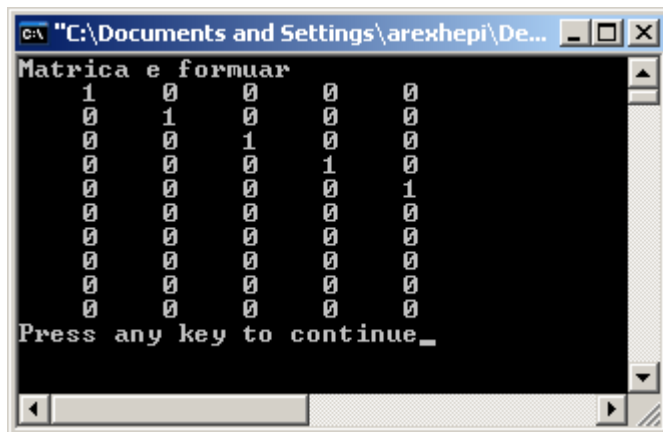
```
        << "\n";
```

```
    for (i=0;i<(2*n);i++)
```

```
    {
```

```
        for (j=0;j<n;j++)
```

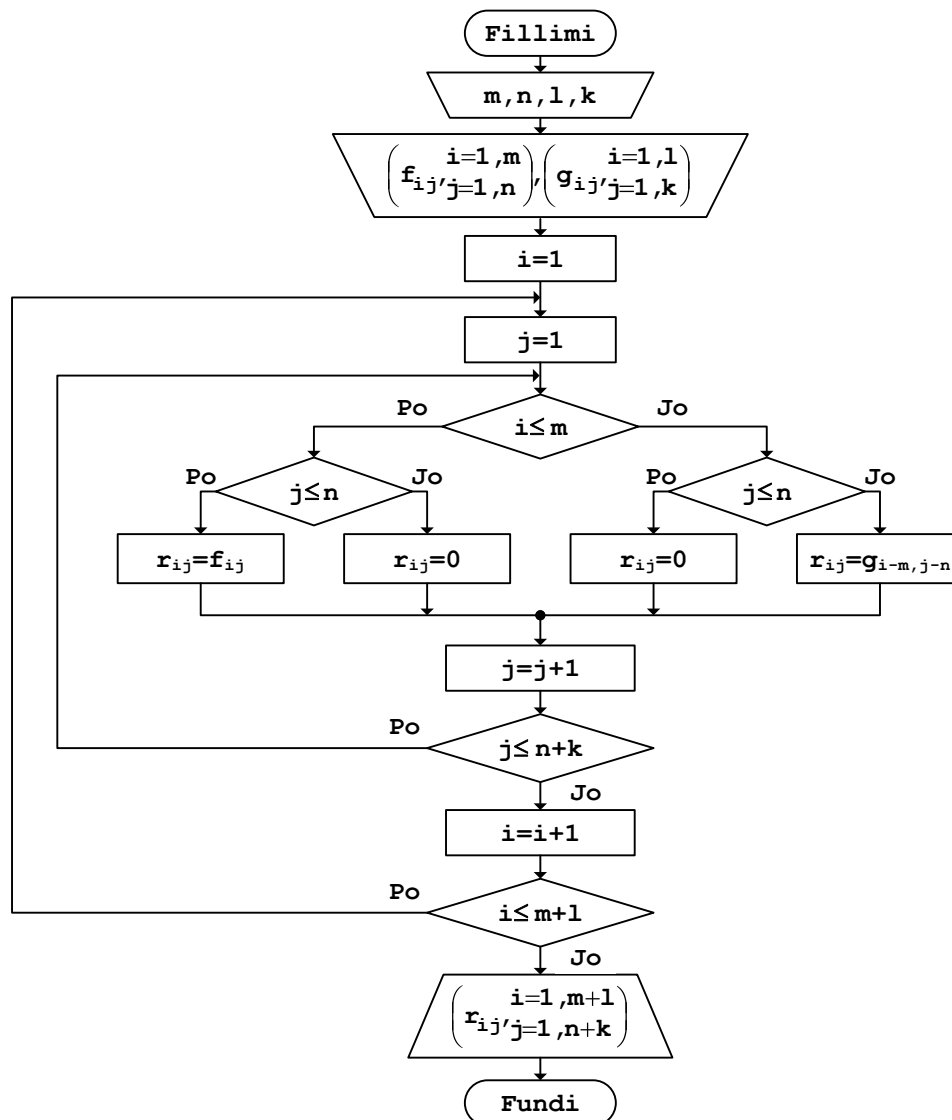
```
{  
    cout.width(5);  
    cout << T[i][j];  
}  
cout << "\n";  
}  
return 0;  
}
```



```
C:\Documents and Settings\arexhepi\De...  
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  
Press any key to continue_
```

## Bashkimi i dy matricave – në diagonale

F	0
0	G



```
//Bashkimi i dy matricave - në diagonale
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    const int 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+1][n+k];
    for (i=0;i<m+1;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+1;i++)
    {
        for (j=0;j<n+k;j++)
        {
            cout.width(5);
            cout << R[i][j];
        }
        cout << "\n";
    }
    return 0;
}

```

```

C:\Documents and Settings\arexhepi\Desktop\Prova C...
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
Press any key to continue.

```



## Krijimi i vektorit prej anëtarëve të matricës

Një rast specifik është krijimi i vektorit prej anëtarëve të matricës ose prej disa anëtarëve të caktuar të matricës, p.sh., vetëm prej anëtarëve negativ të matricës. Duhet pasë parasysh se si do të bëhet indeksimi i anëtarëve të matricës, në vektor. P.sh., le të themi se e kemi matricën

$A(m,n)$  dhe prej saj do të krijojmë vektorin  $B(p)$ , ku  $p$  do të jetë:  $p=m*n$ , për arsye se matrica i ka  $m*n$  anëtarë. Për anëtarët e vektorit, do të kemi:

$B[0] = A[0,0]$ ,  $B[1] = A[0,1]$ , ...  $B[p-1] = A[m-1,n-1]$ . Nëse marrim si shembull matricën me 3 rreshta dhe 3 kolona,  $A(3,3)$ , atëherë do të formohet vektorin  $B(9)$ , dhe do të kemi:

$A[0][0]$	$A[0][1]$	$A[0][2]$	=	1	2	3
$A[1][0]$	$A[1][1]$	$A[1][2]$		4	5	6
$A[2][0]$	$A[2][1]$	$A[2][2]$		8	7	9

$B[9] = (1,2,3,4,5,6,7,8,9)$ , ku indeksat dhe vlerat e anëtarëve do të jenë:

$B[0]=1$   $B[1]=2$   $B[2]=3$   $B[3]=4$   $B[4]=5$   $B[5]=6$   $B[6]=7$   $B[7]=8$   $B[8]=9$

Pra, do të kemi:

$B[0] = A[0,0] = 1$   
 $B[1] = A[0,1] = 2$   
 $B[2] = A[0,2] = 3$   
 $B[3] = A[1,0] = 4$   
 $B[4] = A[1,1] = 5$   
 $B[5] = A[1,2] = 6$   
 $B[6] = A[2,0] = 7$   
 $B[7] = A[2,1] = 8$   
 $B[8] = A[2,2] = 9$

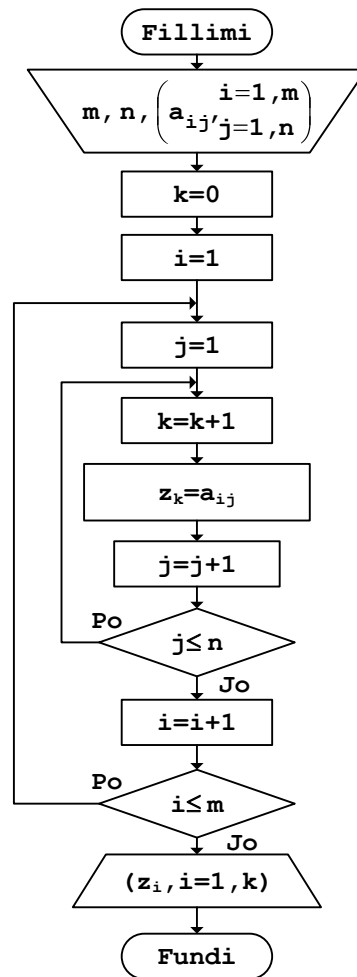
Për të kaluar me radhë nëpër anëtarët e matricës, e dimë se na duhen dy unaza (unazë, brenda unazës), për  $i$  dhe për  $j$ . Pra, numratorët  $i$  dhe  $j$ , na shërbejnë për indeksat e anëtarëve të matricës. Për t'ua ndarë këto vlera anëtarëve përkatës të vektorit, për indeksat e vektorit duhet të definohet një numrator  $k$ , p.sh.,  $k$ , i cili do të rritet për secilin anëtarë të matricës, si në pjesën vijuese të programit:

```

for (i=0; i<m; i++)           //unaza për rreshtat e matricës - i
    for (j=0; j<n; j++)       //unaza për kolonat e matricës - j
    {
        k=k+1;                //numratori për anëtarët e vektorit - k
        B[k]=A[i][j];         //ndarja e vlerave anëtarëve të vektorit
    }

```

Pra, kjo pjesë e programit, do të krijojë vektorin, prej matricës.  
Shembull:



```

//Krijimi i vektorit prej anetareve te matrices
//Rreshtimi i anetareve te matrices ne vektor
#include <iostream>
using namespace std;
int main()
{
    const int m=4,n=3;
    int A[m][n]={ {1,2,3},
                  {4,5,6},
                  {7,8,9},
                  {10,11,12}
                };

    int i,j,k,B[m*n];

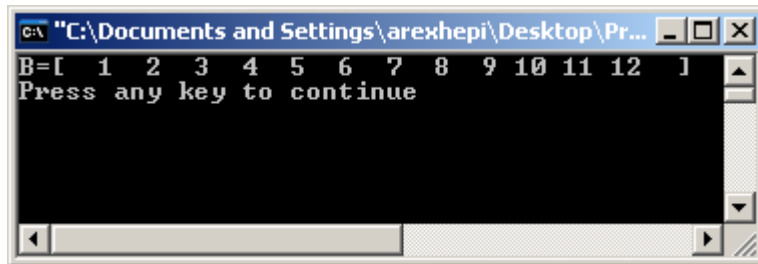
    //Krijimi vektorit
    k=-1;
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
        {
            k=k+1;
            B[k]=A[i][j];
        }

    //Shtypja e vektorit të krijuar
    cout << "B=";
  
```

```

    for (i=0;i<=k;i++)
    {
        cout.width(3);
        cout << B[i];
    }
    cout << "  ]\n";
    return 0;
}

```



Ose, per rastin e vektorit me vlera te çfaredoshme:

```

//Krijimi i vektorit prej anetareve te matrices
//Rreshtimi i anetareve te matrices ne vektor
#include <iostream> using namespace std;
int main()
{
    const int m=4,n=3;
    int A[m][n]={    {3,-4,6},
                     {-9,1,2},
                     {7,-8,1},
                     {-2,5,-3}
                    };

    int i,j,k,B[m*n];

//Krijimi  vektorit
    k=-1;
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
        {
            k=k+1;
            B[k]=A[i][j];
        }

//Shtypja e vektorit të krijuar
    for (i=0;i<=k;i++)
    {
        cout.width(3);
        cout << "B[" << i <<"]="<<B[i]<<"\n";
    }

    return 0;
}

```

```

C:\Documents and Settings\arexhepi...
B[0]=3
B[1]=-4
B[2]=6
B[3]=-9
B[4]=1
B[5]=2
B[6]=7
B[7]=-8
B[8]=1
B[9]=-2
B[10]=5
B[11]=-3
Press any key to continue_

```

Në fillim, para unazave për  $i$  dhe për  $j$ , kemi marrë  $k=-1$ ; , meqenë se, në pjesën për krijimin e anëtarëve të vektorit, së pari e kemi rritur numratorin  $k=k+1$ ; , e pastaj e kemi caktuar anëtarin:  $B[k]=A[i][j]$ ; kështu që të fillojmë prej indeksit të parë:  $k=0$ ; dhe  $B[0]=A[0][0]$ ; .

Nëse, para unazës, numratorin e fillojmë prej,  $k=0$ ; atëherë, në unazë, së pari e japim urdhërin:  $B[k]=A[i][j]$ ; e pastaj, e rrisim numratorin:  $k=k+1$ ;

Pra, ekzistojnë dy opsione:

1. Fillohet prej  $k=-1$ , por në unazë së pari rritet numratori, e pastaj ndahet vlera (krijohet anëtari i vektorit), ose
2. Fillohet prej  $k=0$ , por në unazë së pari ndahet vlera (krijohet anëtari i vektorit), e pastaj rritet numratori  $k$ .

<pre> k=-1; for (i=0; i&lt;m; i++)     for (j=0; j&lt;n; j++)     {         k=k+1;         B[k]=A[i][j];     } </pre>	<pre> k=0; for (i=0; i&lt;m; i++)     for (j=0; j&lt;n; j++)     {         B[k]=A[i][j];         k=k+1;     } </pre>
---	--

Kjo është me rëndësi, sepse kur të shtypen pastaj anëtarët e vektorit, me pjesën:

```

//Shtypja e vektorit
for (i=0; i<=k; i++)
{
    cout.width(3);
    cout << "B[" << i << "]=" << B[i] << "\n";
}

```

duhet pasë parasysh, se ku ka mbetur vlera e fundit e numratorit  $k$ , prej unazave paraprake për  $i$  dhe për  $j$ , që ishin për krijim të anëtarëve të vektorit. Kur fillohet prej  $k=-1$ , atëherë vlera e fundit e indeksit të vektorit, do të jetë pikërisht  $k$ , kështu që kur të shtypet vektori, unaza shkon deri në  $k$ , pra duke e përfshirë edhe vlerën e  $k$ -së,  $i<=k$ : `for (i=0; i<=k; i++)` .

Mirëpo, në rastin kur fillon numratori prej  $k=0$ , atëherë duke e rritur vlerën e numratorit  $k$ , për secilin anëtarë të ardhshëm të matricës, në fund numratori rritet edhe një herë, pas krijimit të anëtarit të fundit të vektorit. Kështu, kur të shtypet me unazën vijuese vektori, për të shtypur numrin e saktë të anëtarëve të vektorit, unaza shkon si:

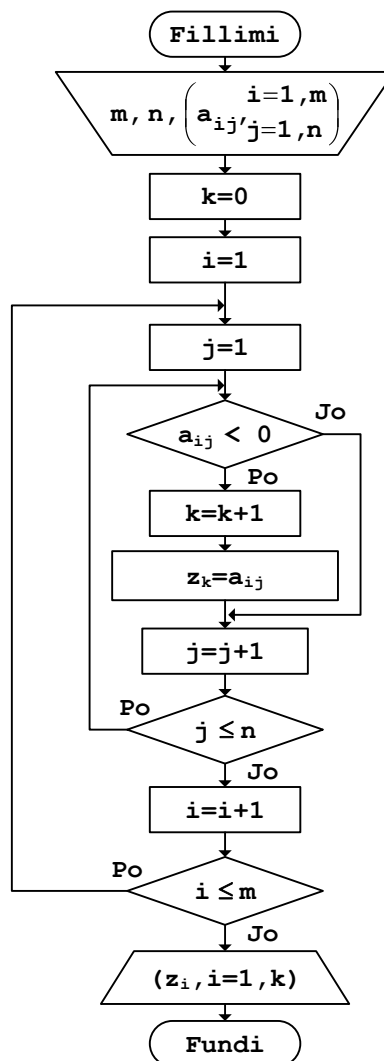
```
for (i=0; i<k; i++)
```

pra,  $i < k$ ; pa e përfshirë edhe vlerën e fundit,  $k$ , sepse ajo është për një më e madhë se indeksi  $i$  fundit i vektorit të formuar. Me shembuj konkret, do të shihen më mirë pasojat e kësaj zgjedhjeje.

Kryesorja, është me rëndësi të dihet, se numratori  $k$ , na e tregon numrin e anëtarëve të vektorit të formuar. Kjo është sidomos e rëndësishme, kur vektori formohet duke zgjedhur vetëm anëtarët e caktuar të matricës, p.sh., vetëm ata negativ. Sepse, vetëm duke e përcjelluar numratorin  $k$ , do ta dijmë se sa anëtarë të matricës, kanë kaluar në vektor.

Fillimisht, meqenëse nuk e dijmë se sa anëtarë të matricës do të kalojnë në vektor, atëherë e rezervojmë (deklarojmë) hapësirën për vektorin  $B$ , si: `int B[m*n]`, sa është numri i të gjithë anëtarëve të matricës, që njëherit është edhe maksimumi i mundshëm, për rastin special kur të gjithë anëtarët e matricës e plotësojnë kushtin për të kaluar në vektor.

Shembull:

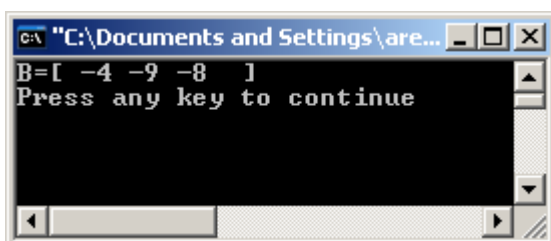


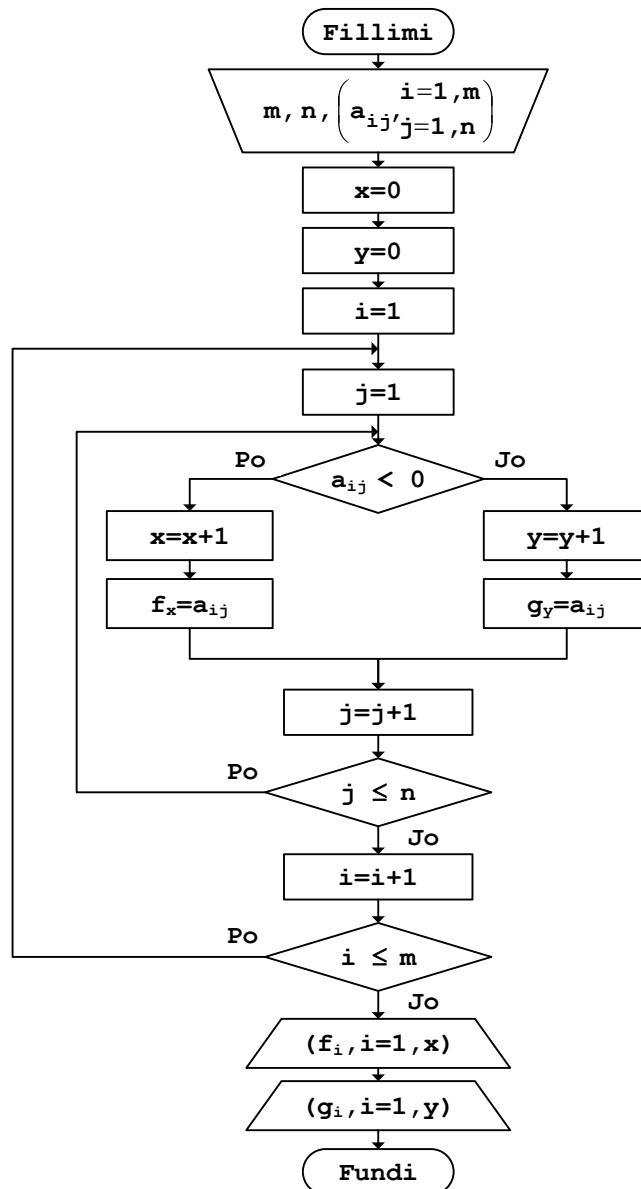
```
//Krijimi i vektorit prej anetareve negativ te matrices
#include <iostream>
using namespace std;
int main()
{
    const int m=3,n=3;
    int A[m][n]={    {3,-4,6},
                     {-9,1,2},
                     {7,-8,1}
                   };

    int i,j,k,B[m*n];

//Krijimi i vektorit
    k=-1;
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
            if (A[i][j]<0)
            {
                k=k+1;
                B[k]=A[i][j];
            }
    cout << "B=[";

//Shtypja e vektorit
    for (i=0;i<=k;i++)
    {
        cout.width(3);
        cout << B[i];
    }
    cout << "  ]\n";
    return 0;
}
```





```

// Krijimi i vektoreve me anetaret pozitiv dhe negativ te matricave
#include <iostream>
using namespace std;
int main()
{
    const int m=4,n=3;
    int A[m][n]={
        {3,-4,6},
        {-9,1,2},
        {7,-8,1},
        {-2,5,-3}
    };
    int i,j,x,y,F[m*n],G[m*n];
    x=-1;
    y=-1;
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
            if (A[i][j]<0) //Krijimi i vektorit negativ
    {

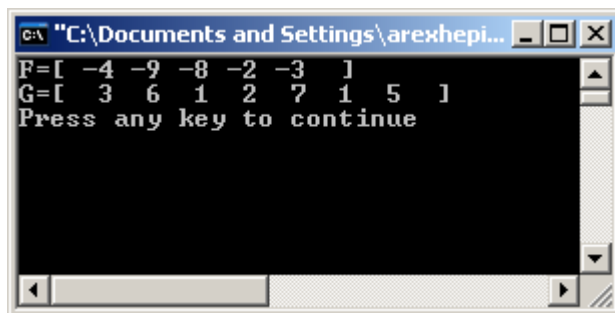
```

```
        x=x+1;
        F[x]=A[i][j];
    }
    else //krijimi i vektorit pozitiv
    {
        y=y+1;
        G[y]=A[i][j];
    }

    cout << "F=";

//Shtypja e vektorit negativ
    for (i=0;i<=x;i++)
    {
        cout.width(3);
        cout << F[i];
    }
    cout << " ]\n";

    cout << "G=";
//Shtypja e vektorit pozitiv
    for (i=0;i<=y;i++)
    {
        cout.width(3);
        cout << G[i];
    }
    cout << " ]\n";
    return 0;
}
```

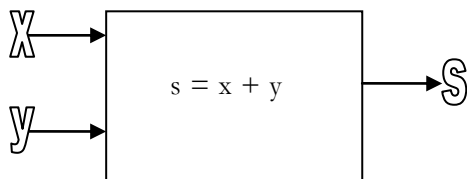




## Funksionet

Funksioni është një pjesë e veçantë e programit (një nën-program), një modul që funksionon i pavarur nga pjesa tjetër e programit, duke pranuar parametrat hyrës (të definuar formalisht) dhe duke kthyer rezultatin e llogaritjes së bërë përbrenda tij, në pjesën e programit e cila e ka thirrur funksionin. Pra, funksioni duket si një kuti e mbyllur, e cila i ka hyrjet, “kyçjet” për vlerat hyrëse dhe daljen për kthimin e rezultatit të funksionit në programin kryesor. P.sh., nëse kemi një kuti (funksion) me emrin “Shuma”, që kërkon dy vlera hyrëse, “x” dhe “y” (si në figurë), për të kthyer (për të dhënë në dalje) rezultatin “s”, që është shuma e dy vlerave hyrëse, thuhet se kemi funksionin me dy parametra hyrës (të cilët i deklarojmë formalisht me çfarëdo emri, prandaj edhe quhen parametra formal). Për të fituar rezultatin në dalje, kësaj kutije duhet sjellë në hyrje dy vlera (variabla aktuale, për të cilat e dëshirojmë rezultatin e shumës). Se çka ndodhë në brendi të kutisë (funksionit) nuk na intereson, kryesorja e dijmë se nëse i japim në hyrje dy vlera (x dhe y), në dalje e fitojmë shumën e tyre (x+y). Prandaj, sa herë që programi ka nevojë për shumën e dy numrave, ai e thërret funksionin: “Shuma (x,y)” duke ia përcjellur atij vlerat aktuale për parametrat formal, p.sh. **Shuma(7,3)**, (pra kemi për x=7, dhe për y=3).

Funksioni thirret me aq variabla, me sa është deklaruar. Pra, patjetër duhet t’i jipen saktësisht aq variabla (vlera aktuale), sa parametra formal t’i ketë të deklaruar. Përmes shembujve, do të jetë e qartë se si deklarohet dhe si thirret funksioni.



Forma e përgjithshme e funksionit:

```

tipi emri(tipi1 f1, tipi2 f2, ..., tipin fn)
{
    urdhëri/at;
    return rezultati;
}
  
```

ku janë:

tipi - tipi i rezultatit të funksionit.

emri - emri i funksionit.

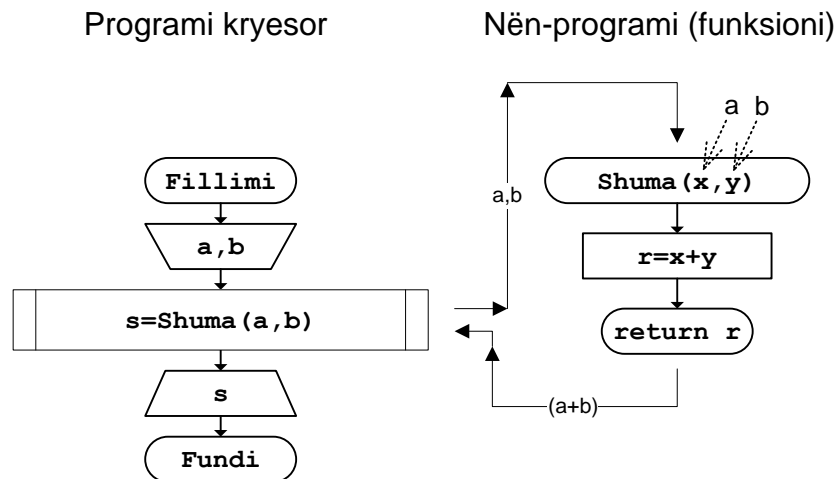
tipi1,... tipin - tipet e parametrave formal.

f1, f2, ..., fn - parametrat formal.

urhdëri/at - urhdëri/at që ekezekutohen brenda funksionit

r - rezultati të cilin e kthen funksioni.

**Bloku skema e programit më nënprogram duket si në vijim:**



Pra, në skema, blloku për thirrje të nënprogrameve paraqitet si :



Programi kryesor e thërret funksionin (nënprogramim) për t'ia kryer një llogaritje. Funksioni e kryen llogaritjen duke i përdorur parametrat aktual të pranuar prej programit kryesor dhe në renditjen e definuar duke i zëvendësuar ata në vend të parametrave të tij formal. Në fund, funksioni, vlerën e llogarituar ia kthen programit kryesor me urdhërin “return” dhe programi kryesor vazhdon aty ku e kishte ndërprerë “punën” për të pritë rezultatin e kërkuar prej nënprogramit.

```

// Programi Funksionil
#include <iostream>
using namespace std;
double Shuma(int x,int y);           //Prototipi i funksionit
int main()
{
    double s;
    int a=7,b=3;
    s=Shuma(a,b);                    /*Thirrja e funksionit
                                     - percjellja e vlerave aktuale*/

    cout << "Shuma s="
         << s
         << "\n";
    return 0;
}

// Nënprogrami Shuma
double Shuma(int x,int y)           //Deklarimi i funksionit
{
    double r;                        //variablat e brendshme te funksionit
    r=x+y;                           //urdherat e funksionit
    return r;                        //kthimi i rezultatit të funksionit
}
  
```

Pra, kur të thirret funksioni shuma, me urdhërin: **s=Shuma(a,b)** ; , atëherë programi kryesor e përcjellë ekzekutimin në funksionin e thirrur, duke ia përcjellë vlerat aktuale a=7 dhe

$b=3$ , për parametrat formal  $x$  dhe  $y$ . Në brendi të funksionit, llogaritet variabla e shumës  $r = x+y$ , e cila në rasiin aktual llogaritet  $r=a+b$ , sepse në hyrjen  $x$  i ka ardhë vlera  $a$ , kurse në hyrjen  $y$  i ka ardhë vlera  $b$ . Këtë variabël (shumën “ $r$ ”) funksioni ia kthen programit përmes urdhërit: **return**  $r$ ; . Kështu, del se urdhëri i programit **s=Shuma(a,b)** ; në fakt është: **s=r** ; , ku  $r$  - është rezultati i kthyer prej funksionit, me urdhërin: **return**  $r$  ; .

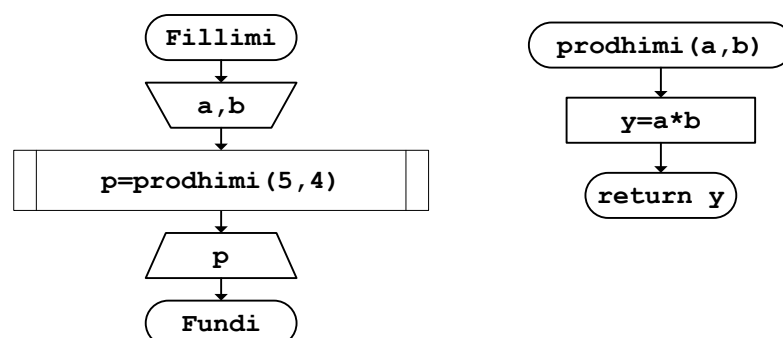
## Funksionet void

Funksionet të cilat nuk kthejnë rezultat fare, quhen funksione **void** (angl. boshe, të zbrazëta, shterpe) dhe definojnë si funksione të tipit **void**. Urdhëri i tyre për kthim të rezultateve shkruhet vetëm **return** ; .

## Inline funksionet

“Trupi” i funksionit zakonisht paraqitet në fund të programit. Nëse komplet funksioni zhvillohet në fillim të programit, në vijë (ang. In line) të rrjedhës së programit, atëherë quhet “Inline function”.

```
// Programi funksioni_inline
#include <iostream>
using namespace std;
inline double prodhimi(int a,int b)
{
    double y;
    y=a*b;
    return y;
}
int main()
{
    double p;
    p=prodhimi(5,4);
    cout << "Prodhimi është p="
        << p
        << "\n";
    return 0;
}
```



## Shembuj funksionesh

Funksionet mund t'i krijojmë për të gjitha rastet e llogaritjeve të cilat kryhen shpeshherë. Për të dizajnuar, funksionin duhet të mendojmë për atë se sa variabla janë të domosdoshme për të mundësuar llogaritjen e rezultatit dhe ato i deklarojmë si parametra të funksionit.

P.sh, funksioni për llogaritjen e shumës së anëtarëve të vargut (serisë së numrave):

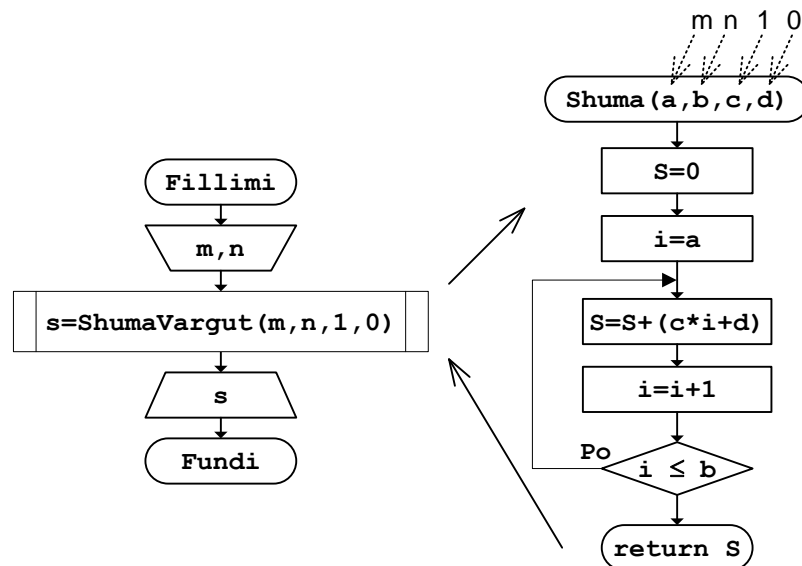
Forma universale e shumës se serisë është:

$$S = \sum_{i=a}^b (c * i + d)$$

Atëherë, ne do të krijojmë një funksion të përgjithshur i cili do të mund të llogarisë shumën për të gjitha format e mundshme të serive të tilla.

P.sh, shumën e anëtarëve të njëpasnjeshëm, prej m gjerë në n do të ishte:

$$S = \sum_{i=m}^n i \quad (\text{Pra, shihet se do të kemi: } a=m, b=n, c=1, d=0)$$



```

// Programi Funksion1
#include <iostream>
using namespace std;

double ShumaVargut(int a, int b, int c, int d);
int main()
{
    int m, n;
    double Shuma;
    m=0;
    n=5;
    Shuma=ShumaVargut(m, n, 1, 0); //Funks. per shumen e vargut

    cout << "Shuma S=: " << Shuma;
    cout << "\n";
}
  
```

```

    return 0;
}
double ShumaVargut(int a,int b, int c, int d)
{
    int i,j;
    double S;
    S=0;
    for (i=a;i<=b;i++)
    {
        S=S+(c*i+d);
    }
    return S;
}

```

\*\*\*\*\*

Te llogaritet:

$$S = \sum_{i=2}^n (3*i+2) \quad (\text{Tani, shihet se do te kemi: } a=2, b=n, c=3, d=2)$$

```

// Programi Funksion2
#include <iostream>
using namespace std;

double ShumaVargut(int a, int b, int c, int d);
int main()
{
    int n;
    double Shuma;
    cout<<"Jepe vleren per n:";
    cin>>n;
    Shuma=ShumaVargut(2,n,3,2);           //Funks. per shumen e vargut

    cout << "Shuma S=: " << Shuma;
    cout << "\n";

    return 0;
}
double ShumaVargut(int a,int b, int c, int d)
{
    int i,j;
    double S;
    S=0;
    for (i=a;i<=b;i++)
    {
        S=S+(c*i+d);
    }
    return S;
}

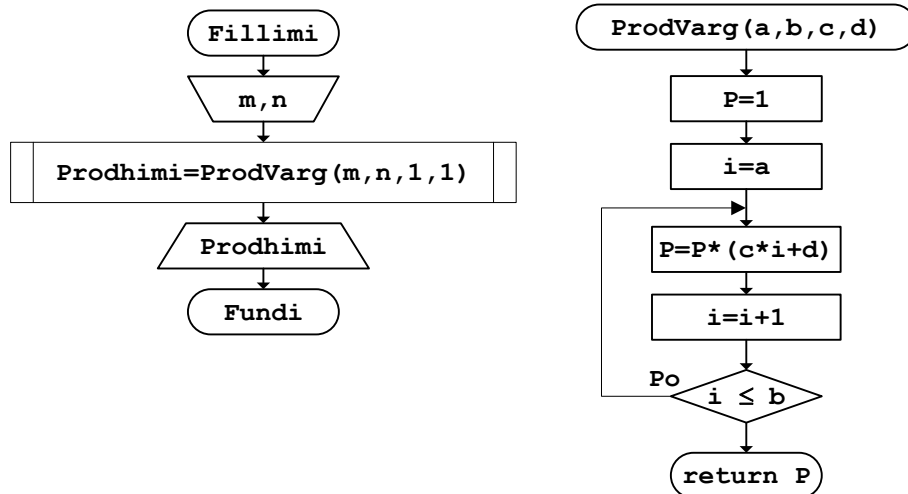
```

Edhe për rastin e prodhimit, forma universale e prodhimit të serisë së numrave është:

$$P = \prod_{i=a}^b (c * i + d)$$

Shembull:

$$P = \prod_{i=m}^n (i+1) \quad (\text{Në këtë rast do të kemi: } a=m, b=n, c=1, d=1)$$



```

// Programi Funksion3
#include <iostream>
using namespace std;

double ProdVarg(int a, int b, int c, int d);
int main()
{
    int m,n;
    double Prodhimi;
    m=0;
    n=5;
    Prodhimi=ProdVarg(m,n,1,1);           //Funks. per prod. e vargut

    cout << "Prodhimi P=: " << Prodhimi;
    cout << "\n";

    return 0;
}

double ProdVarg(int a,int b, int c, int d)
{
    int i,j;
    double P;
    P=1;
    for (i=a;i<=b;i++)
    {
        P=P*(c*i+d);
    }
    return P;
}
  
```

Shembull:

$$P = \prod_{i=m}^n (2i+4) \quad (\text{Në kete rast do te kemi: } a=m, b=n, c=2, d=4)$$

```
// Programi Funksion3
#include <iostream>
using namespace std;

double ProdVarg(int a, int b, int c, int d);
int main()
{
    int m,n;
    double Prodhimi;
    cout<<"Jepni vlerat per kufijte m dhe n:";
    cin>>m>>n;

    Prodhimi=ProdVarg(m,n,2,4);           //Funks. per prod. e vargut

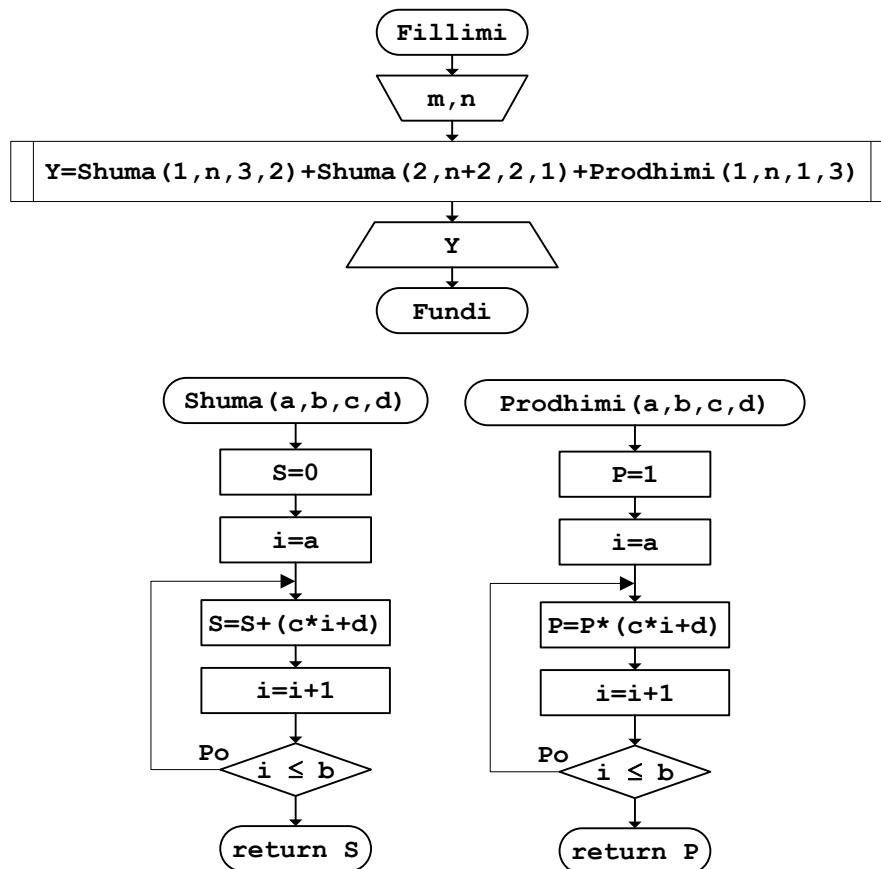
    cout << "Prodhimi P=: " << Prodhimi;
    cout << "\n";

    return 0;
}

double ProdVarg(int a,int b, int c, int d)
{
    int i,j;
    double P;
    P=1;
    for (i=a;i<=b;i++)
    {
        P=P*(c*i+d);
    }
    return P;
}
```

Të llogaritet:

$$Y = 3 \sum_{i=1}^n (3i+2) + \sum_{i=2}^{n+2} (2*i+1) + 4 \prod_{i=1}^n (i+3)$$



```

// Programi FunkSION5
#include <iostream>
using namespace std;

double Shuma(int a, int b, int c, int d);
double Prodхими(int a, int b, int c, int d);
int main()
{
    int n;
    double Y;
    cout<<"Jepe vleren per n:";
    cin>>n;

    Y=3*Shuma(1,n,3,2)+Shuma(2,n+2,2,1)+Prodхими(1,n,1,3);

    cout << "Shuma Y= " << Y;
    cout << "\n";

    return 0;
}

double Shuma(int a,int b, int c, int d)
{
    int i;
  
```

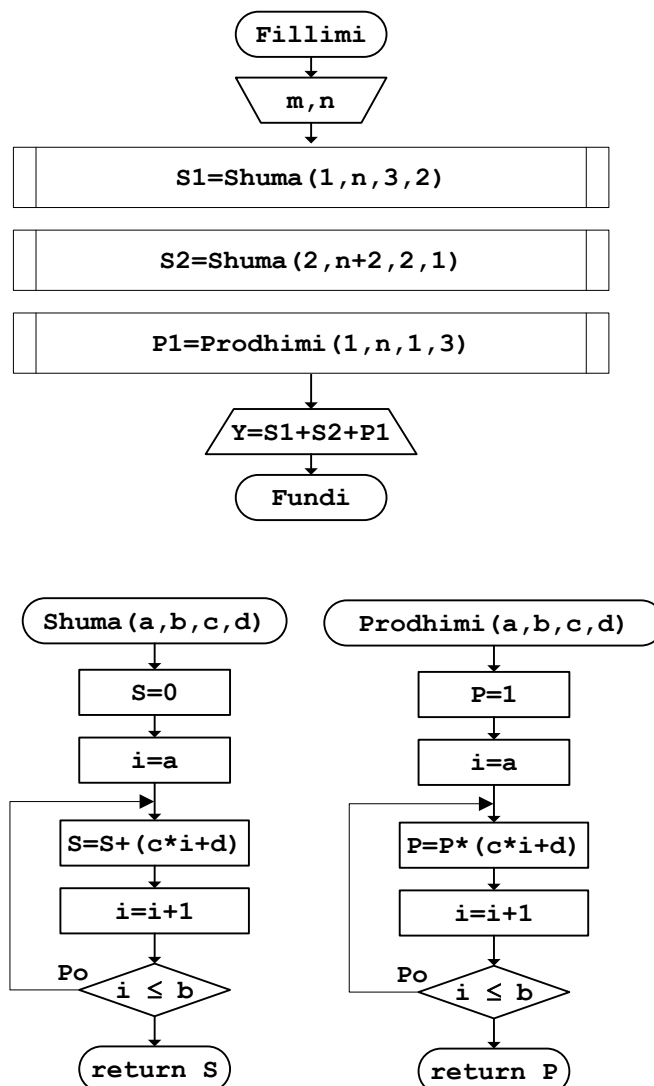


```

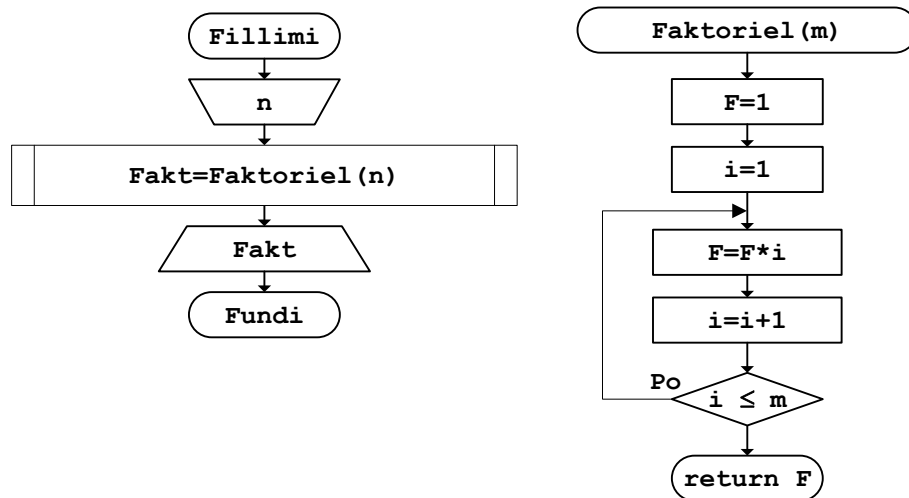
double S;
S=0;
for (i=a;i<=b;i++)
{
    S=S+(c*i+d);
}
return S;
}
double Prodhimi(int a,int b, int c, int d)
{
    int i;
    double P;
    P=1;
    for (i=a;i<=b;i++)
    {
        P=P*(c*i+d);
    }
    return P;
}

```

Për paraqitje më të lehtë, programin kryesor mund ta bëjmë edhe si vijon:



## Llogaritja e faktorelit



```

// Programi Funksion4
#include <iostream>
using namespace std;

double Faktoriel(int m);
int main()
{
    int n;
    double Fakt;
    n=5;

    Fakt=Faktoriel(n);           //Funks. per Faktoriel

    cout << "Faktorieli F = "<<n<<"! = " << Fakt;
    cout << "\n";

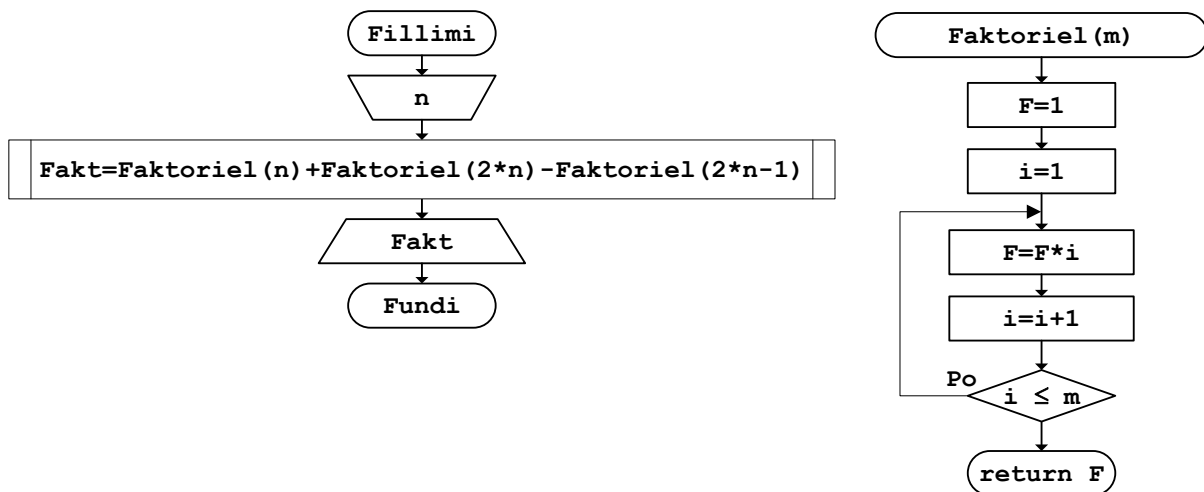
    return 0;
}

/* Funksioni per llogaritje te faktorielit */
double Faktoriel(int m)        {
    int i;
    double F;
    F=1;
    for (i=1; i<=m; i++)
    {
        F=F*i;
    }
    return F;
}
  
```

Pra, shihet se funksionit për Faktoriel, i nevojitet vetëm një variabël hyrëse (kufiri për llogaritje të faktorielit).

Shembull:

Të llogaritet:  $Fakt = n! + (2n)! - (2n-1)!$



```

// Programi FunkSION4
// Fakt = n! + (2n)! - (2n-1)!
#include <iostream>
using namespace std;

double Faktoriel(int m);
int main()
{
    int n;
    double Fakt;
    n=2;

    Fakt=Faktoriel(n)+Faktoriel(2*n)-Faktoriel(2*n-1);

    cout << "Faktorieli F = " << Fakt;
    cout << "\n";

    return 0;
}

/* Funksioni per llogaritje te faktorielit */
double Faktoriel(int m)
{
    int i;
    double F;
    F=1;
    for (i=1;i<=m;i++)
    {
        F=F*i;
    }
    return F;
}
  
```

## Rekurzioni

Rekurzioni, paraqet thirrjen e vet funksionit brenda funksionit, pra thirrjen e vet-vehtes...

```
// Programi Faktoriel-Rekurzioni
#include <iostream>
using namespace std;

double Faktoriel(int m);
int main()
{
    int n;
    double Fakt;
    n=5;

    Fakt=Faktoriel(n);           //Funks. per Faktoriel

    cout << "Faktorieli F = "<<n<<"! = " << Fakt;
    cout << "\n";

    return 0;
}

/* Funksioni per llogaritje te faktorielit - rekurzioni */
double Faktoriel(int m)        {
    int i;
    double F;
    F=1;
    for (i=1;i<=m;i++)
    {
        F=m*Faktoriel(m-1);    // ketu ndodhe rekurzioni
    }
    return F;
}
```

ose, per per vleren hyrese n, te dhene nga shfrytezuesi:

```
//Llogaritja e faktorielit, permes rekurzionit
#include <iostream>
using namespace std;
int main()
{
    int n;
    int fakt;

    cout << "Jepni vleren per n: ";
    cin >> n;

    fakt = Faktoriel (n);

    cout << n << "! = " << fakt << endl;

    return 0;
}
```

```

/* Funkzioni per llogaritje te faktorielit - rekurzioni */
int Faktoriel (int n)
{
    //(funksioni pa variabla te brendshme plotesuese, si F, i, etj.)
    if (n > 1) {          //kontrollojme vleren hyrese n
        return n * Faktoriel (n - 1);
    }
    else {
        return 1;
    }
}

```

## Funksionet dhe Vektorët

```

// Programi FunkzioniVektoril
#include <iostream>
using namespace std;
void FormoVektorin(int X[],int n);
int main()
{
    const int m=5;
    int i,A[m];
    FormoVektorin(A,m);
    cout << "Vektori i formuar\n";
    for (i=0;i<m;i++)
        cout << A[i]
                << " ";
    cout << "\n";
    return 0;
}
void FormoVektorin(int X[],int n)
{
    int i;
    for (i=0;i<n;i++)
        X[i]=2*i;
    return;
}

```

```

// Programi FunkzionVektor2
#include <iostream>
using namespace std;
int ShumaAntVektorit(int X[],int n);
int main()
{
    const int m=5;
    int Shuma;
    int A[m]={1,2,3,4,5};
    Shuma=ShumaAntVektorit(A,m);
    cout << "Shuma e anetareve te vektorit: "
            << Shuma
            << "\n";
    return 0;
}

```

```
int ShumaAntVektorit(int X[],int n)
{
    int i,s;
    s=0;
    for (i=0;i<n;i++)
        s=s+X[i];
    return s;
}
```

```
// Programi FunksionVektor3 - Sortimi dhe shtypja me funksion
#include <iostream>
using namespace std;
void SortoVektorin(int X[],int n);
void ShtypeVektorin(int X[],int n);
int main()
{
    const int m=5;
    int Shuma;
    int A[m]={3,2,1,4,5};

    SortoVektorin(A,m);
    ShtypeVektorin(A,m);

    return 0;
}

void SortoVektorin(int X[],int n)
{
    int i,j,b;
    for (i=0;i<n-1;i++)
        for (j=i+1;j<n;j++)
        {
            if (X[i]>=X[j])
            {
                b=X[i];
                X[i]=X[j];
                X[j]=b;
            }
        }
    return;
}

void ShtypeVektorin(int X[],int n) // funksion i tipit void
{
    int i;
    cout << "X=[ ";
    for (i=0;i<n;i++)
        cout << X[i] //shtypja
        << " ";
    cout << "]"
        << "\n";
    return; //nuk kthen rezultat, sepse punen (shtypjen) e kryen
           // ne brendi te funksionit, prandaj eshte void
}
```

Pra, funksioni `void ShtypeVektorin(int X[], int n)`, nuk kthen rezultat, sepse veprimet i kryen ne brendi te funksionit, keshtu qe s'ka nevojte te ktheje rezultat fare. Programi kryesor, i cili e e ka thirrur funksionin per ta kryer punen e caktuar, ne kete rast s'ka nevojte per rezultat kthyes prej funksionit, por vetem kerkon qe funksioni t'a kryej nje pune te caktuar (ne kete rast shtypjen e rezultatit (vektorit) ne dalje).

```
// Programi FunksionVektor4 - Minimumi dhe Maximumi me funksion
#include <iostream>
using namespace std;
void ShtypeVektorin(int X[],int n);
int AntMaxVektorit(int X[],int n);
int AntMinVektorit(int X[],int n);

int main()
{
    const int m=5;
    int Amin, Amax;

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

    ShtypeVektorin(A,m);

    Amin=AntMinVektorit(A,m);
    Amax=AntMaxVektorit(A,m);

    cout<< "\n Antari min. i vekt: " <<Amin << endl;
    cout<< "\n Antari max. i vekt: " <<Amax << endl;

    return 0;
}

void ShtypeVektorin(int X[],int n)
{
    int i;
    cout << "X=[ ";
    for (i=0;i<n;i++)
        cout << X[i]
            << " ";
    cout << "]"
        << "\n";
    return;
}

int AntMaxVektorit(int X[],int n)
{
    int i,j,Xmax;
    Xmax=X[0];
    for (i=1;i<n;i++)
    {
        if (X[i]>Xmax)
            Xmax=X[i];
    }
    return Xmax;
}
```

```

int AntMinVektorit(int X[],int n)
{
    int i,j,Xmin;

    Xmin=X[0];
    for (i=1;i<n;i++)
    {
        if (X[i]<Xmin)
            Xmin=X[i];
    }
    return Xmin;
}

```

## Funksionet dhe matricat

```

// Programi FunksionMatrical
#include <iostream>
using namespace std;
const int n=3;
void FormoMatricen(int A[][n],int m);
int main()
{
    const int m=3;
    int i,j,A[m][n];

    FormoMatricen(A,m); //Thirre funksionin per krijim te matrices

    cout << "Matrica A: \n";
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
            cout << A[i][j] << " ";
        cout << "\n";
    }
    return 0;
}
void FormoMatricen(int A[][n],int m)
{
    int i,j;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
            A[i][j]=i+j;
    }
    return;
}

```

```

// Programi FunksionMatrica2
#include <iostream>
using namespace std;
const int n=3;
double ShumaAntMatrices(int X[][n],int m);

```



```

int main()
{
    const int m=2, n=3;
    double Shuma;
    int A[m][n]={          {1,2,3},
                           {4,5,6},
                           };
    Shuma=ShumaAntMatrices(A,m);
    cout << "Shuma e anetareve te matrices S="
         << Shuma
         << "\n";
    return 0;
}
double ShumaAntMatrices(int X[][n],int m)
{
    int i,j;
    double s;
    s=0;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
            s=s+X[i][j];
    }
    return s;
}

```

```

// Programi FunksionMatrica2
#include <iostream>
using namespace std;
const int n=3;
double ShumaAntPozMat(int X[][n],int m);
int main()
{
    const int n=3;
    int i,j,m;
    int X[10][n];
    cout<< "Sa rreshta do t'i kete matrica: ";
    cin >> m;

    double Shuma;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
        {
            cout << "X[" << i+1 << "][" << j+1 << "]=";
            cin >> X[i][j];
        }
    }

    Shuma=ShumaAntPozMat(X,m);
    cout << "Shuma e anetareve pozitiv te matrices S="
         << Shuma
         << "\n";
    return 0;
}

```

```
double ShumaAntPozMat(int X[][n], int m)
{
    int i, j;
    double s;
    s=0;
    for (i=0; i<m; i++)
    {
        for (j=0; j<n; j++)
            if (X[i][j]>0)
                s=s+X[i][j];
    }
    return s;
}
```

## Përfshirja e fajllave me direktivën #include

Zakonisht direktivën `#include`, e kemi përdorur për t'i thirrur libraritë e gatshme të C++-it. Kllapat e drejta me simbolet "< dhe >" që përdoret tek : `#include <iostream> using namespace std;`, i tregojnë C++-it se në program përfshihet edhe ndonjë "header file" nga folderi i zakonshëm i C++, aty ku ruhen "header fajllat". Nëse dëshirojmë, që në program ta përfshijmë një fajll tjetër, që është i ruajtur brenda folderit të njëjtë, ku është i ruajtur edhe programi aktual, atëherë përdoret forma:

`#include "Emri_i_fajllit.h"`. Pra, emri shkruhet brenda thojzave të zakonshme. Header fajlli, krijohet duke shkruar përmes menysë: File – New, dhe zgjedhet C/C++ Header File. Emërtohet me emrin me prapashtesë .h dhe pastaj shkruhet dhe ruhet si fajllat e zakonshëm burimor të C++.

P.sh., shpesh here kemi nevojë që të shtypim adresën tonë për kontakt:

```
Avni Rexhepi
Fakulteti Elektroteknik
Prishtinë
Tel: 044 - 174 374
```

Tërë këtë tekst mund ta shtypim me urdhërat vijues, të ruajtur në një Header File (Header fajll) të C++, "Adresa.h", si:

```
//Fajlli "Adresa.h"
cout<<"Avni Rexhepi \n";
cout<<"Fakulteti Elektroteknik \n";
cout<<"Prishtin\x89 \n";
cout<<"Tel: 044 - 174 374 \n";
```

Pastaj, në ndonjë program të C++, mund ta thërrasim për ekzekutim fajllin "Adresa.h", sa herë që të na duhet shtypja e adresës:

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

int main()
{
    cout<< "Jungjatjeta \n\n";
    #include "Adresa.h" //Thirrja e heder fajllit
    return 0;
}
```

Pra, kjo do të jep efektin e njëjtë, sikur në rastin:

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

int main()
{
    cout<< "Jungjatjeta \n\n";

    cout<<"Avni Rexhepi \n";
    cout<<"Fakulteti Elektroteknik \n";
    cout<<"Prishtin\x89 \n";
    cout<<"Tel: 044 - 174 374 \n";

    return 0;
}
```

## Thirrja e funksionit, nga fajlli tjetër

Edhe funksionet mund të shkruhen dhe të ruhen si fajlla të veçantë, e pastaj të përdoren në shumë programe të ndryshme, sipas nevojës. Në rast të tillë, përfshirja e funksionit brenda programit, bëhet me direktivën preprocesorike `#include`.

P.sh., në folderin: “C:\Shembuj” kemi fajllat: “FunksioniMeInclude.cpp” dhe “FormoVektorin.h”. Atëherë në programin kryesor shkruajmë:

```
// Program-Funksioni i thirrur me #include
#include <iostream>
using namespace std;
#include "FormoVektorin.h" //përfshije fajllin "FormoVektorin.h".

int main()
{
    const int m=5;
    int i,A[m];
    FormoVektorin(A,m);
    cout << "Vektori i formuar\n";
    for (i=0;i<m;i++)
        cout << A[i]
            << " ";
    cout << "\n";
    return 0;
}
```

Kurse “Header fajllin” – FormoVektorin.h, e kemi:

```
//Nenprogrami ne fajll te vegante
//Funksioni FormoVektorin(int X[],int n)

void FormoVektorin(int X[],int n)
{
    int i;
    for (i=0;i<n;i++)
        X[i]=2*i;
    return;
}
```

Nëse fajlli i funksionit që thirret në programin kryesor nuk ndodhet në të njëjtin folder me programin kryesor, atëherë kur të përfshihet në program me direktivën `#include`, duhet të shkruhet shtegu i plotë i fajllit, si p.sh., nëse funksioni ndodhet në shtegun: “C:\Shembuj\Funksionet\FormoVektorin.h”, atëherë do të duhej të shkruhet:

```
#include "C:\Shembuj\Funksionet\FormoVektorin.h"
//përfshije fajllin "FormoVektorin.h".
```

Emri i fajllit mund të jepet sipas dëshirës, nuk është e thënë të ketë emrin njësoj si vet funksioni.



Shembull:

**Fajlli: "Include2.cpp" - programi kryesor**

```
//Llogaritja e faktorielit, permes rekurzionit
#include <iostream>
using namespace std;
#include "Faktorieli.h"
int main()
{
    int n;
    int fakt;

    cout << "Jepni vleren per n: ";
    cin >> n;

    fakt = Faktoriel (n);

    cout << n << "! = " << fakt << endl;

    return 0;
}
```

**Fajlli: "Faktorieli.h" - Heder fajlli**

```
/* Funksioni per llogaritje te faktorielit */
int Faktoriel (int n)
{
    //(funksioni pa variabla te brendshme plotesuese, si F, i, etj.)
    if (n > 1) {                //kontrollojme vleren hyrese n
        return n * Faktoriel (n - 1);
    }
    else {
        return 1;
    }
}
```

Mund të përfshihen edhe nga disa heder fajlla në një program. Thirrja e heder fajllit mund të behet edhe në rrjedhë e sipër të programit.

Shembull:

```
//Fajlli: "Shtypja.cpp" - programi kryesor
//Llogaritja e faktorialit, permes rekurzionit
#include <iostream>
using namespace std;
#include "Shtype1.h" // - Përfshirja e fajllit Shtype1.h

int main()
{
    Shtype(); //Funksioni prej Shtype1.h
    #include "Shtype2.h" //thirrja e Shtype2.h
    return 0;
}
```

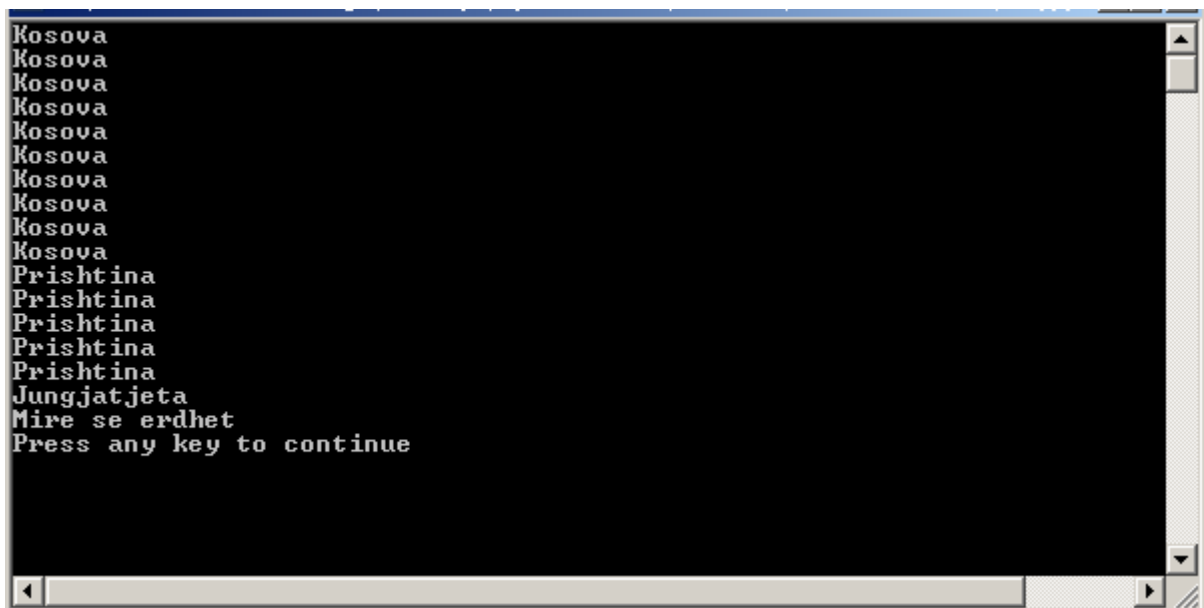
```
//Fajlli: "Shtype1.h"
int i,j;
void Shtype()
{
    for (i=0;i<10;i++)
        cout<<"Kosova\n";

    for (j=0;j<5;j++)
        cout<<"Prishtina\n";

    return;
}
```

```
//Fajlli: "Shtype2.h"
cout << "Jungjatjeta\n";
cout << "Mire se erdhet\n";
```

Rezultati në dalje:



```
Kosova
Kosova
Kosova
Kosova
Kosova
Kosova
Kosova
Kosova
Kosova
Kosova
Prishtina
Prishtina
Prishtina
Prishtina
Prishtina
Jungjatjeta
Mire se erdhet
Press any key to continue
```

## Direktiva #define

Direktiva #define mund të përdoret për definime të ndryshme në programe dhe funksionon si urdhëri për zëvendësim të fjalës me një fjalë tjetër në editorët e teksteve (si. p.sh. Word-i). Format i saj është:

```
#define ARGUMENTI1 argumenti2
```

Fjala ARGUMENTI1 është një fjalë e vetme, pa hapësira. Për emërtim përdoren rregullat e njëjta si për variablat. Mes emrave të dy argumenteve, duhet të ketë së paku një hapësirë. Kudo në program që të shkruhet ARGUMENTI1 është njësoj, sikur të ishte shkruar argumenti2.

Shembull:

```
#include <iostream>
using namespace std;
#define Emri "Avni Rexhepi"
int main()
{
    cout<< Emri << endl;
    return 0;
}
```

Direktiva #define mund të përdoret edhe për definim të funksioneve të ndryshme:

```
#include <iostream>
using namespace std;

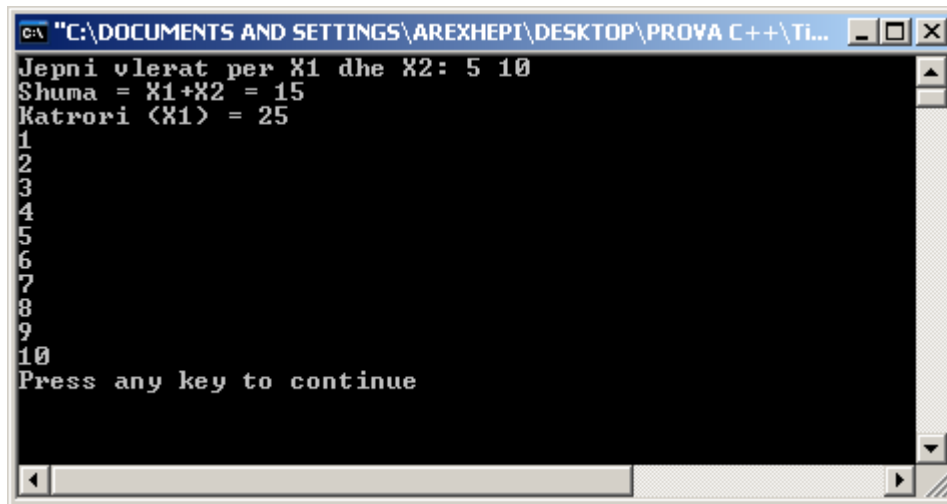
#define MINIMUMI 1
#define MAKSIMUMI 10
#define SHUMA X1+X2
#define Katrori(a) (a*a)

int main()
{
    int i,X1,X2,a;

    cout<<"Jepni vlerat per X1 dhe X2: ";
    cin>>X1>>X2;
    cout<< "Shuma = X1+X2 = "<<SHUMA<<endl;
    cout<< "Katrori (X1) = " << Katrori(X1) <<endl;

    for (i=MINIMUMI;i<=MAKSIMUMI;i++)
        cout << i << endl;

    return 0;
}
```



```
C:\ "C:\DOCUMENTS AND SETTINGS\AREXHEPI\DESKTOP\PROVA C++\Ti...
Jepni vlerat per X1 dhe X2: 5 10
Shuma = X1+X2 = 15
Katrori (X1) = 25
1
2
3
4
5
6
7
8
9
10
Press any key to continue
```

Me direktivën `#define`, mund të bëhet edhe definimi i urdhërave të C++, me fjalë të gjuhës shqipe, si p.sh.: për `cout` – `shtyp`, për `cin` - `lexo`, për `endl` - `rreshtiri`;

```
#include <iostream>
using namespace std;

#define shtyp cout
#define lexo cin
#define rreshtiri endl

int main()
{
    int i,a,b;

    shtyp<<"Jepni vlerat per a dhe b: ";
    lexo>>a>>b;

    shtyp<< "Shuma = a+b = "<<a+b<<rreshtiri;

    for (i=1;i<=10;i++)
        shtyp << i << rreshtiri;

    return 0;
}
```



## Shtojca A - Probleme praktike

### Numrat e rastit në C++

**Funksionet `srand()` and `rand()`** - Libraria standarde e C++ ofron një “vegël” për gjenerimin e numrave të rastit duke përdorur funksionin **`rand()`** (angl. random – i rastit) . Natyrisht, këta numra nuk janë me të vërtetë të rastit, por janë pseudo-numra të rastit. Ata gjenerohen në një varg i cili jep iluzionin e të qenit i rastit, sepse, megjithatë, ky varg i numrave do të përsëritet. Versioni i **`rand()`** që vije me kompajlerin e C++-it është mjaft i thjeshtë dhe nuk është prej më të mirëve (nuk përdoret për ndonjë qëllim shkencor) mirëpo mund të përdoret për programe të thjeshta dhe për lojra.

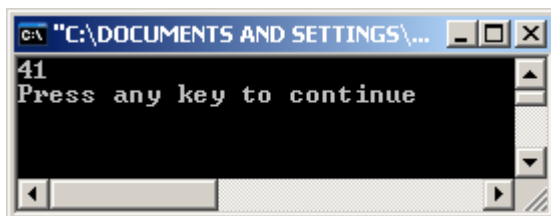
#### Si gjenerohet numri i rastit në C++?

Për të përdorur funksionin `rand()` duhet të përfshijmë në program heder fajllin përkatës `<cstdlib>`. Funksioni `rand()` thjeshtë e kthen pseudo-numrin e plotë të rastit në rangun prej 0 deri në `RAND_MAX` dhe përdoret si vijon:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    int NumriiRastit = rand();
    cout << NumriiRastit << endl;
}
```

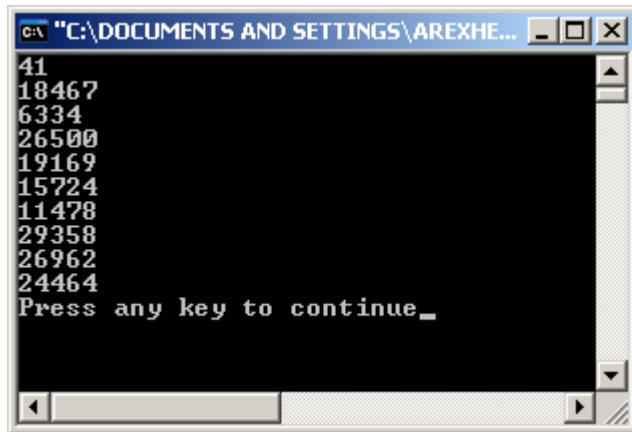


Për të gjeneruar më shumë numra të rastit, përsëritet thirrja e funksionit `rand()`.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    int NumriiRastit;
    for(int i=0; i<10; i++){
        NumriiRastit = rand();
        cout << NumriiRastit << endl;
    }
}
```

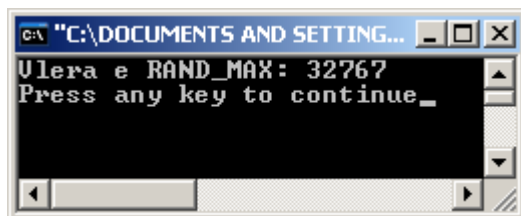


Vlera RAND\_MAX nuk është e specifikuar me standardin e C++, por mund të supozojmë se RAND\_MAX është së paku: 32767. Për të gjetur vlerën e saktë për RAND\_MAX në kompjuterin tuaj, ekzekutoni kodin vijues:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    cout << "Vlera e RAND_MAX: " << RAND_MAX << endl;
}
```



### Pse nevojitet srand()?

Funksioni srand() përdoret për të inicializuar gjeneratorin e pseudo numrave të rastit. Kjo gjë cakton pikën fillestare për vargun e numrave të gjeneruar. Gjeneratorin e numrave të rastit duhet të inicializojmë para se të thërrasim funksionin rand() për herë të parë.

```
#include <cstdlib>
#include <iostream>

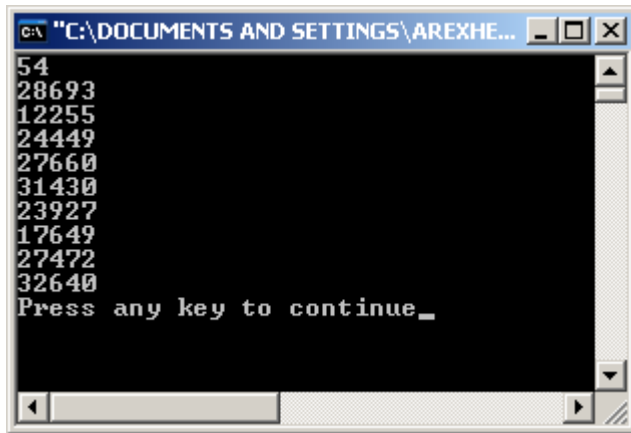
using namespace std;

int main()
{
    int fillimi = 5;
    srand(fillimi);
    int NumriiRastit;
    for(int i=0; i<10; i++)
    {
```

```

    NumriiRastit = rand();
    cout << NumriiRastit << endl;
}
}

```



Nëse inicializojmë gjeneratorin gjithmonë me të njëjtën vlerë të fillimit, secilën herë që të ekzekutohet programi do të fitohen numrat e njëjtë. Kjo normalisht është joadekuate për shumicën e rasteve, pasi që zakonisht kërkohet varg i numrave të rastësishëm (përveq nëse jemi duke kërkuar gjetjen e gabimeve në program (de-bugimi)). Zgjidhja qëndron në përdorimin e vlerës së ndryshme fillestare për secilën herë. Mënyrë e zakonshme është duke marrur si vlerë fillestare kohën momentale të kompjuterit. Pasi që me gjasë secilën herë koha është e ndryshme, do të gjenerohet varg i numrave të ndryshëm.

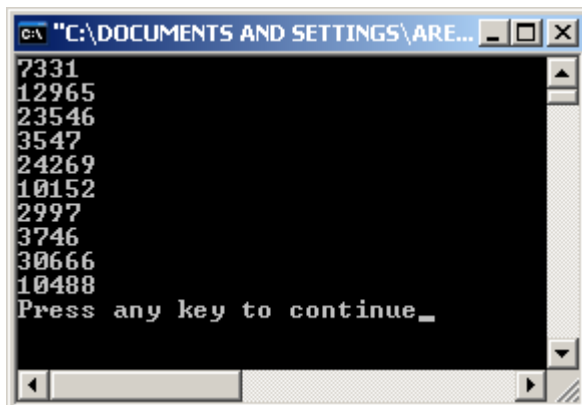
```

#include <cstdlib>
#include <ctime>
#include <iostream>

using namespace std;

int main()
{
    srand(static_cast<unsigned>(time(0)));
    int NumriiRastit;
    for(int i=0; i<10; i++){
        NumriiRastit = rand();
        cout << NumriiRastit << endl;
    }
}

```



Pra, kemi thirrur edhe header-in <ctime> për funksionin time(). Vlera e kthyer nga funksioni time() llogaritet prej numrit pa shenjë “unsigned” duke përdorur operatorin “static\_cast”.

## Si gjenerohen numrat në një brez (rang) të caktuar

Ka disa mënyra për gjenerimin e numrave në një rang të caktuar. Një mënyrë e zakonshme, që haset shpesh është:

```
NumriiRastit = (rand()%10);
```

Kjo do të gjeneroj numra në rangun 0 deri në 9 (% - Pjestimi me modul, ku si rezultat është mbetja nga pjestimi i plote me një numer, në këtë rast pjestimi i plote me 10). Për të filluar prej ndonjë baze tjetër të ndryshme prej 0 (zeros), veprohet si vijon:

```
NumriiRastit = 20+(rand()%10);
```

Kjo do të mundësojë gjenerimin e numrave prej 20 gjerë në 29.

Mirëpo metoda e mësipërme e krijimit të pseudo numrave të rastit është mjaft e varfër sepse i përdorë vetëm shifrat e fundit (me vlerë të vogël, njëshet dhe dhjetëshet) të numrit të kthyer nga rand() dhe këto mund të jenë më pak të rastit për disa zbatime të gjeneratorit të pseudo numrave të rastit.

Si mënyrë më e mirë preferohet:

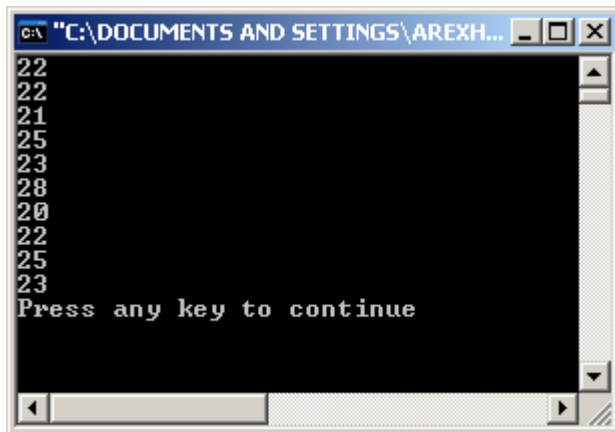
```
NumriiRastit = 20 + int(10.0 * rand()/(RAND_MAX+1.0));
```

Nëse këto i zbatojmë në programin e mësipërm, do të kemi:

```
#include <cstdlib>
#include <ctime>
#include <iostream>

using namespace std;

int main()
{
    srand(static_cast<unsigned>(time(0)));
    int NumriiRastit;
    for(int i=0; i<10; i++){
        NumriiRastit = 20 + int(10.0 * rand()/(RAND_MAX+1.0));
        cout << NumriiRastit << endl;
    }
}
```



Kjo mund të bëhet më e dobishme duke e inkorporuar në funksion, si në vijim:

```
int rangu_i_rastit(int numri_i_vogel, int numri_i_madh)
{
    if(numri_i_vogel > numri_i_madh){
        swap(numri_i_vogel, numri_i_madh);
    }

    int rangu = numri_i_madh - numri_i_vogel + 1;
    return numri_i_vogel+int(rangu*rand()/(RAND_MAX+1.0));
}
```

Pra, funksionit **rangu\_i\_rastit()** i përcillen dy vlera që prezantojnë vlerën më të vogël dhe më të madhe të rangut të dëshiruar. Funksioni verifikon a e kemi renditjen e duhur të vlerës së vogël dhe asaj të madhe. Pastaj llogaritet rangi i kërkuar i numrave të rastit. Në fund, i aplikojmë këta numra për thirrjen e funksionit **rand()** dhe kthimin e vlerave të rastit. Funksioni **swap()** (angl. Swap-shkëmbe, ndërro) kërkon header-in <algorithm>.

Programi i vogël vijues demonstroi përdorimin e funksionit rangu\_i\_rastit(). Inicializojmë gjeneratorin e pseudo numrave të rastit duke përdorur kohën aktuale.

```
#include <algorithm>
#include <cstdlib>
#include <ctime>
#include <iomanip>
#include <iostream>

using namespace std;

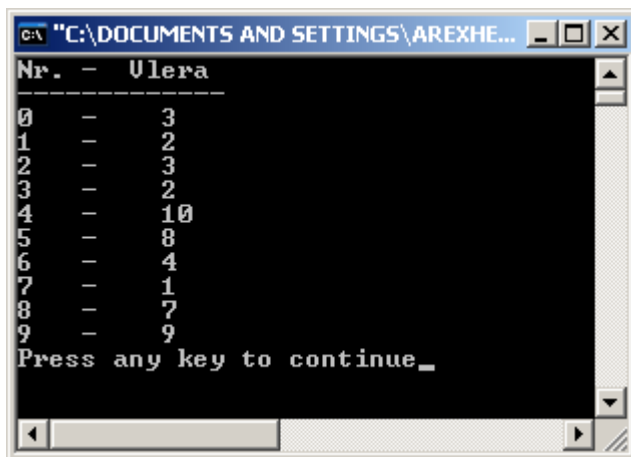
int rangu_i_rastit(int numri_i_vogel, int numri_i_madh)
{
    if(numri_i_vogel > numri_i_madh){
        swap(numri_i_vogel, numri_i_madh);
    }
    int rangu = numri_i_madh - numri_i_vogel + 1;
    return numri_i_vogel + int(rangu * rand()/(RAND_MAX + 1.0));
}

int main()
{
    srand(static_cast<unsigned>(time(0)));
```

```

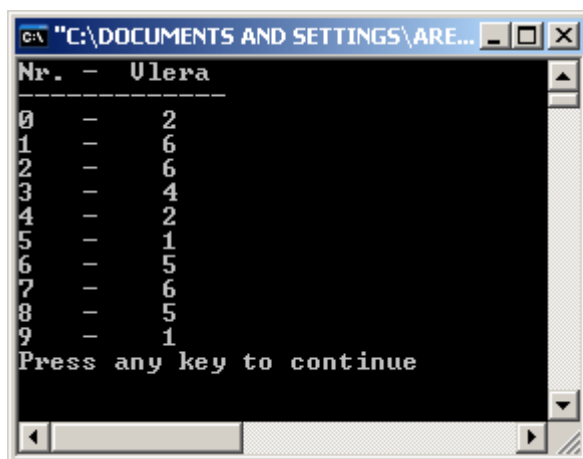
int numri_i_rastit;
cout<<"Nr. -  Vlera \n";
cout<<"-----\n";
for(int i=0; i < 10; i++)
{
    numri_i_rastit = rangu_i_rastit(1,10);
    cout << i <<"    -    " << numri_i_rastit << endl;
}
}

```



Për të përdorur funksionin rangu\_i\_rastit në programin për simulimin e zarit, do të thërrasim:

```
rangu_i_rastit(1,6);
```



## Llogaritjet kohore

### Time

**Time** është tip i variablës i aftë për prezentim të kohës dhe përkrahë operacionet aritmetike. Ky tip i variablës kthehet nga ana e funksionit “time( )” dhe përdoret si parametër nga disa funksione të tjera nga heder fajlli: <ctime>.

Kjo është një vlerë e cila në përgjithësi tregon numrin e sekondave të kaluar që nga ora: 00:00 e datës 1 Janar 1970 UTC (Universal Time). Kjo është marrë për arsye historike, pasi që i korrespondon vulës kohore të UNIX-it (Unix timestamp), por është e përhapur në libraritë e C-së, për të gjitha platformat.

Në header fajllin <ctime> kemi funksionet: clock, time, difftime etj.

### Programi i orës

```
Clock_t clock ( void );
```

Funksioni “clock( )” kthen numrin e të rrahurave të orës (tik-ave) të kaluara që prej fillimit të ekzekutimit të programit. Në rast gabimit, funksioni kthen vlerën -1.

Makro konstanta “CLOCKS\_PER\_SEC” ose “CLK\_TCK” specifikon relacionin ndërmjet tikave dhe sekondit (të rrahura brenda sekondit).

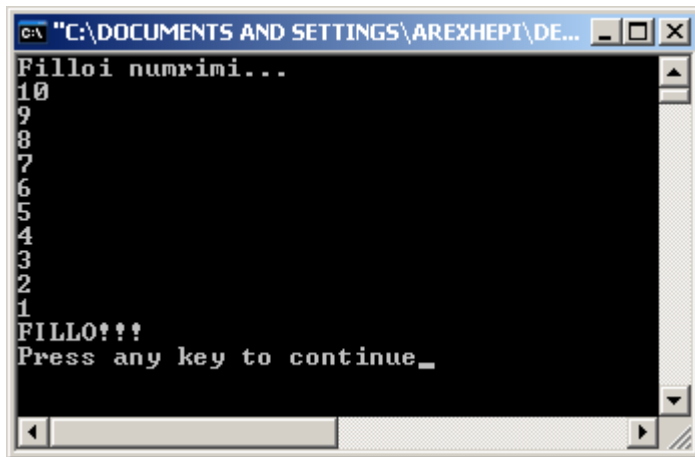
Momenti inicial i references, i përdorur prej ores (clock) si fillim i ekzekutimit të programit mund të ndryshojë varësisht prej kompjuterit. Për të llogaritur kohën aktuale të procesimit të programit, vlera e kthyer e orës (clock) duhet të krahasohet me vlerën e kthyer prej ndonjë thirrjeje initiale (parprake) të clock-ut.

Clock\_t është tip i definuar në <ctime> (dhe <time.h>, në “C”) si një tip me mundësi prezentimi të tikave të orës dhe përkrahë operacionet aritmetike (në përgjithësi, është e tipit long integer).

```
/* Shembull i clock: numrimi ne zbritje - countdown */
#include <iostream>
#include <ctime>
using namespace std;

void prit ( int sekonda ) //funksioni prit - per vonese kohore
{
    clock_t fundi; //variabla e tipit clock_t (nga libraria time)
    fundi = clock () + sekonda * CLOCKS_PER_SEC ;
    while (clock() < fundi) {} //perserite gjersa clock() < fundi
}

int main ()
{
    int n;
    cout<<"Filloi numrimi...\n";
    for (n=10; n>0; n--)
    {
        cout<<n<<endl;
        prit (1); //thirrja e funksionit prit, per vonese 1-sekond
    }
    cout<<"FILLO!!!\n";
    return 0;
}
```



Funksioni **time()** mund të përdoret edhe vetëm sa për të treguar kohën aktuale të kompjuterit.

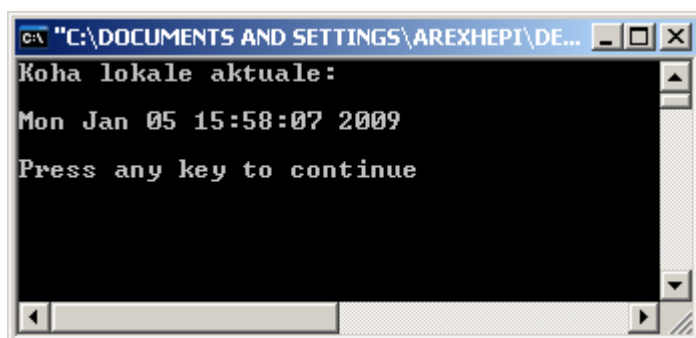
Shembull:

```
/* Shembull - time */
#include <iostream>
#include <ctime>
using namespace std;

int main ()
{
    time_t koha;    //tipi time_t, është i definuar ne <ctime>

    time ( &koha );
    cout<< "Koha lokale aktuale:\n\n";
    cout<< ctime (&koha) <<"\n";

    return 0;
}
```



**Diferenca e kohës:**

Për të llogaritur diferencën kohore, gjegjësisht kohën e kaluar prej një ngjarjeje deri te një tjetër, mund të përdoret funksioni **difftime()**, i cili si parametra i ka dy momente kohore, për të cilat llogaritet distancën kohore:

```
/* Shembull - difftime */
#include <iostream>
#include <ctime>
using namespace std;
```



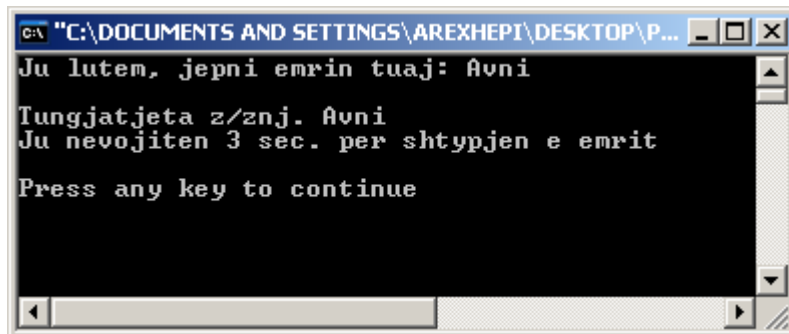
```

int main ()
{
    time_t fillimi, fundi;
    char VleraHyrese [25];
    double diferenca;

    time (&fillimi);
    cout<<"Ju lutem, jepni emrin tuaj: ";
    cin>>VleraHyrese;
    time (&fundi);
    diferenca = difftime (fundi, fillimi);
    cout<<"\nTungjatjeta z/znj. "<<VleraHyrese<<"\n";
    cout<<"Ju nevojiten "<<diferenca<<" sec. per shtypjen e emrit \n\n";

    return 0;
}

```



```

/* Shembull i llogaritjes së diferencës kohore - me precizitet të lartë */
#include <iostream>
#include <time.h>
using namespace std;

int main()
{
    int TikaTeKaluar;
    double MilisekondaTeKaluar, SekondaTeKaluar, MinutaTeKaluar;
    char VleraHyrese [100];

    clock_t Fillimi, Fundi;    // Fillimi dhe Fundi per timer

    Fillimi = clock() * CLK_TCK;    //fillo timer-in

    cout<<"Ju lutem, jepni emrin tuaj: "; //Pjesa ku shkaktohet vonese
    cin>>VleraHyrese;    //deri sa te shkruhet emri

    Fundi = clock() * CLK_TCK;    //ndale timer-in

    TikaTeKaluar = Fundi - Fillimi;
    //numri i tikave prej Fillimi deri te Fundi

    MilisekondaTeKaluar = TikaTeKaluar/1000;
    //milisekonda prej Fillimi deri te Fundi

    SekondaTeKaluar = MilisekondaTeKaluar/1000;
    //sekonda prej Fillimi deri te Fundi

    MinutaTeKaluar = SekondaTeKaluar/60;
}

```

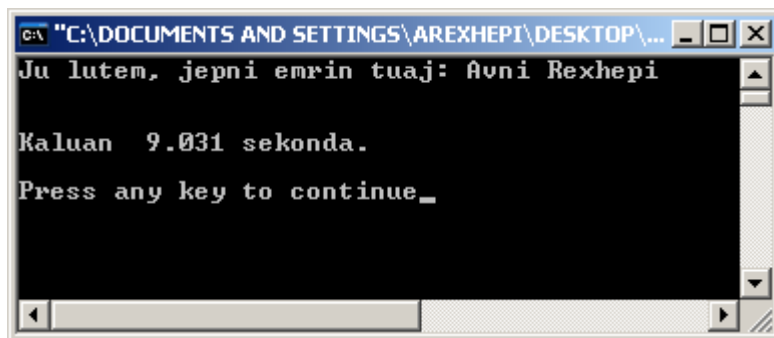
```

//minuta prej Fillimi deri te Fundi

if(SekondaTeKalar < 1)
    cout<<"\n\nKaluan "<<MilisekondaTeKalar<<" milisekonda.\n\n";
else if(SekondaTeKalar == 1)
    cout<<"\n\nKaloi 1 sekond.\n\n";
else if(SekondaTeKalar > 1 && SekondaTeKalar < 60)
    cout<<"\n\nKaluan "<<SekondaTeKalar<<" sekonda.\n\n";
else if(SekondaTeKalar >= 60)
    cout<<"\n\nKaluan "<<MinutaTeKalar<<" minuta.\n\n";

    return 0;
}

```



## Ngjyra e tekstit

Për të shtypur tekstet me ngjyra të ndryshme, duhet të përdoren disa funksione dhe disa të ashtuquajtur Handler (angl. Handler – mbajtës, manovrues, etj, të cilët mundësojnë manipulime të ndryshme në programe) nga heder fajlli <windows.h>. Meqenëse i tejkalon kufinjtë e asaj se çka shpjegohet në këtë semestër, do ta japim të gatshëm përmes një shembulli, ashtu që kur të ju nevojitet, e përdorni.

Mund të definohen 16 ngjyrat themelore, duke përorur numrat përkatës të tyre:

Black	= 0	- e zezë
Blue	= 1	- e kaltër
Green	= 2	- e gjelbër
Cyan	= 3	- cian
Red	= 4	- e kuqe
Magenta	= 5	- magjenta
Brown	= 6	- kafe
LightGray	= 7	- hiri, e qelët
DarkGray	= 8	- hiri e mbyllët
LightBlue	= 9	- e kaltër e qelët
LightGreen	= 10	- e gjelbër e qelët
LightCyan	= 11	- cyan e qelët
LightRed	= 12	- e kuqe e qelët
LightMagenta	= 13	- magjenta e qelët
Yellow	= 14	- e verdhë
White	= 15	- e bardhë

```

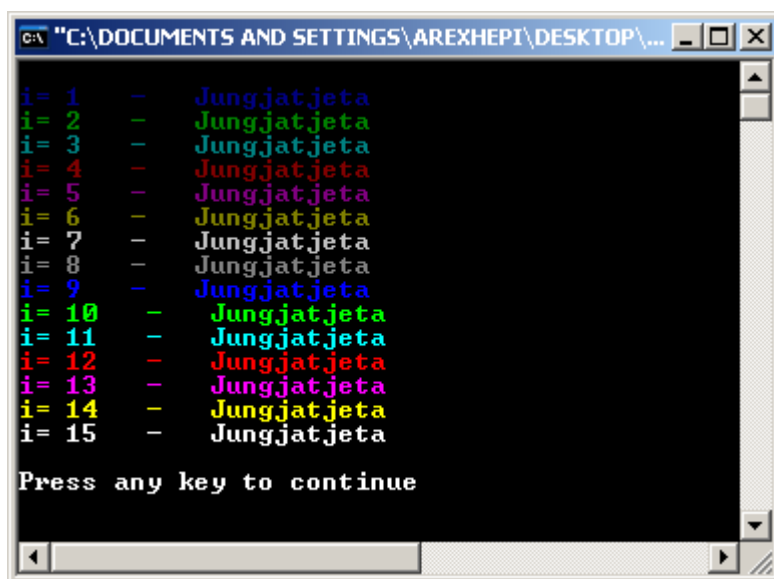
//Programi - teksti me ngjyra
#include <iostream>
#include <windows.h>
using namespace std;

void caktongjyren(unsigned short color)    //Funksioni per caktimin
{                                           //e ngjyres se tekstit
    HANDLE hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hcon,color);
}

int main()
{
    int i;

    for (i=0;i<=15;i++)
    {
        caktongjyren (i);
        cout<< "i= "<<i<< "    -    Jungjatjeta"<<endl;
    }
    cout<<"\n";
    return 0;
}

```



Me `#include` duhet t'a përfshini header fajllin `<windows.h>` dhe të definoni funksionin e dhënë me "handler-at" përkatës. Pastaj, para tekstit të cilin dëshironi t'a shtypni, e thirrni funksionin për ngjyrën e tekstit. Duhet pasur parasysh, që për të filluar shtypjen me ngjyrë të re, shtypja paraprake duhet të përfundohej më zbraze të buffer-it të tekstit, duke përdorur urdhërin "endl" ose "flush", pas shtypjes me "cout", si p.sh.:

```

cout<<"Jungjatjeta"<<endl;
    ose
cout<<"Jungjatjeta\n"<<flush;

```

**Shembull:** Ja një shembull i një loje, si shembull për përdorimin e numrave të rastit, matësit të kohës dhe përdorimit të ngjyrave në program:

```

#include <cstdlib>
#include <ctime>
#include <iomanip>
#include <iostream>
#include <windows.h>

using namespace std;

void ngjyra(unsigned short color); //funksioni ngjyrën e tekstit

int main()
{
    int numri_i_rastit, i;
    int hamendja, numri, tentime;
    time_t fillimi, fundi; //variablat për matjen e kohës
    double diferenca; //variabla për diferencën kohore

    //inicializimi i gjeneratorit të rastit
    srand(static_cast<unsigned>(time(0)));

    time (&fillimi); //koha e fillimit të lojës

    for (i=1; i<=10; i++) //prej 10 numrave të rastit, e marrim atë të fundit
    {
        numri_i_rastit = 1 + int(100 * rand()/(RAND_MAX + 1.0));
        //numrat mes 1 dhe 100

        // për t'i parë numrat e rastit, largoni komentin nga rreshti vijues
        // cout<<"Nr. i rastit - "<<numri_i_rastit<<endl;
    }

    // për ta parë numrin për lojën, largoni komentin nga rreshti vijues
    // cout<<"Nr. i rastit kesaj radhe - "<<numri_i_rastit<<endl;

    tentime=0; //numratori i tentimeve
    numri=numri_i_rastit;
    ngjyra(14); //ngjyra e verdhë
    cout << "Qellojani sa eshte numri i rastit mes 1 dhe 100\n" << endl;
    ngjyra(11); //ngjyra e kaltër e qelët
    cout << "Me pak - do te thote duhet numer me i vogel\n";
    ngjyra(12); //ngjyra e kuqe
    cout << "Me shume - do te thote duhet numer me i madh\n\n";
    do //fillimi i unazës
    {
        ngjyra(15); //ngjyra e bardhë
        cout << "Jepni numrin qe mendoni se ka qelluar: ";
        ngjyra(14); //ngjyra e verdhë
        cin >> hamendja;

        if (hamendja > numri)
        {
            ngjyra(11); //ngjyra e kaltër e qelët
            cout << "Me pak" << endl;
        }
        if (hamendja < numri)
        {
            ngjyra(12); //ngjyra e kuqe
            cout << "Me shume" << endl;
        }
        tentime=tentime+1; //pas çdo tentimi, rritet numratori i tentimeve
    } while (hamendja != numri); //deri sa ta qëlloni numrin e rastit

```

```

    time (&fundi);                                //koha e përfundimit të lojës

    ngjyra(10);                                    //ngjyra e gjelbër
    cout << "\n\nFituat. Numri i rastit eshte " << numri << endl;
    ngjyra(14);                                    //ngjyra e verdhë
    cout << "Per t'ia qelluar, juve ju nevojiten: ";
    ngjyra(11);                                    //ngjyra e kaltër e qelet
    cout << tentime << " tentime.\n";

    diferenca = difftime (fundi,fillimi);
    ngjyra(14);                                    //ngjyra e verdhë
    cout << "Per gjetjen e numrit ju nevojiten  : ";
    ngjyra(11);                                    //ngjyra e kaltër e qelet
    cout <<diferenca<<" sekonda. \n\n";
}

//Funksioni për caktimin e ngjyrës së tekstit
void ngjyra(unsigned short color)
{
    HANDLE hcon = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hcon,color);
}

```

```

C:\DOCUMENTS AND SETTINGS\AREXHEPI\DESKTOP\PROVA C++\...
Qellojani sa eshte numri i rastit mes 1 dhe 100
Me pak - do te thote duhet numer me i vogel
Me shume - do te thote duhet numer me i madh
Jepni numrin qe mendoni se ka qelluar: 50
Me pak
Jepni numrin qe mendoni se ka qelluar: 30
Me shume
Jepni numrin qe mendoni se ka qelluar: 35
Me shume
Jepni numrin qe mendoni se ka qelluar: 40
Me pak
Jepni numrin qe mendoni se ka qelluar: 36

Fituat. Numri i rastit eshte 36
Per t'ia qelluar, juve ju nevojiten: 5 tentime.
Per gjetjen e numrit ju nevojiten : 7 sekonda.
Press any key to continue_

```

## Shtojca B - Pointerët

Pointerët janë variabla të cilat mbajnë adresat në C dhe në C++. Ata ofrojnë fuqi dhe pajisje për programerin që të ju qaset dhe të manipulojnë të dhënat në një mënyrë të ndryshme që nuk ofrohet në disa gjuhë tjera programuese. Ata janë të dobishëm për përcjelljen e parametrave në funksione, në mënyrë që të lejojnë funksionin që të ndryshojë dhe të kthejë vlerat në programin thirrës. Kur përdoren në mënyrë të gabuar, ata shpesh janë burim i gabimeve në program dhe frustrimeve për programerët.

Kur programi ekzekutohet, të gjitha variablat ruhen në memorie, secila në ndonjë lokacion apo adresë të veçantë. Në mënyrë tipike, variabla dhe adresa e saj e memories përmbajnë vlerat e të dhënave. P.sh., kur deklarohet:

```
int numri = 5;
```

vlera 5 ruhet në memorie dhe mund t'i qasemi përmes variablës "numri". Pointeri është lloj special i variablës, që e ruan adresën e memories e jo vlerën e të dhënave. Njësoj siç ndryshohet vlera kur përoret variabla normale, vlera e adresës së ruajtur në pointer ndryshohet kur manipulohet me variablën pointer.

Zakonisht, adresa e ruajtur në pointer është adresa e ndonjë variable tjetër.

```
int *ptr;
ptr = &numri    // Ruan adresën e variablës numri në ptr.
                // Operatori unar & (address of - adresa e)
                // kthen adresën e variablës.
```

Për të marrë vlerën e cila është e ruajtur në lokacionin e memories të pointerit duhet të bëhet "dereferencimi" i pointerit. Dereferencimi bëhet përmes operatorit unar "\*" (pointer to – treguesi në, aty ku tregon pointeri).

```
int totali;
totali = *ptr;
        // Vlera në adresën e ruajtur në ptr i ndahet variablës totali.
```

Variablat që i deklaroni në program, ekzistojnë në memorie gjatë kohës së ekzekutimit të programit. Secila variabël, vendoset në një adresë të caktuar të memories.

P.sh., nvse ne deklarojmë një variabël të tipit integer, si në vijim:

```
int numriIm = 25;
```

kjo do të zë një sasi të mjaftueshme të memories për një vlerë integer dhe do të inicializojë atë memorie për të mbajtur vlerën që e prezenton numrin 25 në formatin integer. Ne nuk e dimë se cila është adresa aktuale ku është vendosur vlera e jonë, por kjo është një çështje për të cilën kujdeset programi (C++) dhe ai e di ku ta gjejë atë. Normalisht, nëse nuk do ta dinte ku e ka vendosur dhe ruajtur, nuk do të kishte mundësi t'i qasej përsëri asaj vlere. Në fakt, adresa aktuale ku ruhet variabla numriIm mund të ndryshojë secilën herë që ekzekutohet programi, por programi e di se si të merret me këtë problem. Kryesorja, ai e di çdo herë se ku i ka ruajtur vlerat, pra në cilën adresë në memorie.

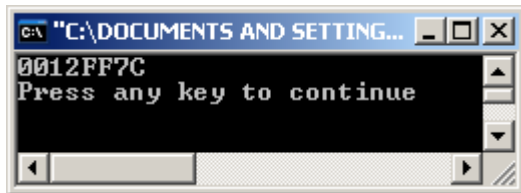
Siç u tha më parë, C++ ofron një mënyrë që t'i referohemi adresës së variablës, duke e përdorur operatorin "address of" (adresa e), që është simboli "&" (ampersand) i lidhur me emrin e variablës. Pra, adresën e memories së variablës numriIm mund t'i referohemi përmes:

```
&numriIm.
```

Pasi paska mundësi t'i referohemi adresës, atëherë ne mund ta paraqesim atë në dalje (ta shtypim në ekran), si në vijim:

```
#include <iostream>
using namespace std;
int main()
{
    int numriIm = 25;
    cout << &numriIm << endl;
    return 0;
}
```

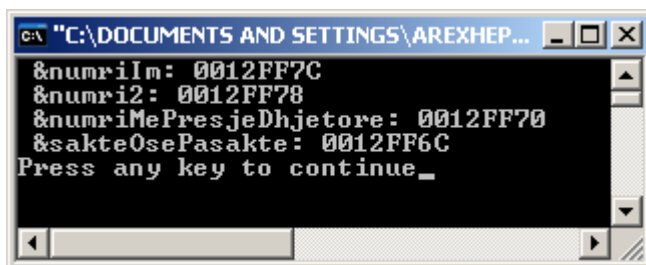
Kur të ekzekutohet programi, në dalje(ekran) do të paraqitet një vlerë, si:



Kjo vlerë paraqet adresën e memories në të cilën është ruajtur vlera 25 në formatin heksadecimal (Kjo vlerë ndryshon varësisht prej kompjuterit të përdorur, por formati si adresë është gjithmonë i këtillë).

Operatorin e adresës mund ta përdorim edhe për tipet tjera të variablave. P.sh., provoni të ekzekutoni programin vijues:

```
#include <iostream>
using namespace std;
int main()
{
    int numriIm = 37;
    int numri2 = 42;
    double numriMePresjeDhjetore = 1.234;
    bool sakteOsePasakte = true;
    cout << " &numriIm: " << &numriIm << endl;
    cout << " &numri2: " << &numri2 << endl;
    cout << " &numriMePresjeDhjetore: " << &numriMePresjeDhjetore << endl;
    cout << " &sakteOsePasakte: " << &sakteOsePasakte << endl;
    return 0;
}
```



Pra, siç mund të shihet, secila variabël ka adresën e vet dhe zë hapësirën e vet në memorie.

## Ruajtja e adresës në pointer (tregues)

Tani që u pa se si mund t'i referohemi adresës së memories, le të shohim se si mund ta ruajmë këtë adresë të memories. Për këtë qëllim, na hyjnë në punë pointerët (angl. Pointer – tregues, shenjues).

Pointeri është variabël që ruan një adresë.

Pra, pointeri është një tip i variablës në C++. Njësoj siç C++ i ka tipet integer për t'i ruajtur numrat e plotë, double për numrat real, bool për variablat buleanë etj., C++ e ka tipin pointer për t'i ruajtur adresat.

Më herët patëm variablën integer “numriIm”

```
int numriIm = 25;
```

e gjithashtu e paraqitëm adresën e tij me:

```
cout << &numriIm << endl;
```

Tani, le të shohim si ta ruajmë adresën e variablës numriIm në një variabël pointer.

Kur të deklarohet variabla e tipit pointer duhet t'ia tregojmë tipin e variablës që ruhet në atë adresë ku tregon ai, gjegjësisht duhet të tregojmë se për në cilin tip të variablës tregon pointeri. Të shohim si deklarohet një pointer që tregon në variablën tonë, të tipit integer. Kjo mund të bëhet në ndonjërin prej mënyrave vijuese:

```
int* PointeriIm;  
int *PointeriIm;
```

Pra, fillohet me deklarin e tipit “int” dhe pastaj si simbol që tregon se është fjala për pointer, pason simboli “\*” (ylli) i cili mund të shënohet menjëherë pas tipit ose para emrit të variablës (kah ana e variables).

Gjithashtu, mund ta deklarojmë dhe ta inicializojmë drejtpërdrejtë variablën e tipit pointer:

```
int *PointeriIm = &myInteger;
```

Kështu, kur ekzekutohet programi, ai e ndanë hapësirën e nevojshme të memorjën për ta ruajtur adresën e memories. Ai e ruan në këtë hapësirë adresën e variablës numriIm, e cila si u pa më herët në program, ishte: 0012FF7C

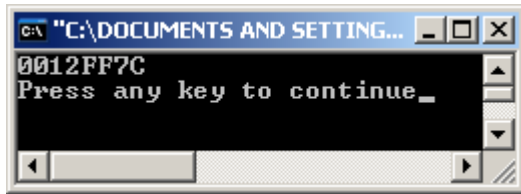
Me fjalë tjera, vlara: 0012FF7C ruhet në variablën PointeriIm.

Le të shohim programin vijues:

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int numriIm = 25;  
    int* PointeriIm = & numriIm;  
    cout << PointeriIm << endl;  
    return 0;  
}
```

Kësaj radhe, në vend se të shtypim drejtpërdrejt adresën e variablës numriIm, ne e kemi ruajtur atë në pointerin PointeriIm dhe pastaj e kemi shtypur vlerën e tij. Me rastin e ekzekutimit fitohet rezultati:

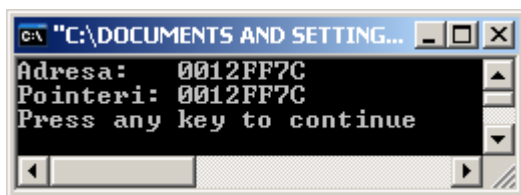




(Vërejtje: mos u brengosni nëse vlera që ju e fitoni është tjetër, pasi që kjo mund të jetë e ndryshme, sa herë që të ekzekutohet programi ose mund të jetë e njëjta vlerë, pas disa ekzekutimeve. Kjo gjë është e varur prej platformës (kompjuterit) dhe cilado mënyrë e paraqitjes është normale).

Për të qenë edhe më bindëse, le të shohim programin vijues:

```
#include <iostream>
using namespace std;
int main()
{
    int numriIm = 25;
    int *PointeriIm = & numriIm;
    cout << "Adresa:   " << &numriIm << endl;
    cout << "Pointeri: " << PointeriIm << endl;
    return 0;
}
```



Pra, adresën e kemi shtypur drejtpërdrejt dhe përmes pointerit. Të dy vlerat duhet të jenë identike.

Për këtë program themi se variabla PointeriIm tregon (pointon) në variablën numriIm. Kjo është vetëm një mënyrë e përshtatshme e shprehjes së faktit që pointeri mbanë (ruan) adresën e variablës numriIm.

Tipi i pointerit PointeriIm është “int”. Pra, nëse kemi dy deklarime si në vijim, duhet të shihet dallimi mes deklarimit të variablës së tipit integer dhe deklarimit të treguesit (pointerit) në variablën e tipit integer:

```
int n;           // Deklarimi i një variable të tipit integer
int *p;          // Deklarimi i pointerit në një variabël të tipit integer
```

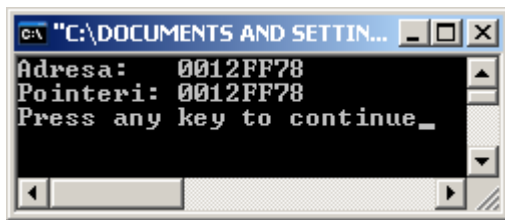
Shembulli në vijim është i njëjtë me të mëparshmin, vetëm se këtë herë për variabël të tipit numër real (numër me presje dhjetore, floating Point - numër me pikë të lëvizshme).

```
#include <iostream>
using namespace std;
int main()
{
    double numriReal = 1.234;
    double *Pointeri2 = &numriReal;
```

```

cout << "Adresa: " << &numriReal << endl;
cout << "Pointeri: " << Pointeri2 << endl;
return 0;
}

```



Vëreni se deklarimi i variablës pointer reflekton faktin se ajo tani tregon (pointon, shenjon) në variablën e tipit double.

```

double *Pointeri2 = &numriReal;
ose
double* Pointeri2 = &numriReal;

```

Më parë kishim:

```
int *PointeriIm = &numriIm;
```

Pra, si u tha edhe më parë, tipi i paraqitur në deklarimin e pointerit na tregon se në çfarë tipi të variablës tregon (pointon, shenjon) pointeri: int, double etj.

### Si i qasemi variablës përmes pointerit

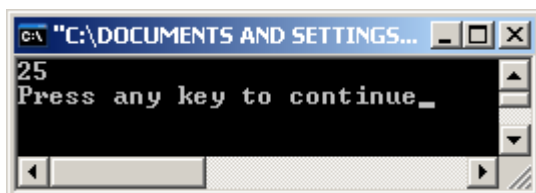
Tani që e dijme si të marrim adresën e variablës dhe si ta ruajmë atë në pointer, le të shohim se si mund të përdoret pointeri për të ju qasur variablës në të cilën ai tregon (pointon, shenjon). Nuk është me rëndësi se kur ose përse do të na duhej kjo gjë, kryesorja është që të kuptohet se si mund të bëhet kjo, e pastaj kurdo që të nevojitet, të mund të përdoret.

**Shembull:**

```

#include <iostream>
using namespace std;
int main()
{
int numriIm = 25;
int *PointeriIm = &numriIm; //Pointeri ruan adresën e variablës
cout << *PointeriIm << endl; //shtypet variabla/vlera në të
cilën tregon pointeri
return 0;
}

```



Kodi i mësipërm është i njohur prej shembujve paraprak, përveq rreshtit:

```
cout << *PointeriIm << endl;
```

Pra, siç shihet, me rastin e ekzekutimit shtypet vlera 25.

Kemi deklaruar pointerin në variablën e tipit int (Integer) dhe e kemi quajtur PointeriIm. E inicializojmë vlerën e PointeriIm për ta mbajtur adresën e variablës së tipit Integer numriIm (gjë që e pamë edhe më parë) dhe pastaj, me:

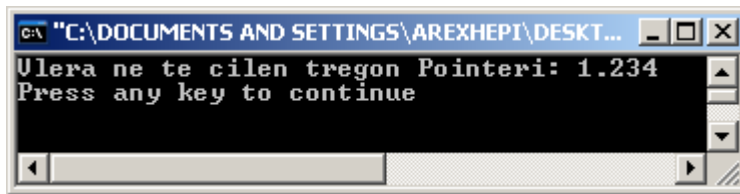
```
*PointeriIm
```

në fakt i qasemi vlerës së numrit integer në të cilën tregon pointeri, e cila në këtë rast është 25.

Pra, ky urdhër bën që të shtypet vlera në të cilën tregon pointeri. Ky ishte operatori i dereferencimit të vlerës nga pointeri.

Ja edhe një shembull, që përdorë pointerin në variablën e tipit double, që e pamë më parë:

```
#include <iostream>
using namespace std;
int main()
{
    double numriReal = 1.234;
    double *Pointeri2 = &numriReal;
    cout << "Vlera ne te cilen tregon Pointeri: " << *Pointeri2 <<
endl;
    return 0;
}
```



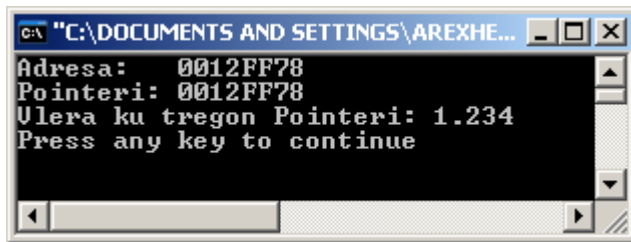
Edhe këtë herë, programi është i ngjashëm me atë të mëparshmin për numrat real, vetëm se në fund shtypet me urdhërin

```
cout << *Pointer2 << endl;
```

me të cilin në dalje shtypet variabla në të cilën tregon pointeri Pointeri2, që në këtë rast është vlera 1.234.

Me pak ndryshime, në vijim kemi:

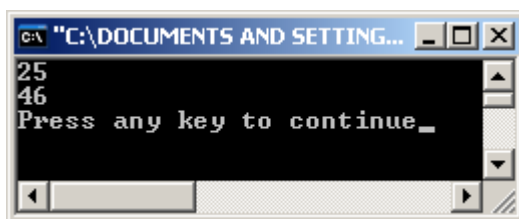
```
#include <iostream>
using namespace std;
int main()
{
    double numriReal = 1.234;
    double *Pointeri2 = &numriReal;
    cout << "Adresa: " << &numriReal << endl;
    cout << "Pointeri: " << Pointeri2 << endl;
    cout << "Vlera ku tregon Pointeri: " << *Pointeri2 << endl;
    return 0;
}
```



## Ndryshimi i vlerave përmes pointerëve

Vlerat në të cilat tregon pointeri mund të ndryshohen edhe përmes pointerit:

```
#include <iostream>
using namespace std;
int main()
{
    int numriIm = 25;
    int* PointeriIm = &numriIm;
    cout << numriIm << endl;
    *PointeriIm = 46;          //aty ku tregon pointeri=46
    cout << numriIm << endl;
}
```



Sikur në rastet e mëparme, fillojmë me inicializimin e variablës numriIm me vlerën 25, pastaj inicializojmë pointerin PointeriIm me adresën e variablës numriIm. Në rreshtin vijues, e shtypim vlerën e variablës numriIm, që në dalje jep vlerën 25. Pastaj, rreshti/urdhëri:

```
*myPointer = 46;
```

ia ndanë vlerën 46 variablës në të cilën tregon pointeri PointeriIm. Me fjalë të tjera, ia ndanë vlerën 46 variablës numriIm (sepse pointeri tregon në të). Efektivisht, kjo është ekuivalente me urdhërin:

```
numriIm = 46.
```

Është me rëndësi të kuptohet se çka po ndodhë ketu. Vlera 46 nuk i ndahet pointerit. Në fakt, vet variabla e pointerit nuk është ndryshuar fare në asnjë mënyrë dhe ajo akoma vazhdon të ruaj (mbajë) adresën e memories së variablës së tipit integer numriIm. Ajo çka kemi bërë është, sikur se të themi: merre adresën e cila ruhet në pointerin PointeriIm dhe ndryshoje (modifikoje) variablën që është e ruajtur në atë adresë, ashtu që të jetë 46. Në vend se t'i referohemi drejtpërdrejt variablës numriIm, i jemi referuar duke përdorur adresën e saj, të ruajtur në pointerin PointeriIm.

Rreshti i fundit e shtypë në dalje vlerën e variablës numriIm përsëri, po këtë herë ajo është 46.

Ja edhe versioni për numrin real:

```
#include <iostream>
using namespace std;
int main()
{
    double numriReal = 1.234;
    double *Pointeri2 = &numriReal;
    cout << numriReal << endl;
    *Pointeri2 = 9.876;
    cout << numriReal << endl;
    return 0;
}
```



Urdhëri:

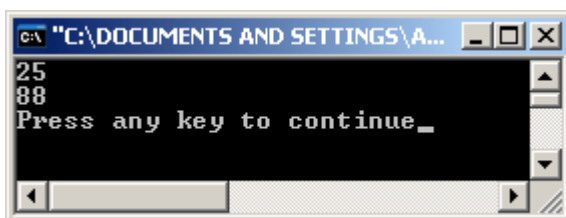
```
*Pointeri2 = 9.876;
```

ia ndanë vlerën 9.876 variablës në të cilën tregon pointeri, që në këtë rast është numriReal.

### Ndryshimi i variablës së tipit pointer

Njësoj si edhe me variablat tjera, mund të ndryshojmë vlerën të cilën e ruan (mbanë) variabla e tipit pointer. Pasi që pointeri ruan adresë, kur të ndryshohet vlera e tij do të thotë se atij po i ndahet ndonjë adresë tjetër. Le të shohim një shembull:

```
#include <iostream>
using namespace std;
int main()
{
    int numri1 = 25;
    int *PointeriIm = &numri1;
    cout << *PointeriIm << endl;
    int numri2 = 88;
    PointeriIm = &numri2;
    cout << *PointeriIm << endl;
}
```



Në fillim inicializojmë pointerin PointeriIm për të ruajtur adresën e variablës numri1 dhe për të vërtetuar këtë e shtypim vlerën në të cilën tregon ai, gjë që do të shtypë vlerën 25. Pastaj, pointerit PointeriIm ia ndajmë adresën e variablës tjetër numri2.

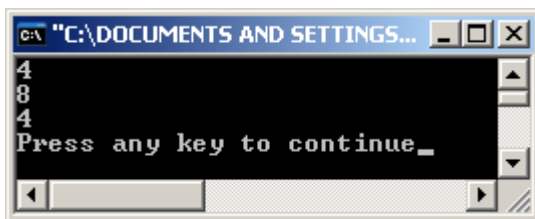
```
PointeriIm = &numri2;
```

Tani pointeri tregon në variablën numri2, kështu që kur ta shtypim përsëri vlerën në të cilën tregon ai, do të shtypet vlera 88.

### Përdorimi i pointerëve si parametra të funksioneve

Kur funksionit ia përcjellim ndonjë variabël, krijohet një kopje e variablës dhe përdoret përbrenda funksionit. Vlera origjinale nuk ndikohet nga ndryshimet që i bëhen kopjes përbrenda funksionit. P.sh., le të marrim rastin vijues:

```
#include <iostream>
using namespace std;
void shtypeDyfishin(int vleraLokale)
{
    vleraLokale = vleraLokale * 2;
    cout << vleraLokale << endl;
}
int main()
{
    int Vlera1 = 4;
    cout << Vlera1 << endl;
    shtypeDyfishin(Vlera1);
    cout << Vlera1 << endl;
    return 0;
}
```



Pra, në fillim inicializojmë variablën Vlera1 me vlerën 4 dhe e shtypim vlerën e saj. Pastaj, e thërrasim funksionin shtypeDyfishin dhe ia përcjellim variablën Vlera1 si parametër, me vlerën 4. Në funksion, krijohet kopja lokale e variablës Vlera1 me emrin vleraLokale dhe ajo shumëzohet me 2. Pastaj vlera e re e variablës vleraLokale shtypet në ekran.

Variabla Vlera1 mbetet e pandikuar nga ndryshimet e bëra në kopjen lokale përbrenda funksionet, kështu që kur të shtypet për herë të dytë variabla Vlera1, vlera e saj akoma është 4.

Le të shohim në vazhdim, çka do të ndodhë, nëse funksionit ia përcjellim pointerin që tregon në variabël, në vend të variablës:

```
#include <iostream>
using namespace std;
void shtypeDyfishin(int *PointeriIm)
{
    *PointeriIm = *PointeriIm * 2;
    cout << *PointeriIm << endl;
}
int main()
{
    int Vlera1 = 4;
    cout << Vlera1 << endl;
    shtypeDyfishin(&Vlera1);
    cout << Vlera1 << endl;
    return 0;
}
```

Kësaj radhë funksioni prej treguesin (pointerin) në një variabël të tipit integer (në Vlera1) që t'i përcillet si parametër.

```
void shtypeDyfishin(int *PointeriIm)
```

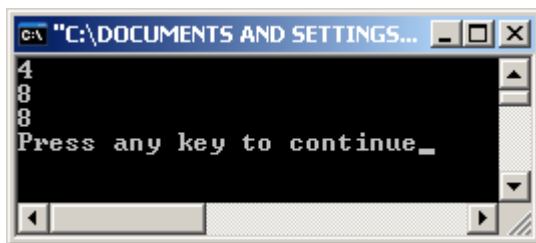
Kur e thërrasim funksionin, atij ia përcjellim adresën e variablës Vlera1.

```
shtypeDyfishin(&Vlera1);
```

Kur e ndryshojmë vlerën në të cilën tregon pointeri PointeriIm përbrenda funksionit, pasi që pointeri mbanë adresën e variablës Vlera1, çfarëdo ndryshimi i bërë në vlerën në të cilën tregohet me pointer në fakt ndryshon edhe vlerën e ruajtur nga variabla Vlera1.

```
*PointeriIm = *PointeriIm * 2; // dyfishon vlerën e ruajtur në Vlera1
```

Kur kthehet vlera nga funksioni, Vlera1 mbanë vlerën e re 8. Prandaj, rezultati i këtij programi do të jetë:



Nëse e krahasojmë me versionin e mëparshëm, i cili në dalje kishte:

```
4
8
4
```

shihet dallimi i ndikimit të funksionit në parametrat e përcjellur si vlerë e variablës ndaj rastit të përcjelljes së adresës së variablës.

Pra, kur e përcjellim variablën si parametër të funksionit, krijohet një kopje e variablës e cila përdoret prej funksionit dhe çfarëdo ndryshimesh të bëra e lënë variablën origjinale (jashtë funksionit) të paprekur (të pandikuar). Nëse funksionit ia përcjellim pointerin, mund ta ndryshojmë edhe variablën origjinale përmes pointerit (i cili tregon në atë variabël).

## Dukshmëria e variablave

Variablat e deklaruar përbrenda funksionit ose në ndonjë bllok tjetër të brendshëm të programit janë të dukshme (kanë fushëveprim) vetëm përbrenda bllokut të tyre dhe nuk mund të përdoren jashtë kufinjëve të tyre. P.sh, në shembullin vijues, kemi variablat globale: numri dhe Emri, të cilat mund të përdoren në cilëndo pjesë të programit dhe kemi variablat lokale: Vlera1 dhe Vlera2 të funksionit main( ) dhe ato: a, b, S të funksionit Siperfaqja.

#include <iostream> using namespace std;	
int numri; char Emri[15];	<b>Variablat Globale</b>
int main() {	
int Vlera1; double Vlera2;	<b>Variablat Lokale</b>
... cout << Vlera1 << endl; cout << Emri << endl; cout << S << endl;	<b>Urdhërat</b>
return 0; }	
int Siperfaqja(int a, int b) { int S S=a*b; return S }	<b>Variablat Lokale</b>

Pra, dukshmëria e variablave është e kufizuar përbrenda nivelit të bllokut në të cilin janë deklaruar. Sidoqoftë, kemi mundësi të deklarojmë variablat globale të cilat janë të dukshme në cilëndo pjesë të kodit, edhe përbrenda edhe jashtë funksioneve. Për të deklaruar variablën globale, ajo thjeshtë duhet të deklarohet jashtë cilitdo funksion ose bllok, drejtpërdrejt në trupin e programit.



## Shtojca C - Shkruarja dhe leximi i fajllave

C++ mundëson shkruarje dhe leximin e të dhënave (manipulimin e tyre) nga dhe në fajlla. Kjo është një prej çështjeve shumë të rëndësishme të çdo gjuhe programuese. Cilido aplikacion nga jeta reale është e pritshme që të përpunojë sasi të mëdha të informacionit. Një prej teknikave të thjeshta të transferimit dhe regjistrimit të të dhënave është përmes fajllave. Teknikat më të avancuara për ruajtjen dhe manipulimin e të dhënave përdorin bazat relacionale të të dhënave ose ndonjë format special si XML. Për fillim, le të shohim si përdoren fajllat tekstual.

Një prej çështjeve të rëndësishme me rastin e shkruarjes dhe leximit nga fajllat është formati i të dhënave. Pa marrë parasysh se shfrytëzuasi fundor i programit do të jetë ndonjë kopjuter tjetër ose ndonjë person, formati i dhënë në dalje mund të jetë po aq i rëndësishëm sa edhe vet përmbajtja. Nëse dalja konsumohet nga ndonjë program tjetër, atëherë formati në dalje me gjasë do të jetë i parapërcaktuar dhe specifik, në mënyrë që programi konsumues të jetë në gjendje të lexojë dhe të ju qaset të dhënave. Nëse fajlli në dalje është për t'u lexuar nga njerëzit, formati i të dhënave mund të ndikojë në kuptueshmëri dhe dobi. Manipulatorët iostream ofrojnë një mënyrë për të kontrulluar formatin dalës.

### Hyrja dhe dalja nga fajlli

Teknikat për hyrje dhe dalje, i/o, në C++ janë virtualisht identike me ato të prezentuara më parë për leximin dhe shtypjen në dhe nga pajisjet standarde, ekrani dhe tastiera. Për të realizuar hyrje dhe dalje në fajlla duhet të përdoret direktiva: `#include <fstream>`.

Fstream përmbanë definicionet e klasave për klasat e përdorura në hyrje/dalje të fajllave. Përbrenda programit që ka nevojë për hyrje dalje të fajllave, për secilin fajll të kërkuar në dalje, inicializohet një objekt i klassës “ofstream”. Objekti “ofstream” përdoret njësoj si objekti cout për dalje standarde. Objekti “ifstream” përdoret njësoj si objekti standard “cin”.

### Shembull:

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream FajlliIm("c:/out.txt");
        // Krijon ofstream objektin me emrin FajlliIm

    if (! FajlliIm) // Gjithnmonë testo hapjen e fajllit
    {
        cout << "Gabim me rastin e hapjes së fajllit per
dalje/shkruarje" << endl;
        return -1;
    }

    FajlliIm << "Jungjatjeta" << endl;

    FajlliIm.close();

    return 0;
}
```

Pra, së pari krijohet objekti ofstream, me emrin FajlliIm. Konstruktori për klasën ofstream i merr dy argumente. I pari tregon emrin e fajllit, si C-string, kurse i dyti modin e fajllit. Ekzistojnë dy mode të zakonshme të hapje së fajllit: truncate (prerja) dhe append (shtimi). E nënkuptuar, nëse nuk specifikohet modi dhe fajlli veç ekziston, ai prehet (do të thotë, fajlli ekzistues fshihet dhe krijohet i ri, me të njëjtën emër). Modi specifikohet përmes përdorimit të definimit të enumeratorit në klasën ios. Klasa ios përmbanë anëtarët të cilët i përshkruajnë modet dhe gjendjet për klasat input (hyrja) dhe output (dalja). Enumeracioni (grupimi) është një mënyrë e definimit të vargut të konstanteve. Megjithatë, këto i merrni të gjitha si të gatshme.

- Nënkuptohe, që nëse fajlli nuk ekziston, ai krijohet
- Nënkuptohe, që fajlli do të jetë tekst fajll, e jo fajll binar
- Nënkuptohe, që fajlli ekzistues do të cunohet (prehet)
- Për të kombinuar modet, bashkoni me “or” duke përdorur “|”.

Modi	Kuptimi
Out	Hapje fajllin ose rrjedhën për shkruarje/insertim (output-dalja)
In	Hapje fajllin ose rrjedhën për lexim/ekstraktim (input-hyrje)
App	Bashkangjit/shto në fajllin ekzistues. Secili insertim (dalje) do të shkruhet në fund të fajllit.
Trunc	Preje (fshije) fajllin ekzistues (sjellja e zakonshme, e nënkuptuar).
ate	E hapë fajllin pa e fshirë, por lejon që të dhënat të shkruhen kudo brenda tij.
binary	Trajto fajllin si fajll binar. Fajlli binar i ruan të dhënat në format intern (ninar, 1 dhe 0), e jo në fajll të lexueshëm tekstual. P.sh., fload do të ruhet si prezantimi i tij prej katër bajtash, e jo si string.

Për të hapur fajllin, duke fshirë/prerë përmbajtjen ekzistuese, mund të përdoret cilido prej urdhërave vijues:

```
ofstream FajlliIm("EmriiFajllit");
ofstream FajlliIm("EmriiFajllit", ios::out);
ofstream FajlliIm("EmriiFajllit", ios::out | ios::trunc);
```

Për të hapur fajllin, për të shtuar në përmbajtjen e mëparshme:

```
ofstream myFile("EmriiFajllit", ios::app);
```

Për të hapur fajllin për dalje binare:

```
ofstream myFile("EmriiFajllit ", ios::binary);
```

Për të hapur fajllin për hyrje (lexom prej tij), përdoret objekti i klasës ifstream:

```
ifstream inFile("EmriiFajllit ");
```

### Shembull:

Shkruarja e programit që kërkon prej shfrytëzuesit emrin e fajllit dhe pastaj shkruan në atë fajll. Pastaj, hapet fajlli në modin e shtimit ashtu që përmbajtja ekzistuese të mos humbet nëse shfrytëzuesi specifikon ndonjë fajll ekzistues.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
```

```

string EmriiFajllit;

cout<<"Jepni emrin e fajllit, p.sh.: c:/ndonjeEmer.txt"<< endl;
cin >> EmriiFajllit;

ofstream oFajlli(EmriiFajllit.c_str(),ios::app);

if (! oFajlli)
{
    cout<<"Gabim, me rastin e hapjes se fajllit per dalje "<< endl;
    return -1;
}

oFajlli << "Jungjatjeta" << endl;
oFajlli << "Zgjedhje e parashikuar e daljes, apo jo??" << endl;

return 0;
}

```

### Metoda të dobishme për klasat Input dhe Output

Klasa ifstream ka disa metoda të dobishme për input (hyrje). Këto metoda ndodhen gjithashtu edhe në klasën cin, që përdoret për lexom prej hyrjes standarde (tastierës). Këto metoda përdoren për të lexuar prej ndonjë rrjedhe hyrëse. Një rrjedhë hyrëse është burim i hyrjes, si: tastiera, fajlli ose baferi.

- Operatori i nxjerrjes >>  
Ky operator i mbingarkuar trajton të gjitha tipet brendshme të të dhënave në C++. Nënkuptohe se, hapësirat nuk mirren parasyshë. Pra, zbrazëtira, tab, rreshti i ri, formfeed, Enter etj., evitohen.
- get()  
Kjo formë e get, merr një karakter nga rrjedha hyrëse, d.t.th, prej hyrjes standarde, fajllit ose baferit. Nuk i eviton zbrazëtirat. Kthen tipin int.
- get(char &ch)  
Kjo formë e get, merr një karakter nga rrjedha hyrëse, por e ruan vlerën në variablën karakter të përcjellur brenda si argument.
- get(char \*buff, int buffsize, char delimiter='\n')  
Kjo formë e get, lexon karakteret në C-string baferin e përcjellur si argument (buffsize karaktere lexohen), deri te ndaresi (delimiter-i) ose deri sa të haset fundi i fajllit. “\n” është karakteri për rreshtin e ri. Delimiteri nuk lexohet në bafer, por në vend të kësaj, lihet në rrjedhën hyrëse. Duhet të largohet veçantë, por duke përdorur një tjetër “get” ose “ignore” (injoinim). Për shkak të këtij hapi të shtuar, kjo formë e get shpesh është burim i gabimeve dhe duhet të evitohet. Për fat të mirë, ekziston një metodë më e mire, “getline”, e cila e lexon edhe delimiterin dhe duhet të përdoret në vend të kësaj forme të “get”.
- getline  
Ka disa forma të dobishme të “getline”.
- ignore(int n=1, int delim=traits\_type::eof)  
Kjo metodë lexon dhe inojoron n karaktere nga rrjedha hyrëse.
- peek()  
Kjo metodë kthen karakterin e ardhëshëm prej rrjedhës hyrëse, por nuk e largon atë. Është e dobishme për të shikuar paraprakisht se cili është karakteri i ardhëshëm.
- putback(char &ch)  
Kjo metodë e vendosë “ch-në” në rrjedhën hyrëse. Karakteri në “ch” pastaj do të jetë karakteri i ardhëshëm i lexuar prej rrjedhës hyrëse.
- unget()  
Kjo metodë, e rivendosë karakterin e fundit, në rrjedhën hyrëse.

- `read(char *buff, int n)`  
Kjo metodë përdoret për të realizuar një lexim të paformatizuar të `n` bajtave nga rrjedha hyrëse në baferin e karaktereve.

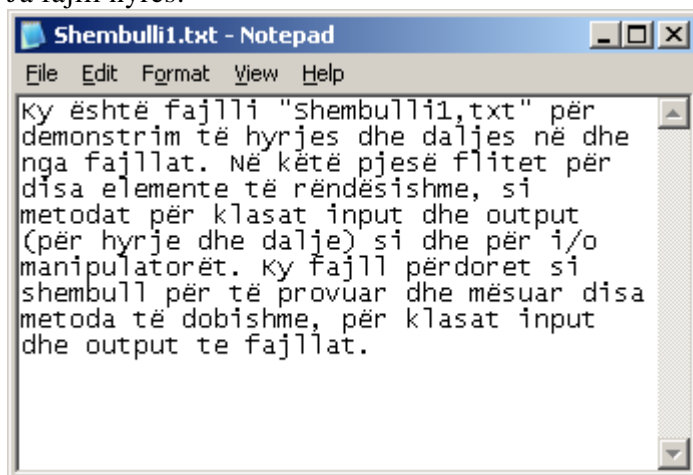
Klasa `ofstream` ka disa metoda të dobishme për rrjedhën dalëse. Një rrjedhë dalëse është dalja standarde (ekrani), fajlli ose baferi. Këto metoda, gjithashtu ndodhen në objektin `cout`, që përdoret për dalje standarde.

- Operatori i insertimiti, `<<`  
Ky operator i mbingarkuar do të trajtojë të gjitha tipet e brendshme të të dhënave të C++-it.
- `put(char ch)`  
Kjo metodë vendosë një karakter në rrjedhën dalëse.
- `write(char *buff, int n)`  
Kjo metodë përdoret për të realizuar një shkruarje të paformatizuar prej baferit të karaktereve në rrjedhën dalëse.

### Shembull:

Supozojmë se duhet të lexohet fajlli dhe të përcaktohet numri i karaktereve alfanumerike, i zbrazëtirave dhe i fjalive. Për të përcaktuar numrin e fjalive do të numrohen pikat (.). nuk do të mirren parasysh kryerreshtat dhe "tab-et".

Ja fajlli hyrës:



### Programi:

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    int zbrazetira = 0;
    int numri_i_karaktereve = 0;
    int numri_i_fjalive = 0;
    char ch;

    ifstream inputFajlli("c:/Shembulli1.txt");

    if (! inputFajlli)
    {
        cout << "Gabime, me rastin e hapjes se fajllit" << endl;
        return -1;
    }
}
```

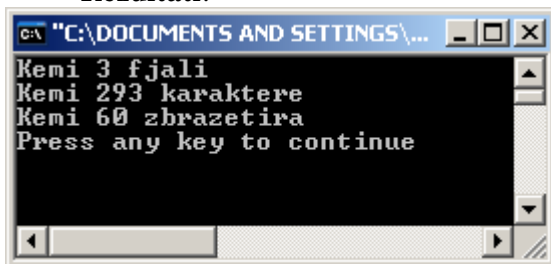
```

    }

    while (inputFajlli.get(ch))
    {
        switch (ch) {
            case ' ':
                zbrazetira++;
                break;
            case '\n':
            case '\t':
                break;
            case '.':
                numri_i_fjalive++;
                break;
            default:
                numri_i_karaktereve++;
                break;
        }
    }

    cout << "Kemi " << numri_i_fjalive << " fjali" << endl;
    cout << "Kemi " << numri_i_karaktereve << " karaktere" << endl;
    cout << "Kemi " << zbrazetira << " zbrazetira" << endl;
    return 0;
}

```

**Rezultati:****Shembull:**

Programi që kopjon përmbajtjen e një fajlli, në një fajll tjetër. Ky program, i kërkon shfrytëzuesit që të jep emrat e fajllave për hyrje dhe për dalje dhe pastaj e bën kopjimin.

```

#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main()
{
    char ch;
    string inputFajlli;
    string outputFajlli;

    cout << "Jepni emrin e fajllit burimor, si (c:\inputfajlli): ";
    cin >> inputFajlli;

    cout << "Jepni emrin e fajllit cak, si (c:\outputfajlli): ";
    cin >> outputFajlli;
}

```

```

ofstream oFajlli(outputFajlli.c_str());
ifstream iFajlli(inputFajlli.c_str());
//Verifikimi i gabimit gjate hapjes se fajllit
//eshte anashkaluar per shkurtesi.

while (iFajlli.get(ch))
{
    oFajlli.put(ch);
}

return 0;
}

```

### I/O Manipulatorët

Deri tani, kemi pranuar formatizimin e nënkuptuar të daljes. C++ definon edhe një bashkësi manipulatorësh të cilët mund të përdoren për të ndryshuar gjendjen e objekteve të iostream. Këto kontrollojnë mënyrën e formatizimit të të dhënave. Ato janë të definuara në fajllin <ios> që duhet të përfshihet përmes direktivës include. Zakonisht, nuk është e nevojshme të përfshihet në mënyrë eksplicite, pasi që në mënyrë indirekte është i përfshirë përmes librarive tjera, si <iostream>.

Manipulatorët dhe mënyrë e veprimit të tyre:

Manipulatori	Përdorimi
boolalpha	Bën, që variablat booleane të paraqiten si true ose false (tekstualisht).
noboolalpha (default)	Bën, që variablat booleane të paraqiten si 0 ose 1.
dec (default)	Specifikon se numrat (integer) paraqiten në bazën 10.
Hex	Specifikon se numrat (integer) paraqiten si heksadecimal (në bazën 16).
Oct	Specifikon se numrat (integer) paraqiten si oktal (në bazën 8).
Left	Teksti në fushën dalëse rreshtohet majtas .
Right	Teksti në fushën dalëse rreshtohet djathtas.
Internal	Bën që shenja e numrit të rreshtohet majtas kurse vlera djathtas.
noshobase (default)	Çkyqë paraqitjen e prefiksit që tregon bazën e numrit.
showbase	Kyqë paraqitjen e prefiksit që tregon bazën e numrit.
noshowpoint (default)	Paraqet pikën decimale vetëm nëse ekziston pjesa decimale.
showpoint	Paraqet pikën decimale gjithmonë.
noshowpos (default)	Nuk paraqet shenjën "+" si prefiks para numrave pozitiv.
showpos	Paraqet shenjën "+" si prefiks të numrave pozitiv.
skipes (default)	Kapërcen hapësirat e bardha (zbrazëtirat, tab, rreshti i ri) nga ana e operatorit të hyrjes >>.
noskipes	Hapësirat e bardha nuk kapërcehen nga operatori i nxjerrës, >>.
fixed (default)	Bën që numrat me presje dhjetore të paraqiten në notacion fiks (fixed).
scientific	Bën që numrat me presje dhjetore të paraqiten në notacion shkencor (scientific).
nouppercase (default)	0x paraqitet për numrat heksadecimal, e për notacionin shkencor.
uppercase	0x paraqitet për numrat heksadecimal, E për notacionin shkencor.

Manipulatorët e tabelës së mësipërme modifikojnë gjendjen e objektit iostream. Kjo do të thotë se, kur të përdoren njëherë në një objekt në iostream, ata do të kenë efekt në të gjitha

hyrjet dhe daljet vijuese të atij objekti. Ka disa manipulatore që përdoren për të modifikuar ndonjë dalje të veçantë, por nuk modifikojnë gjendjen e objektit.

#### Caktimi i gjerësisë së daljes (Output Width)

`setw(w)` – vendosë gjerësinë për dalje (output) ose hyrje (input) në ë; kërkon që të përfshihet me `include`.

`width(w)` – funksion i klasës `iostream` (member function).

#### Mbushja e hapësirave të bardha

`setfill(ch)` – mbushë hapësirat e bardha të daljes me karakterin `ch`; kërkon që të përfshihet.

`fill(ch)` – funksion i klasës `iostream` (member function).

#### Caktimi i precizitetit

`setprecision(n)` – vendosë paraqitjen e numrave me presje dhjetore me precizitet `n`. Kjo nuk ndikon mënyrën e trajtimit të numrave me presje dhjetore gjatë llogaritjeve në program.

#### Shembull:

```
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

int main()
{
    int intNumri = 15;

    cout << "Numer Integer" << endl;
    cout << "Standard: " << intNumri << endl;
    cout << "Octal: " << oct << intNumri << endl;
    cout << "Hex: " << hex << intNumri << endl;
    cout << "Kycja e paraqitjes se baze" << showbase << endl;
    cout << "Dec: " << dec << intNumri << endl;
    cout << "Octal: " << oct << intNumri << endl;
    cout << "Hex: " << hex << intNumri << endl;
    cout << "Ckycja e paraqitjes se baze" << noshowbase << endl;
    cout << endl;

    double doubleVlera = 12.345678;

    cout << "Numer me presje dhjetore" << endl;
    cout << "Default: " << doubleVlera << endl;
    cout << setprecision(10);
    cout << "Preciziteti 10: " << doubleVlera << endl;
    cout << scientific << "Notacioni shkencor: " << doubleVlera << endl;
    cout << uppercase;
    cout << "Shkronja e madhe E: " << doubleVlera << endl;
    cout << endl;

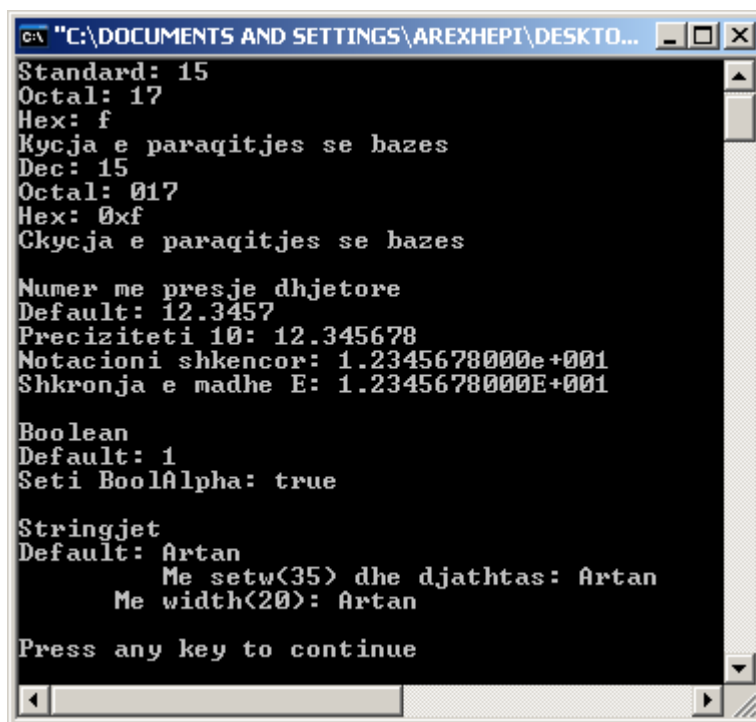
    bool vleraBooleane = true;

    cout << "Boolean" << endl;
    cout << "Default: " << vleraBooleane << endl;
    cout << boolalpha << "Seti BoolAlpha: " << vleraBooleane << endl;
    cout << endl;
```

```
string EmriIm = "Artan";

cout << "Stringjet" << endl;
cout << "Default: " << EmriIm << endl;
cout << setw(35) << right << "Me setw(35) dhe djathtas: "
    << EmriIm << endl;
cout << width(20);
cout << "Me width(20): " << EmriIm << endl;
cout << endl;

return 0;
}
```



```
C:\DOCUMENTS AND SETTINGS\AREXHEPI\DESKTO...
Standard: 15
Octal: 17
Hex: f
Kycja e paraqitjes se bazes
Dec: 15
Octal: 017
Hex: 0xf
Ckycja e paraqitjes se bazes

Numer me presje dhjetore
Default: 12.3457
Preciziteti 10: 12.345678
Notacioni shkencor: 1.23456780000e+001
Shkronja e madhe E: 1.23456780000E+001

Boolean
Default: 1
Seti BoolAlpha: true

Stringjet
Default: Artan
    Me setw(35) dhe djathtas: Artan
    Me width(20): Artan

Press any key to continue
```



## Referencat

Në C++ referenca është një pseudonim (nofkë) për një variabël. Dobia e përdorimit të referencave është gjatë përdorimit të përcjelljes së vlerave në funksione. Ato ofrojnë një mundësi për të kthyer vlerat prej funksioneve.

```
int vlera;           // Declaron një integer
int &rVlera = vlera; // Deklaron një referencë për në integer-in val
```

Para emrit të referencës përdoret simboli "&". Në këtë kontekst, "&" thirret si operatori i referencës. Kjo tregon se rVlera është referencë, e jo objekt i zakonshëm. Ndryshe, shenja "&" përdoret për të marrë adresën e objektit (tek pointerët). Në atë kontekst, "&" quhet operatori i adresës. Adresa e operatorit përdoret për të marrë adresën, zakonisht, për të inicializuar pointerin:

```
int vlera;           // Deklaron nje integer
int *pvlera = &vlera; // Deklaron pointerin dhe e inicializon atë me adresën e variables vlera.
```

Referenca nuk është objekt unik. Ajo është thjeshtë një pseudonim ose një sinonim për një objekt tjetër. Identifikatori i referencës (emri) mund të përdoret kudo ku mund të përdoret identifikatori i referuar. Çdo ndryshim i bërë në referencë do të aplikohet edhe në objektin origjinal. Gjithashtu, çdo ndryshim i bërë në objektin origjinal do të shihet edhe përmes referencës.

```
#include <iostream>
using namespace std;

int main()
{
    int vlera = 1;
    int &rVlera = vlera;

    cout << "vlera eshte " << vlera << endl;
    cout << "rVlera eshte " << rVlera << endl<<endl;

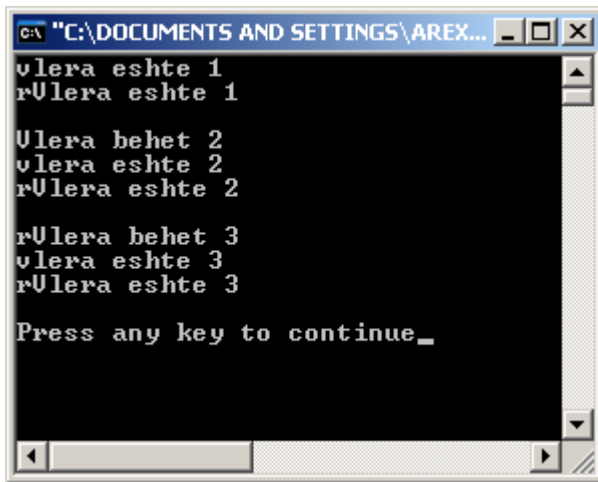
    cout << "Vlera behet 2" << endl;
    vlera = 2;

    cout << "vlera eshte " << vlera << endl;
    cout << "rVlera eshte " << rVlera << endl<<endl;

    cout << "rVlera behet 3" << endl;
    rVlera = 3;

    cout << "vlera eshte " << vlera << endl;
    cout << "rVlera eshte " << rVlera << endl<<endl;

    return 0;
}
```



Referenca duhet të inicializohet kur të krijohet. Nëse nuk vepohet kështu, do të paraqitet gabim në program.

Referenca nuk mund të ricaktohet. Nëse tentoni të ricaktoni referencën, rezultati do të jetë një caktim i vlerës së objektit origjinal të referuar.

#### Shembull:

```
#include <iostream>
using namespace std;

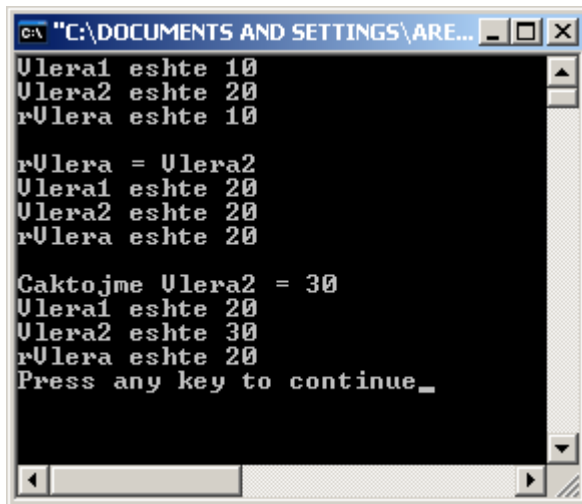
int main()
{
    int Vlera1 = 10;
    int Vlera2 = 20;
    int &rVlera = Vlera1;

    cout << "Vlera1 eshte " << Vlera1 << endl;
    cout << "Vlera2 eshte " << Vlera2 << endl;
    cout << "rVlera eshte " << rVlera << endl;

    rVlera = Vlera2;    //Çka ndodhe ketu?
    cout << endl << "rVlera = Vlera2" << endl;
    cout << "Vlera1 eshte " << Vlera1 << endl;
    cout << "Vlera2 eshte " << Vlera2 << endl;
    cout << "rVlera eshte " << rVlera << endl;

    Vlera2 = 30;
    cout << endl << "Caktojme Vlera2 = 30" << endl;
    cout << "Vlera1 eshte " << Vlera1 << endl;
    cout << "Vlera2 eshte " << Vlera2 << endl;
    cout << "rVlera eshte " << rVlera << endl;

    return 0;
}
```



```

C:\ "C:\DOCUMENTS AND SETTINGS\ARE... - [X]
Vlera1 eshte 10
Vlera2 eshte 20
rVlera eshte 10

rVlera = Vlera2
Vlera1 eshte 20
Vlera2 eshte 20
rVlera eshte 20

Caktojme Vlera2 = 30
Vlera1 eshte 20
Vlera2 eshte 30
rVlera eshte 20
Press any key to continue_

```

Ndarja e vlerës `rVlera = Vlera2`; ia ndanë vlerën e `Vlera2` `rVlera`-s. Pasi që gjithçka që veprohet në referencë aplikohet edhe në objektin e referuar, atëherë variablës `Vlera1` i është ndarë `vlera` e `Vlera2`, 20. nëse referenca do të ri-caktohet për të ju referuar variablës `Vlera2`, atëherë `Vlera1` nuk do të ndryshojë. Pastaj, `vlera` e `Vlera2` është ndryshuar në 30. Me këtë rast është ndryshuar vetëm `Vlera2`.

Referencat mund të bëhen edhe në objektet e definuara të shfrytëzuesit, njësoj si edhe në variablat e brendshme të C++.

```

#include <iostream>
using namespace std;

class Katrori
{
public:
    Katrori(int brinja): _brinja(brinja) {};
    int MerreBrinjen() {return _brinja;}
    void CaktoBrinjen(int brinja) {_brinja = brinja;}
    int Syprina() {return _brinja * _brinja;}
private:
    int _brinja;
};

int main()
{
    Katrori Katror(15);
    Katrori &refKatror = Katror;

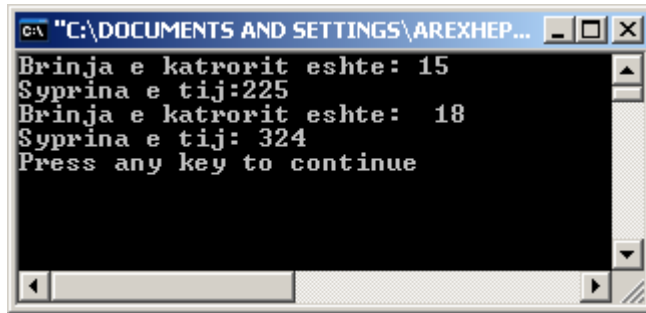
    cout<<"Brinja e katrorit eshte: "<<Katror.MerreBrinjen()<<endl;
    cout << "Syprina e tij:" << refKatror.Syprina() << endl;

    refKatror.CaktoBrinjen(18);

    cout<<"Brinja e katrorit eshte: "<<refKatror.MerreBrinjen()<<endl;
    cout << "Syprina e tij: " << Katror.Syprina() << endl;

    return 0;
}

```

A screenshot of a Windows command prompt window. The title bar reads "C:\ \DOCUMENTS AND SETTINGS\AREXHEP...". The window contains the following text:

```
Brinja e katrorit eshte: 15  
Syprina e tij:225  
Brinja e katrorit eshte: 18  
Syprina e tij: 324  
Press any key to continue
```

Pra, objekti Katror dhe referenca e tij refKatror mund të përdoren në këmbim. Ata janë sinonime për njëri tjetrin. Gjithashtu, vëreni se referenca mirret në objektin Katrori, e jo në klasën Katrori. Katror është një instancë e klasës Katrori.

Dobia më e madhe e referencave është kur ato përdoren në funksionet globale ose metoda të klasave. Në këtë rol, ato ofrojnë një mënyrë për kthimin e vlerave të shumëfishta prej funksioneve ose metodave pa përdorimin e pointerëve. Referencat në mënyrë strikte përdoren si pseudonime, si në shembujt e mëparshëm. Ato janë thjeshtë një identifikatorë tjetër (alternativ) për objektin e njëjtë.

## Përmbajtja:

Hyrje në programim .....	2
Interpreterët dhe kompajlerët .....	2
Hyrje në C++ .....	3
Variablat, tipet e tyre .....	8
Deklarimi .....	8
Deklarimi dhe inicializimi .....	9
Tipet, identifikatorët dhe fjalët e rezervuara .....	9
Identifikatorët dhe fjalët e rezervuara të C++ .....	12
Leximi dhe Shtypja .....	13
Rezervimi i hapësirës .....	13
Mbushja me mostër .....	14
Preciziteti .....	15
Sistemi numerik .....	17
Leximi i fjalëve dhe fjalive .....	18
Reshtimi i tekstit .....	20
Stringu .....	21
Kushtet (Degëzimet) .....	25
Bloqet e urdhërave në degë .....	28
Kushtet e shumëfishta dhe kushtet ndërthurura .....	29
Operatori “?” .....	31
Switch ( ) .....	32
Kapërcimi pa kusht: goto .....	35
Unazat (Loops - laqet) .....	38
Unaza while (gjersa) .....	38
Unaza do – while .....	39
Unaza for .....	41
Format e unazës for .....	43
Ndërprerja e unazës (break) .....	44
Kapërcimi i hapi të unazës (Continue) .....	45
Vargjet .....	47
Deklarimi dhe inicializimi i vargjeve .....	47
Vektorët .....	48
Matricat .....	52
Veprimet me vektorë/matrica dhe gjetja e anëtarëve të caktuar të vektorit/matricës .....	54
Anëtari më i madh (më i vogël) .....	57
Sortimi – radhitja sipas madhësisë .....	58
Formimi i fushave numerike .....	60
Bashkimi i dy matricave .....	66
Bashkimi i dy matricave – në diagonale .....	70
Krijimi i vektorit prej anëtarëve të matricës .....	72
Funksionet .....	80
Funksionet void .....	82
Inline funksionet .....	82
Shembuj funksionesh .....	83
Rekurzioni .....	91
Funksionet dhe Vektorët .....	92
Funksionet dhe matricat .....	95
Përfshirja e fajllave me direktivën #include .....	98
Thirrja e funksionit, nga fajlli tjetër .....	99
Direktiva #define .....	102

Shtojca A - Probleme praktike .....	104
Numrat e rastit në C++ .....	104
Llogaritjet kohore .....	110
Ngjyra e tekstit .....	113
Shtojca B - Pointerët .....	117
Dukshmëria e variablave .....	127
Shtojca C - Shkruarja dhe leximi i fajllave .....	128
Referencat .....	136

