

SHKOLLA E MESME KOMUNALE GJIMNAZI
“SAMI FRASHËRI” KUMANOVË

Nderim Rrahmani

GJUHËT PROGRAMUESE
VITI III

DREJTIMI: MATEMATIKO-NATYROR (KOMB A, B)

Shtator 2017

Përmbajtja

1. PROGRAMIMI DHE GJUHËT PROGRAMUESE	4
1.1. Algoritmet dhe programimi	4
1.2. Gjuhët programuese	5
1.2.1. Gjuhët e ulta programuese	5
1.2.2. Gjuhët e larta programuese	7
1.3. Tipet e gjuhëve programuese (paradigmat)	9
1.4. Ruajtja dhe organizimi i të dhënave në memorje	11
1.5. Paraqitja e numrave të plotë në kompjuter	12
2. C++ RRETHINA PËR ZHVILLIM	14
2.1. Struktura e një programi dhe elementet themelore në C++	14
2.2. Ndryshoret, tipet e ndryshoreve në C++	16
3. KLASAT DHE OBJEKTET NË C++	22
3.1. Koncepti i programimit të orientuar në objekte	22
3.1.1. Klasat dhe objektet	23
3.1.2. Trashëgimi	24
3.2. Forma e përgjithshme e klasës	26
3.2.1. Krijimi i metodave	27
3.2.2. Krijimi i objekteve dhe puna me objektet	28
3.3. Kontrollat e qasjes për atributet dhe metodat	30
3.4. Konstruktoret	31
3.5. Metodat Get dhe Set	33
3.6. Trashëgimi	35
4. PJESËT STRUKTURE DHE KODI NË POO	41
4.1. Strukturat për degëzim	41
4.1.1. Struktura për degëzim if-else	41
4.1.2. Struktura për degëzim switch	43
4.2. Strukturat për përsëritje	45
4.2.1. Struktura për përsëritje do-while	45
4.2.2. Struktura për përsëritje while	47
4.2.3. Struktura për përsëritje for	49
4.3. Urdhërat break dhe continue	49

5.	FUNKSIONET, FUNKSIONET ANËTARE TË KLASËS	52
5.1.	Funksionet standarde në C++	52
5.2.	Definimi i funksioneve jostandarde	54
5.3.	Ndryshoret lokale dhe globale	57
5.4.	Parametrat e funksionit sipas vlerës, referencës dhe adresës	59
5.4.1.	Kalimi i parametrave me reference	59
5.4.2.	Kalimi me parametrave me adresë	60
5.5.	Mbingarkimi i funksionieve	61
6.	VARGJET DHE VEKTORËT	62
6.1.	Kuptimi për vargun	62
6.1.1.	Inicializimi dhe deklarimi i vargut	63
6.1.2.	Vargu si parametër i funksionit	65
6.2.	Kompleksiteti kohor dhe memorues e algoritmeve	67
6.3.	Kërkimi në varg	69
6.3.1.	Kërkimi linear (Linear search)	69
6.3.2.	Kërkimi binar (Binary Search)	71
6.4.	Sortimi (rradhitja) e elementeve të vargut	72
6.5.	Krijimi i vargut me objekte	75
6.6.	Hyrje në C++ shabllonin vektor të librisë standarde	77
6.7.	Vargu tekstual	81
6.8.	Klasa string	86
6.8.1.	Operacionet me string	88
7.	MATRICAT	90
7.1.	Deklarimi dhe manipulimi me matrica	90
	LITERATURA	95

1. PROGRAMIMI DHE GJUHËT PROGRAMUESE

1.1. Algoritmet dhe programimi

Algoritmi paraqet grumbull të veprimeve me një radhë të fiksuar, të cilët ndërmerren gjatë zgjidhjes së një problemi të caktuar.

Shembuj të algoritmeve mund të jenë receta për gatimin e një ushqimi, dërgimi i një porsie përmes telefonit dhe shumë probleme tjera që hasen në përditshmëri.

E rëndësishme për algoritmet është të thuhet se ato mund të jenë **detal** ose të **përgjithshëm**, varësisht nga ajo se hapat e zgjidhjes jipen hollësisht apo jo.

Në rastin e programimit kompjuteri për të zgjidhur një problem i nevojitet “lista e përbërësve” ose të dhënat, të cilat mund të jenë të shprehura në disa forma si: numra, tekste, zë, fotografi të cilat janë të ruajtura në kompjuter që më vonë këta “përbërës” do ti quajmë ndryshore. Ndryshoret mund të përfaqësojnë të dhëna të cilat përpunohen dhe ruhen në kompjuter në forma të ndryshme si të dhëna numerike, tekst, zë ose fotografi.

Përpunimi i të dhënave në kompjuter bëhet përmes programeve të veçanta të cilët shkruhen nga programuesit. Çdo program mund të thuhet se është algoritëm i shëndruar në urdhëra.

Shembuj të programeve ose softuerit kompjuterik kemi të cekur paraprakisht ku mund të vërejmë se detyra e softuerit është përpunimi, ruajtja, bartja i të dhënave të ndryshme në bazë të rregullave të caktuara paraprakisht ose të dhëna nga shfrytëzuesi.

Shkurtimisht mund të definohet si:

Një program kompjuterik ose program, paraqet listë të instruksioneve e shkruar për ta “kuptuar” dhe egzekutuar kompjuteri që të kryej detyrë të caktuar.

Programimi paraqet procesin zhvillimit dhe implementimit të bashkësive të ndryshme të instruksioneve që kompjuteri të kryej detyrë të caktuar. Ose shkurtimisht programimi paraqet procesin e shkruarjes së programit.

1.2. Gjuhët programuese

Programi kompjuterik është bashkësi e urdhërave të cilat egzekutohen në kompjuter.

Urdhërat të cilët i jipen kompjuterit shkruhen duke u bazuar rregullave të caktuara të cilat kompjuteri për ti egzekutuar duhet që ti “kuptojë”. Bashkësia e rregullave të cilat mundësojnë përpilimin e urdhërave të kuptueshëm për sistemin kompjuterik paraqet gjuhën programuese.

Gjuha programuese është gjuhë e cila përmban një bashkësi të rregullave me anë të cilave i urdhërojmë kompjuterët të kryejnë detyra të caktuara. Qëllimi i gjuhës programuese është që makinës llogaritëse (kompjuterit) ti jepen instruksionet të shkruara të cilat do të egzekutohen për të kontrolluar sjelljen e kompjuterit ose të shprehin algoritme.

Një gjuhë programuese mundëson që të përpilohen programe për kompjuter, të cilat mundësojnë që kompjuteri të kryej operacione llogaritëse, kontrollojë hyrjen dhe daljen e të dhënave, kontrollojë pajisjet e kompjuterit si printer, hard disk, dhe shumë funksione tjera të dobishme për shfrytëzuesit e sistemit kompjuterik.

Egzistojnë mijëra gjuhë të ndryshme programuese të krijuara të cilat mundësojnë krijimin e aplikacioneve të shumëllojshme.

Të gjitha shembujt e cekur të softuerit sistemor dhe aplikativ janë të krijuara me anë të gjuhëve programuese.

Të dhënat të cilat përpunohen dhe ruhen në kompjuter ruhen me shifrat “0” dhe “1”, do të thotë se kompjuteri gjatë përpunimit të të dhënave njih vetëm dy simbole. Gjithashtu instruksionet në nivelin më të ulët jipen vetëm me shifrat “0” dhe “1”. Kombinimi i tyre mundëson përpilimin e urdhërave dhe egzekutimin e operacioneve komplekse me të dhëna të cilat konvertohen në formë të kuptueshme për shfrytëzuesin e sistemit kompjuterik.

Varësisht nga ajo se urdhërat e shkruara janë një gjuhë e afërt me parimin e punës së kompjuterit apo të njeriut dallojmë këto lloje të gjuhëve programuese.

- Gjuhë të ulta programuese
- Gjuhë të larta programuese

1.2.1. Gjuhët e ulta programuese

Karakteristikë të gjuhët e ulëta programuese është se instruksionet në gjuhët e ulëta programues jipen drejtëpërdrejtë në processor, kanë qasje të drejtëpërdrejtë tek

regjistrat e procesorit dhe memorja, programet e shkruara në këtë gjuhë posedojnë performanca të shkëlqyera gjatë egzekutimit .

Gjuha makinës është përfaqësues tipik dhe kryesor i gjuhëve të ulëta programuese. Urdhërat në gjuhën e makinës përbëhen prej bitave (binary digits), 1 dhe 0 dhe të cilën në fakt e "kupton" procesori i kompjuterit dhe me anë të cilës në fillimin e zhvillimit të kompjuterëve janë shkruar programet e para. Çdo urdhër i programit është shkruar vetëm me ndihmën e shifrave binare.

Shembull: Të llogaritet shuma e numrave **a** dhe **b** dhe te ruhet në **c**.

Kompjuteri duhet të njehsojë shumën e dy numrave të ruajtur në memorjen e tij të "emërtuara" si **a** dhe **b** dhe të llogarisë shumën e tyre në memorjen e rezervuar **c**. Kompjuteri për çdo adresim ose urdhër përdorë numrat binar.

Nëse adresa e rezervuar për **a** është 111001, për **b** është 101011, për **c** është 100011 operatori për mbledhje (+) duhet të shkruhet si 111101 urdhëri për lexim të ndryshores është 111000, urdhëri për ruajtjen e të dhënës në një adresë është 000111 atëherë, pjesë nga programi mund të shkruhej si:

```
111000111001
000111 100011
111000101011
111101 101011100011;
```

"Programi" i dhënë i cili cakton shumën e dy numrave dhe e rezultatin e fituar e ruan në një lokacin të memorjes është duket i pakuptueshëm. Gjithashtu, pasi që gjuha e makinës është e vështirë për t'u kuptuar dhe zbatuar, dhe është e vështirë që komandat (udhezimet) të mbahen mend (çdo udhëzim duhet të shënohet me anë të numrave 1 dhe 0), janë krijuar gjuhët programuese të cilat në fakt janë përkthim i urdherave me bita (1 dhe 0) që mundet të mbahet në mend lehtë dhe me anë të së cilës mund të programohet.

Kjo mënyrë e programimit ka qenë shumë e vështirë prandaj inxhinierët kanë përsosur mënyrën e programimi duke krijuar gjuhën assembler.

Gjuha assembler i takon gjuhëve të nivelit të ulët të gjuhëve programuese për një kompjuter ose pajisje të tjera programueshëm, në të cilën ka një korrespondence (lidhje) shumë të fortë (përgjithësisht një-për-një) midis gjuhës dhe arkitekturës së kodit të makinës.

Kodi i gjuhës assembler konvertohet në kod të ekzekutues të makinës nga programi i njohur si assembler; Urdhërat e shkruara në gjuhën assembler janë më të afërta me gjuhën e njeriut, dhe mundëson që programuesi për të shkruar urdhëra dhe operacione përdorë shprehje nga gjuha angleze, gjithashtu mundëson përdorimin e ndryshoreve.

Shembull: Të llogaritet shuma e dy vlerave **a** dhe **b** dhe rezultati të ruhet në memorjen e rezervuar **c**.

```
mov ax,a      ;vendose vlerën e ruajtur num1 në ax
add ax,b      ;cakto shumën e dy numrave
mov c,ax      ;vendose shumën e dy numrave në ndryshoren sum
```

Gjuha assembler urdhërat i egzekuton sipas radhës ashtu edhe si janë shkruar ku urdhëri i parë:

`mov ax, a` mundëson që vlerën e ruajtur në lokacionin **a** të vendoset në adresën e memorjes e quajtur **ax**

urdhërin e dytë `add ax,b` mundëson që vlerës **ax** i shtohet vlera e ruajtur në **b**.

Me urdhërin e tretë `mov c,ax` mundësohet që në memorjen **c** të ruhet vlera e **ax**.

Me tre urdhërat arrihet që të caktohet shuma e numrave të ruajtur në **a** dhe **b** dhe shuma e tyre të ruhet në memorjen **c**.

Dallimi mes gjuhës së makinës dhe gjuhës assembler është i qartë, gjuha assembler është më e afërt me gjuhën natyrore (gjuhën angleze), më e lehtë për ta kuptuar për njeriun. Më këtë edhe më lehtë shkruhen programet dhe më lehtë kuptohen programet e shkruara më parë.

1.2.2. Gjuhët e larta programuese

Gjuhët programuese mund të përdoren për të krijuar programe që kontrollojnë sjelljen e një dhe makinë apo për të shprehur algoritme në mënyrë precize.

Urdhërat në gjuhët e larta programuese nuk përshkruajnë detalisht instruksinet që duhet të egzekutojë kompjuteri sikurse në rastin e gjuhëve të ulëta programuese por jipen në nivel më të lartë, në të njëjtën kohë duke qenë më i afërt me gjuhën e njeriut.

Gjuhët e larta programuese janë më të lehta për të programuar dhe kuptuar programi i shkruar pasi që përdorin shprehja nga gjuha angleze, sikurse shprehjet if, for, Begin, end; operacionet aritmetikore u janë përshtatur me ato që përdorim në praktike, +,-,*,<,>, DIV,MOD, dhe shprehje tjera të afërta me gjuhët natyrore të njeriut.

Shembull: Të caktohet shuma e vlerave **a** dhe **b** dhe rezultati të ruhet në ndrrshoren **c**.

Urdhëri që do ti jipet gjuhës së lartë programuese do të duket si në vijim

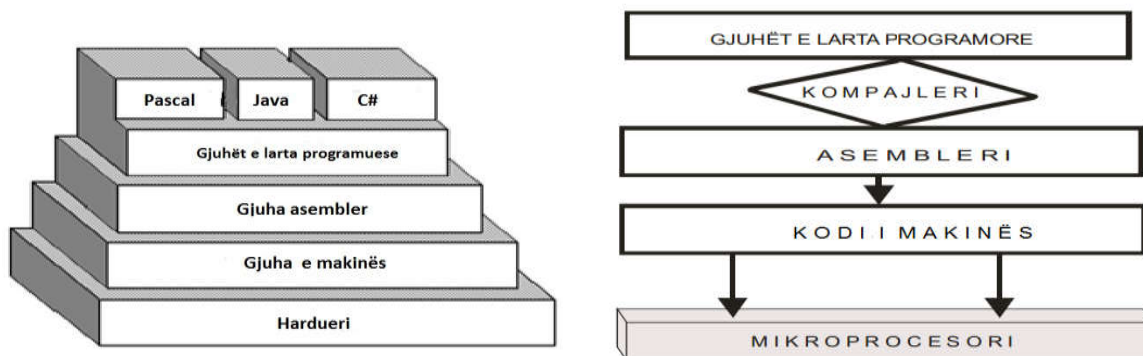
```
c=a+b;
```

Dallimi mes gjuhës assembler dhe gjuhës së lartë programuese shembujt e mësipërm e bëjnë të qartë. Edhe pse rezultati është i njëjtë në të dy rastet, urdhërat në gjuhën e lartë programuese janë më të afërta me gjuhën natyrore të njeriut dhe më të lehta për tu shkruar dhe kuptuar nga programuesit.

Sot, janë një numër i konsiderueshëm i gjuhëve programore. Ndër gjuhët më të njohura janë: C++, Java, Pascal, Delphi, C#, php etj...

Përparësitë e gjuhëve të larta programuese janë:

- Urdhërat lehtë shkruhen, kuptohen
- Egzekutimi nuk varet nga lloji i procesorit
- Përfshijnë domen të gjërë të të dhënave



Hierarkia e gjuhëve programuese

Në disa tekste mund të haset se gjuhët programuese ndahen në tri kategori (gjuhë e ulëta , gjuhët e mesme dhe gjuhët e larta programuese)

Shkurtimisht

- **Algoritmi paraqet grumbull të veprimeve me një radhë të fiksuar, të cilët ndërmerren gjatë zgjidhjes së një problemi të caktuar. Algoritmi mund të jetë detal ose i përgjithshëm**
- **Programi kompjuterik paraqet listë të instruksioneve e shkruar për ta “kuptuar” dhe egzekutuar kompjuteri që të kryej detyrë të caktuar.**
- **Programimi paraqet procesin zhvillimit dhe implementimit të bashkësive të ndryshme të instruksioneve që kompjuteri të kryej detyrë të caktuar.**
- **Gjuhët e programimit përdoret për të krijuar programe që kontrollojnë sjelljen e një makine (kompjuteri) dhe/ose të shprehin algoritme të përshkruar saktësisht.**
- **Gjuhët programuese ndahen në gjuhë të larta dhe gjuhë të ulëta programuese**
- **Gjuhët e ulëta programuese janë të përafërta me gjuhën e kompjuterit dhe janë më pak të kuptueshme për njeriun, shprehja e algoritmeve me këto gjuhë është më e vështirë**
- **Gjuhët e larta programuese janë më të afërta me gjuhët natyrore, më të kuptueshme për njeriun , shprehja e algoritmeve është më e lehtë.**

1.3. Tipet e gjuhëve programuese (paradigmat)

Paradigma programore paraqet stilin thelbësor të programimit kompjuterik. Paradigma dallon nga metodologjia nga se metodologjia paraqet stilin për zgjidhjen e një problemi specifik në inxhinieringun softuerik. Paradigmat dallojnë nga njëra tjetra për nga konceptet dhe abstraksionet që i shfrytëzojnë për paraqitjen e elementeve të programit (si p.sh. objektet, funksionet, variablat, etj.) si dhe dallojnë për nga hapat që i marrin për përpilimin e një llogaritjeje (p.sh. vlerësimi, rrjedhshmëria e të dhënave, etj.) Një gjuhë programuese mund të përkrah paradigma të shumëfishtë. Për shembull, programet që shkruhen në C++ ose Object Pascal mund të jenë vetëm procedurale, ose vetëm të orientuara në objekte, mirëpo këto programe mund të përmbajnë edhe elemente të dy paradigmave. Se cilat elemente të paradigmave përfshihen në program varet tërësisht nga dizajnerët e softuerit.

Disa nga paradigmat e programimit janë:

Programimi imperative, paradigme programore e cila shfrytëzon **urdhëra** të cilat ndryshojnë gjendjen e programit. Programet imperative përbëhen nga urdhëra të cilat i egzekuton kompjuteri. Programimi imperativ përshkruan se si operon programi. Shembull i gjuhës programuese imperative janë gjuhët programuese C++, java, C#.

Programimi deklarativ , paradigm e programimit mënyrë e cila ndërton struktura dhe elemente programore, e cila shpreh logjikën e llogaritjes pa përkthyer kontrollin e rrjedhës së urdhërave. Shembull I I gjuhës programuese declarative janë gjuhët Prolog, Lisp.

Programimi procedural, paradigmë e nxjerrë nga programimi structural, e bazuar në thirrjen e procedurës. Procedurat njihen edhe si rutina, nënprograme ose funksione (jo identike me funksionet matematike) , të cilat përmbajnë varg të instruksioneve. Shembull I gjuhës programuese procedurale janë gjuhët Pascal, C, C++, Java, C#.

Programimi I orientuar në objekte, paradigm e programimit e bazuar në konceptin e “objektit”, I cili përmban të dhëna, në formë të fushave njihen edhe si atribut, dhe kod, në formë të procedurës, i njohur edhe si metodë. Vetë e objektit është qasja e procedurës deri te vlerat e fushave që I përmban objekti. Shembull I gjuhëve të orientuara në objekte janë C++, Java, C#, Smalltalk, php.

Programimi funksional (Functional programming) paradigm- stil I ndërtimit të strukturave dhe elementeve të programeve kompjuterike e cila llogaritjen e konsideron si njësim I funksioneve matematike dhe nuk preferon ndryshimin e gjendjes dhe ndryshimin e të dhënave. Është paradigë deklarative që tregon se programimi bëhet përmes shprehjeve ose deklarimeve në vend të urdhërave. Në kodin funksional vlera dalëse e funksionit varet nga vlerat hyrëse të funksionit. Shembull I gjuhëve funksionale janë Common Lisp, Scheme, Haskell etj.

Programimi logjik paradigm e bazuar në logjikën formale. Programi prëbëhet nga bashkësi të urdhërave në formë logjike, duke shprehur fakte dhe rregulla në domenin e një problemi. Shembull i gjuhëve logjike përfshin Prolog, ASP dhe Datalog.

Në **Programimin e Orientuar në Objekte**, programorët mund ta perceptojnë paradigmen si një koleksion të objekteve ndërvepruese; përderisa në **Programimin Funksional**, programi mundet të perceptohet si një sekuencë të funksioneve vlerësuese. Disa gjuhë programuese përkrahin një paradigëmë të vetme, si **Smalltalk** që përkrah vetëm *Programimin e Orientuar në Objekte*, ose **Haskell** që përkrah vetëm *Programimin Funksional*. Mirëpo, ekziston një numër i gjuhëve programuese që përkrahin paradigma të shumta, si p.sh. **Object Pascal, C++, C#, Visual Basic, Common Lisp, Scheme, Python, Ruby** dhe **Oz**.

1.4. Ruajtja dhe organizimi i të dhënave në memorje

Elementi themelor prej të cilit janë ndërtuar pjesët elektronike të kompjuterëve është transistori i cili mund të jetë në dy gjendje – të kyçur ose të shkyçur. Këto dy gjendje korrespondojnë

me shifrat binare 1 dhe 0. Me 0 shënohet se te rrjedhja e rrymës, deri sa me 1 shënohet se ka rrymë

te elementi. Për këtë veti të transistorëve, të gjitha të dhënat dhe instruksionet te kompjuteri janë paraqitur me shënime të përbëra prej zerove dhe njësheve (shënime binare). Sistemi numerik i cili për bazë i ka këto shifra quhet *sistem numerik binar*, por shifrat 0 dhe 1 quhen *shifra binare*.

Memoria përbëhet prej qelizave, në një qelizë mund të ruhet vetëm një shifër binare. Kjo është sasia më e vogël e informatave e cila mund të mbahet mend te memoria dhe quhet **bit** (bit

= binary digit). Biti mund të ketë vlerë 0 ose 1. Bitet janë grupuar në varg prej 8 biteve të cilat quhen **bajt** (byte). Me një bajt të paraqitet një shifër, një shkronjë ose një shenjë. Biti shënohet me b, bajti me B.

Vargu prej 4 bitësh quhet gjysëmbajt (ang. nibble).

Vargu prej 16 bitësh quhet fjalë (ang word)

0,1 –bit

00110011- bajt (8 bita)

1100 gjysëmbajt (4 nibble)

00110011 00110011 fjalë (16 bit word)

Te memoria e brendshme ekzistojnë lokacione të ndryshme për ruajtje të llojeve të ndryshme të të dhënave dhe instruksioneve. Çdo e dhënë te memoria ka vetëm adresë të veçantë me të cilën mundëson shfrytëzimin e të dhënave të dëshiruara. Në këtë mënyrë, procesori mund të ketë qasje dhe të ndërmerret të dhënë të nevojshme. Kur te ndonjë lokacion memorik (adresë) do të shkruhet ndonjë përmbajtje, përmbajtja e tij paraprake humbet. Kapaciteti i memories shprehet me bajt, përkatësisht me njësi më të mëdha – kilobajt (1KB = 1024 B), megabajt (1MB = 1024KB), gigabajt (1 GB = 1024MB), terabajt (1 TB = 1024 GB) etj.

1.5. Paraqitja e numrave të plotë në kompjuter

Të gjitha të dhënat instruksionet në memorjen e kompjuterit ruhen në formë binare do të thotë në 0 dhe 1. Të dhënat numerike ruhen dhe organizohen në formë që janë më lehtë për tu shprehur në gjuhën e makinës për ekzekutimin e operacioneve matematikore dhe për ruajtjen e tyre në memorjen e kompjuterit.

Paraqitja e numrave të plotë në kompjuter bëhet si në figurën vijuese:

S	0	1	0	.	.	.	1	0
---	---	---	---	---	---	---	---	---

Biti i parë tregon shenjën e numrit nëse $S=0$ numri është pozitiv nëse $S=1$ numri është negativ. Numrat pozitiv paraqiten ashtu që numri dekad konvertohet në binar dhe dhe biti i parë është 0.

Numrat negativ fitohen nga pozitiv duke gjetur komplementin e parë dhe të dytë të numrit të kundërt pozitiv.

Komplementi i parë i një numri binar fitohet ashtu që 0 shëndrrohen në 1-she dhe 1-she konvertohen në 0.

$$00110100_2 \rightarrow 11001011_2$$

Komplementi i dytë fitohet ashtu që komplementi i parë rritet për 1.

$$11001011_2 + 1_2 = 11001100_2$$

$$00110100_2 = 52_{10}$$

$$11001100_2 = -52_{10}$$

n- numri i bitave

shembull:

010101= shifra e parë 0 tregon se numri është pozitiv, ndërsa $10101_2 = 21_{10}$

Shembull për $n=8$;

Numri më i madh pozitiv është $2^{n-1}-1$

$$2^{8-1}-1=2^7-1=127$$

7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1

Numri më i vogël negativ -2^{n-1}

$$-2^{8-1} = -2^7 = -128$$

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0

Zero

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

2. C++ RRETHINA PËR ZHVILLIM

2.1. Struktura e një programi dhe elementet themelore në C++

Për ta kuptuar strukturën e një programi në C++ të analizojmë programin e dhënë

```
// Programi i pare ne C++
#include <iostream>
using namespace std;
int main ()
{
    cout << "Miresevini ne programim ";
    return 0;
}
```

nëse analizojmë çdo vijë nga kodi programor do të mund të kemi:

```
// Programi i pare ne C++
```

paraqet komentim te kodit burimor përdoret për të qartësuar urdhërat e programit dhe nuk analizohet nga kompjaluesi, gjithashtu teksti mes shenjave /* dhe */ njihet si koment nga gjuha programuese C++

```
#include <iostream>
```

urdhër paraprosesorik i cili mundëson shfrytëzimin e urdhërave për komunikim me ekranin dhe tastierën gjegjësisht paraqitjen e ndryshoreve dhe konstave tekstuale në ekran dhe leximin e vlerave nga tastiera. Keto urdhëra janë pjesë e biblioteks së funksioneve të gatshme të C++. Urdhërat paraprosesorik interpretohen para kompajlimit të programit dhe shkruhen pas shenjës (#)

```
using namespace std;
```

Ky urdhër mundëson që urdhërat **cin** dhe **cout** të përdoren pa shkruar std:: para tyre, pasi që urdhërat cin dhe cout janë pjesë e domenit (namespace) **std**.

```
int main ()
```

Funksioni kryesor nga fillon egzekutimi i programit. int tipi kthyes i funksionit (int =numër i plotë)

Kllapa e madhe e hapur ({}) tregon fillimin e një blloku programor nëshembullin e dhënë tregon fillimin funksionit kryesor main, ndërsa kllapa e madhe e mbyllur (}) tregon fundin e bllokut programor në shembullin e dhënë fundin e funksionit kryesor.

```
    cout << "Miresevini ne programim ";
```

urdhëri cout përdoret për të paraqitur tekstin në thonjëza në ekran.

```
    return 0;
```

vlera kthyesë e funksionit main.

Alfabeti i gjuhës C++ (Bashkësia e simboleve të lejuara)		
Simbole alfa-numerike	Shkronja të vogla:	a b c ...z
	Shkronja të mëdha:	A B C ...Z
	Shifra	0 1 2 3 4 5 6 7 8 9
Simbole speciale	~ ! @ # \$ % ^ & * () - + / = { } [] , . : ; ' " < > ? ®	
Simbole të padukshme (separatorë)	Vend i zbrazët (blank)	
	Rresht i ri (new line)	
	Vende të zbrazëta (tab)	

Fjalët e gjuhës C++

1. Fjalë të rezervuara - emërtojnë llojet (tipet) e të dhënave, komandat e strukturave kontrolluese (int, float, for, if, etj)
2. Identifikatorë - përdoren për emërtimin e konstanteve, ndryshoreve, funksioneve dhe strukturave të tjera në program (a,b,x,y, shuma,numri,mosha, etj)
3. Operatorë përdoren për operacione të ndryshme me të dhënat (+, -, *, /, %, >, >=, <>, ||, etj)

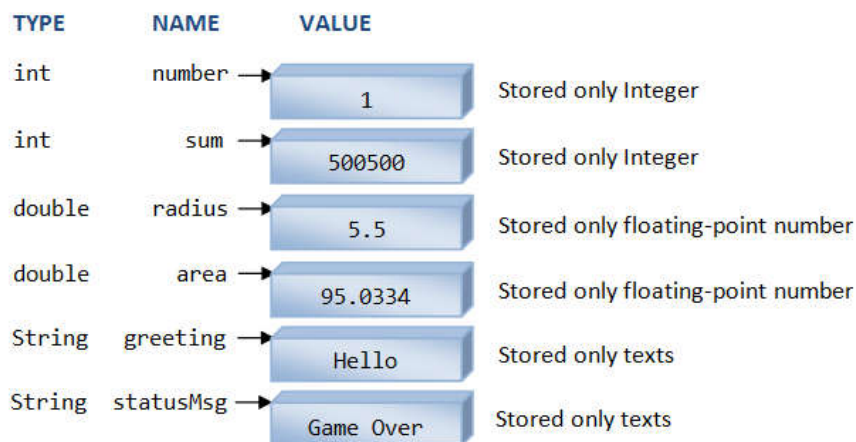
Fjalë të rezervuara të gjuhës C++				
auto	do	int	short	union
break	double	long	signed	unsigned
case	else	mutable	sizeof	virtual
catch	enum	new	static	void
char	extern	operator	struct	volatile
class	float	private	switch	while
const	for	protected	template	
continue	friend	public	this	
default	goto	register	throw	
delete	if	return	typedef	

RREGULLAT PËR EMËRTIMIN E IDENTIFIKATORËVE:

- Mund të përmbajnë shkronja (a..z, A..Z), shifrat (0..9) dhe nënvizë (_)
- Nuk mund të përmbajnë simbole speciale (\$, @, %, etj)
- Mund të fillojnë me shkronjë ose nënvijë, por nuk mund të fillojnë me shifra

2.2. Ndryshoret, tipet e ndryshoreve në C++

Ruajtja e të dhënave nga gjuhët programuese bëhet në memorje të rezervuar më parë ose në mënyrë dinamike gjatë proçit të egzekutimit të programit. Vlera e ruajtur në një adresë të caktuar të memorjes mund ndryshojë gjatë egzekutimit të programit. Element i rëndësishëm i gjuhës programuese C++ janë **ndryshoret**. **Ndryshoret paraqesin emra të cilat iu jipen pjeseve të vecanta të memorjes ashtuqë programi të manipulojë me vlerën e ruajtur në atë pjesë të memorjes.**



A variable has a **name**, stores a **value** of the declared **type**.

Fig 41. Kuptimi për ndryshoren

Gjuha programuese C++ përpunon disa lloje të ndryhsoreve, disa nga të cilat janë:

Emërtimi	Përshkrimi	Rangu i vlerave të ndryshores , shembuj
Bool	Tip logjik merr vlerat 1 (e vertete), 0 (e pavertete).	0(false) 1(true)
Char	Karakter	'a', 'b', '3', '\$',...
short	Numër i plotë	{-32768 deri në 32767}
Int	Numër i plotë.	{-2147483648 deri në 2147483647}
long int (long)	Numër i plotë	{-2147483648 deri në 2147483647}
long long int (long long)	Numë i plotë	-9,223,372,036,854,775,807 deri në 9,223,372,036,854,775,807
unsigned int	Numër i plotë pozitiv	{0 deri në 4294967295}
Float	Numër dhjetor	+/- 3.4e +/- 38 (~7 shifra të sakta pas presjes dhjetore)
Double	Numër dhjetor precizitet i dyfishtë.	+/- 1.7e +/- 308 (~15 shifra të sakta pas presjes dhjetore)
string	Varg karakteresh (tekst)	"Teknologjia e informacionit", "abc234",....

Deklarimi i ndryshoreve;

```
[TipiTeDhenes] [EmriINdryshores];
bool t; //Deklarohet ndryshore e tipit logjik (bool -tipi i te
dhenes, t- emri i ndryshores)
bool a,b; //Deklarohen dy ndryshore te tipit logjik
int n,m; //deklarohen dy ndryshore te tipit numer i plote
float f; //deklarohet ndryshore e tipit numer dhjetor
char shenja;
string emri, mbiemri;
```

Dhënia e vlerës për ndryshoren;

```
a=true; // ndryshorja a merr vleren true (e vertete)
shenja='$';
emri="Endrit";
mbiemri="Islami";
n=100; //ndryshoraj n merr vleren 100
m=n+50; //ndryshorja m merr vleren (150)
f=34.54;
```

sintaksa e përgjithshme e urdhërit për dhënien e vlerës është

ndryshorja=shprehja;

Ndryshoret gjithashtu mund të inicializohen edhe përmes tastierës përmes urdhërit:

cin>>ndryshorja;

Paraqitja në ekran e ndryshoreve bëhet me urdhërin cout;

cout<<ndryshorja;

Nëse kemi më shumë ndryshore atëherë ndahen përmes operatorit <<;

cout<<ndryshorja1<<ndryshorja2<<ndryshorja3;

Gjithashtu mund të kombinohen konstanta tekstuale dhe ndryshore:

```
cout<<"vlera e ndryshores x " <<x<<" vlera e ndryshores y: " <<y;
```

Shembull: Te deklarohen, jipen vlerat përmes tastierës dhe paraqiten në ekran ndryshoret në të cilat ruhen të dhënat për nxënësin: emri, mbiemri, mosha, gjinia dhe statusi (i rregullt/çrregullt):

```
#include<iostream>
using namespace std;
int main()
{
    string emri, mbiemri;
    char gjinia;
    int mosha;
    bool iRregullt;
    cout<<"Shkruaj të dhënat për nxënësin:"<<endl;
    cout<<"Emri: ";
    cin>>emri;
    cout<<"Mbiemri: ";
    cin>>mbiemri;
    cout<<"gjinia:";
    cin>>gjinia;
    cout<<"mosha:";
    cin>>mosha;
```

```

    cout<<"statusi (1=i rregullt 0 i crregullt): ";
    cin>>iRregullt;
    cout<<"-----\n-----\n";
    cout<<"keni futur te dhenat per "<<endl;
    cout<<"Nxenesi "<<emri<<" "<<mbiemri<<" gjinia "<<gjinia<<" mosha
    "<<mosha<<endl;
    cout<<"statusi (1=i rregullt 0 i crregullt) "<<iRregullt;

    return 0;
}

```

Operacionet me te dhena;

Operacionet me variablat e tipit logjik;

Operacione DHE (&&) a && b ;

Operatori OSE (||) a || b ;

Operatori Negacion (!) !a;

Për të njehësuar shprehje më më shumë se dy operatorë logjik shfrytëzohet prioriteti i operatorëve dhe kllapat, sikurse në matematikë. Pavarësisht prioritetit nëse shprehja përdor kllapat (), atëherë prioriteti i operatorëve nuk konsiderohe gjegjësisht së pari njehsohet shprehja në kllapa, pastaj vlen rregullat sipas prioritetit. Prioriteti paraqitet në tabelën në vazhdim.

Prioritet i lartë	!
	&&
Prioritet i ulët	

Tabela e prioritetit të operatorëve logjik

Shembull;

```

#include<iostream>
using namespace std;
int main()
{
    bool p,q,r;
    p=true;
    q=false;
    cout<<"NEGACION "<<p<<" = "<<!p<<endl;
    r=p&&q;
    cout<<p<<" DHE "<<q<<" = "<<r<<endl;
    r=p||q;
    cout<<p<<" OSE "<<q<<" = "<<r;
    return 0;
}

```

Shembull: Të llogaritet vlera e shprehjes: për p=true, q=false, r=true;

- a) `p && q || (!q && r)`
- b) `!p || q || !(q && r) || !p`
- c) `p && !(q && p) || !r`

Operacionet binare aritmetike

Simboli	Pershkrimi
+	Mbledhja
-	Zbritja
*	Shumezimi
/	Pjestimi
%	Mbetja gjatë pjestimit

Për të njehësuar shprehje më më shumë se dy operatorë logjik shfrytëzohet prioriteti i operatorëve dhe kllapat, sikurse në matematikë. Pavarësisht prioritetit nëse shprehja përdor kllapat (), atëherë prioriteti i operatorëve nuk konsiderohe gjegjësisht së pari njehsohet shprehja në kllapa, pastaj vlen rregullat sipas prioritetit. Prioriteti paraqitet në tabelën në vazhdim.

Prioriteti lartë	*, /, %
Prioritet i ulët	+, -

Tabela e prioritetit të operatorëve aritmetik

```
int a,b=5,c=3,d;
a=b+c;
c=a/b;
d=a*b/2-b%c;
etj..
```

Operacionet unare;

C++ mundëson disa shprehje të shkruhen shkurtimisht.

Shprehja	Është ekuivalent me
<code>m += x;</code>	<code>m = m + x;</code>
<code>a -= 5;</code>	<code>a = a - 5;</code>
<code>a /= b;</code>	<code>a = a / b;</code>
<code>x*= numri + 1;</code>	<code>x= x * (numri + 1); te perfshihe ne inicializim</code>

Shto për një dhe zvogëlo për një:

```
int a,b, c=7,d=4;
```

```

c++; //c=c+1;
c--; //c=c-1;
++d; //d=d+1;
--d; //d=d-1;
a=c++; //eshte ekuivalent me a=c; c=c+1;
a=++c; // eshte ekuivalent me c=c+1; a=c;
#define pi 3.14253

```

Operatorët relacional mundësojnë krahasimin e shprehjeve ku vlera kthyesë e tyre është e saktë (1) ose e pasaktë (0). Sintaksa e përdorimit të operatorëve relacional është:

shprehja1 OPERATORI shprehja2

Simboli	Pershkrimi
==	i barabartë
!=	Jo i barabartë
>	Më i madh se
<	Më i vogël se
>=	Më i madh ose i barabartë me
<=	Më i vogël ose i barabartë me

Tabela e operatorëve për krahasim

2==1+6

3+4!=3+3

3+4<=3+3

Ushtrime

- Cili është tipi dhe vlera e rezultatit të shprehjeve;
 - 25 % 4*3
 - 5*(2+9/4)
 - 12+(35*2+3/2)
 - 36%7/3*2
- Të llogaritet vlera e shprehjes nëse x=12, y=4 DHE z=12
 - x- y/(3%x)
 - x*y%5+4
 - 3*x+z-y/3.0
- Të shkruhet programi në C++ i cili njehson:
 - Sipërfaqen e rrethit $S=r^2\pi$
 - Perimetrin e rrethit $P= 2r\pi$
 - Sipërfaqen e trekëndëshit nëse janë dhënë gjatësitë e brinjëve të tij përmes formulës së Heronit: $S_{\text{perfaqja}}=\sqrt{s(s-a)(s-b)(s-c)}$ ku $s=\frac{a+b+c}{2}$

4. Të shkruhen në gjuhën programues C++ shprehjet aritmetike ekuivalente me shprehjet e dhëna:
 - a) $A = \left(\frac{x+4}{3-2y}\right)\left(\frac{2+x}{3}\right)$
 - b) $B = \frac{(a-b)}{c} + \frac{c(3+b)}{3a}$
 - c) $C = \frac{2*(x-3)}{y-\frac{x}{3}} * \frac{3x}{x-3y}$
5. Të zgjidhet
 - a. barazimi linear $ax+b=c$
 - b. jobarazimi linear $ax+b>c$
 - c. jobarazimi linear $ax+b<c$
6. Të caktohet çmimi i mallit nëse është dhënë çmimi bazë përqindja dhe sasia e mallit (shembull, çmimi bazë për një kompjuter personal është dhënë 12 000 den, përqindja e tatimit 5% dhe sasia kompjuterëve (numri i tyre) 25 kompjuterë)
7. Koha e dhënë në sekonda: të shprehet në orë, minuta dhe sekonda
8. Të gjendet shuma e shifrave të numrit treshifror.

3. KLASAT DHE OBJEKTET NË C++

3.1. Koncepti i programimit të orientuar në objekte

Para se të thellohem në terminologjinë OO, të shqyrtojmë pyetjen: **Çfarë është, në të vërtetë, objekti?**

Shumica e njerëzve në jetën e përditshme mendojnë nëpërmjet objekteve, botën e përjetojnë si një varg objektësh, të cilat gjithashtu mund të përbëhen nga objekte ose mund të komunikojnë me të tjerët. Një shembull shumë i thjeshtë: kur e shikoni një person e shqyrtoni si një objekt. Personi ka vetitë, siç janë ngjyra e syve, sjellja dhe mënyra e ecjes. OO, siç tregon edhe vetë emri, është i orientuar në drejtim të objekteve, prandaj për fillim është më së miri të ceket se objekti është njëri prej njësive mësimore themelore të programimit OO. Në qoftë se programin e konsideroni si organizëm, do të ishte i përbërë nga modulet me funksione të ndryshme, siç janë organet. Këto module funksionale përbëhen nga pjesët e veta, që janë në të vërtetë, objektet që komunikojnë ndërmjet vete.

Objekti përkufizohet si njësi e posaçme që përmban të dhënat dhe sjelljen. Të dhënat tregojnë gjendjen e objektit.

Marrim si shembull objektin e një shkolle. Që të jetë objekt, duhet ta paraqesim me një gjendje dhe sjellje të caktuar. Fillimisht të përgjigjemi në pyetje se çfarë është gjendja e një shkolle? Në pyetje janë karakteristikat e saj siç janë emri, viti i themelimit, adresa, numri i nxënësve etj. Në figurën 4.1. mund të vini re se janë paraqitur nxënësit që hyjnë në shkollë. Edhe ata janë objekte, por të tipit të ndryshëm në krahasim me shkollën dhe kanë gjendjen dhe sjelljen.

SHKOLLA
Emri: "Sami Frashëri"
Viti i themelimit: 2010
Numri i nxënësve: 1154
Vendi : Kumanovë
Gjuha mësimore: Shqip

NXËNËSI
Emri: "Liridon Fetai"
Data e lindjes: 05.06.1999
viti: III
Paralelja: 7
Drejtimi: Matematiko Natyror A
Numri i mungesave: 3

Në figurë janë paraqitur karakteristikat e këtyre dy objekteve. Vëmë re se çdo karakteristikë ka një vlerë të veten. Këto janë vlerat që një objekt e dallojnë nga objekti tjetër (p.sh. nuk do të ketë çdo nxënës emrin e njëjtë, mbiemrin apo numrin e mungesave). Të gjitha këto vlera kanë tip të caktuar. Mund të vëreni se emri i shkollës është varg karakteresh, numri i nxënësve është numër i plotë. Në program, këto të dhëna do të jenë të shënuara nëpërmjet tipave përkatës të gjuhës programuese C++.

Tani duhet të përgjigjemi në pyetjet, çfarë paraqet sjellja e një shkolle? Edhe pse mendja e shëndoshë na tregon se një shkollë nuk ka "sjellje", përveç nëse ka ndihmesën e një teknologjie moderne (p.sh. godinat inteligjente), ne si programues mund të shqyrtojmë këtë pyetje nga një kënd tjetër. Sjellja nuk nënkupton doemos një aktivitet që shkolla e ndërmerr, por edhe aktivitetet që bëhen në shkollë i përkasin kategorisë së sjelljes. Kështu, për shembull sjellje shkolle mund të jenë: ndryshimi i emrit të shkollës, ndërtimi i pjesës së re të ndërtesës me çfarë zmadhohet sipërfaqja e përgjithshme e shkollës ose ndërtimi i paraleles së re për një numër më të madh të nxënësve të klasës së tetë, etj. Çka mund të themi për sjelljen e nxënësit. Mund të përfshihen aktivitet si: Merr pjesë në orë të mësimi, merr pjesë në aktivitet të grupit letrar, etj.

3.1.1. Klasat dhe objektet

Në natyrë mund të takohet një numër i madh i entiteteve të ngjashme. Disa janë identikë disa kanë shumë pak dallime, kurse disa të tjera janë plotësisht të ndryshëm. Kështu, për shembull, është themeluar klasifikimi I qenieve të gjalla në gjitarë, zvarranik, shpezë, peshq, etj. Pra, klasa i përkufizon disa veti të përbashkëta dhe sjellje të entiteteve që numërohen nën klasifikimin e saj, por që dallohen në detajet e lidhura me vetitë e caktuara.

Në programim, *klasa* mund të imagjinohet si një shabllon nga i cili formohen objektet. Klasa përkufizon se cilat *karakteristika* do t'i ketë secili objekt i asaj klase, si dhe *sjelljet* e saj. Pra çdo objekt është një instancë e klasës së vet. Shembull tabela për klasën nxënësi

Në fjalorin e programimit të orientuar në objekte, *instanca* është thjesht shprehja për objektin, në dallim nga *klasa*, që paraqet objektin në mënyrë abstrakte. Prandaj termat "*objekti nxënës*", "*instanca nxënës*" i referohet të njëjtës gjë: objektit që është formuar nga klasa *Nxenesi*.

Në mënyrë të njëjtë si në botën reale, në të cilën çdo objekt ka karakteristikat dhe sjelljen që e dallojnë nga objektet e tjera, ashtu edhe në gjuhën programuese Java, çdo klasë përmban karakteristikat (atributet) dhe posedon sjelljen e përkufizuar, me të cilën dallohet nga klasat e tjera.

Karakteristika (attribute) konsiderohen të gjitha vetitë që klasën dhe objektin e dallojnë nga të tjerët dhe e përkufizojnë identitetin e vet, p.sh. forma, ngjyra, madhësia etj.

Të gjitha **funksionet** që ndikojnë në karakteristikat e objektit konsiderohen si sjellje të tij.

P.sh. funksioni i ndryshimit të paraleles, ose drejtimit, ka një ndikim konkret të përkufizuar në paralelen gjegjësisht në drejtimin e objektit. Të gjitha funksionet me të cilat ndikojnë në gjendjen e një objekti (ose nëpërmjet të cilave objektet ndikojnë në mënyrë individuale në gjendjen e vet), në POO quhen **metoda**. Në lëmin e programimit, që të përcaktohet se si sillet

një objekt, përdoren metodat që kryejnë detyra të caktuara. Ato, punën e vetë e kryejnë mbi instancat e asaj klase, mirëpo ndikimi i tyre nuk është i kufizuar ekskluzivisht në gjendjen e një objekti të vetëm.

Metodat, të cilave nuk mund t'u qasen objektet e tjera nga jashtë quhen *sjellje e brendshme*. Në anën tjetër, metodat mund të "ftojnë" edhe metodat në objektet e tjera dhe në atë mënyrë të "lutet" objekti tjetër që të ndryshojë gjendjen e tij. Një mënyra e tillë e komunikimit shpeshherë quhet *dërgimi i mesazheve*. Programimi i orientuar në objekte është i ngritur në tre parime bazë: enkapsulimi, të trashëguarit dhe poliformizmi. Enkapsulimi do të thotë se të gjitha informacionet për objektin (vetitë) dhe proceset (metodat) mbi të përmbahen në definimin e objektit. Për shembull, të vëzhgojmë sistemin kompjuterik si objekt. Atë e përshkruajmë me ndihmën e vetive të tij: lloji i procesorit, madhësia e diskut, memories, shpejtësia e modemit, madhësia e monitorit.... Secila nga karakteristikat e cekura paraqet veti të sistemit kompjuterik. Metodat paraqesin mënyrat në të cilat do të sillet sistemi kompjuterik në situata të ndryshme, për shembull, aktivizimi i ndonjë programi. Në atë moment sistemi operativ ekzekuton një varg operacionesh, kurse shfrytëzuesi në ndërkohë

3.1.2. Trashëgimi

Një ndër vetitë më të fuqishme të programit OO është përdorimi i shumëfishtë i kodit programues. Edhe në gjuhët procedurale, në një masë të caktuar është e mundshme të bëhet përdorimi i shumëfishtë i pjesëve të kodit – duke përkufizuar funksione/ metoda që më vonë mund të thirren (aktivohen) një numër të çfarëdoshëm herësh. Mirëpo programimi OO shkon një hap më tutje dhe mundëson përkufizimin e konceptit të trashëgimit:

Trashëgimi mundëson krijimin e klasës që është e ngjashme me atë paraprahe, por që megjithatë ka edhe disa veti të veta të posaçme.

Trashëgimi mundëson krijimin e hierarkisë në atë mënyrë që klasën e përgjithshme mund ta trashëgojnë klasat e tjera. Klasat specifike trashëgojnë attribute dhe sjellje të klasës së përgjithshme dhe/ose shtojnë atë që për të është e vetme. Në atë rast, ekziston raporti prindër – pasardhës.

Në C++ klasa e përgjithshme (d.m.th. klasa që trashëgohet) quhet **mbiklasa** (anglisht *superclass*), kurse klasa që trashëgon quhet **nënklasa** (anglisht *subclass*). Çdo klasë ka mbiklasën e vet, dhe mund të ketë më shumë nënklasa. Nënklasa trashëgon të gjitha ndryshoret dhe metodat që janë përkufizuar në mbiklasën dhe i shton ndryshore ose/dhe metodat e veta të posaçme. Një shembull është paraqitur në figurën 1., kurse mbi raportin e klasave A, B, C dhe D në hierarki mund të thuhet:

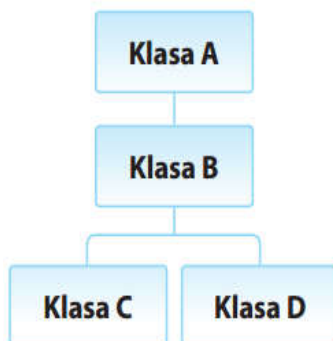


Fig 1.Trashëgimia

- Klasa A është *mbiklasë* e klasës B, kurse klasa B është *nënklasë* e klasës A (shpeshherë thuhet se klasa B e trashëgon klasën A).
- Klasa B është *mbiklasë* e klasës C dhe D, kurse klasat C dhe D janë *nënklasa* (e trashëgojnë) klasën B.

Një nga problemet kryesore që paraqitet në një organizatë hierarkike të tillë është klasifikimi i karakteristikave dhe sjelljeve të përbashkëta të klasave të ndryshme. P.sh. shqyrtojmë klasat *NxenesiShkollaProfesionale* dhe *NxenesiIT* që paraqesin nxënësit e shkollave të mesme: klasa e parë përfshin vetëm nxënësit e një shkolle profesionale, kurse klasa e dytë gjimnazistët që për lëndë zgjedhëse kanë marrë grupin e lëndëve nga informatika. Të dy klasat i karakterizojnë: emri, mbiemri, numri evidentues, emri i shkollës, paralelja, nota nga matematika (si edhe për objektet e klasës *Nxenesi*). Mirëpo, për klasën *NxenesiShkollaProfesionale* është e njohur edhe nota nga praktika profesionale të cilën, nxënësit, janë të detyruar ta bëjnë, kurse për *NxenesiIT* nota nga programimi. Domethënë, të dy klasat e trashëgojnë klasën *Nxenesi*, siç është paraqitur në figurën.

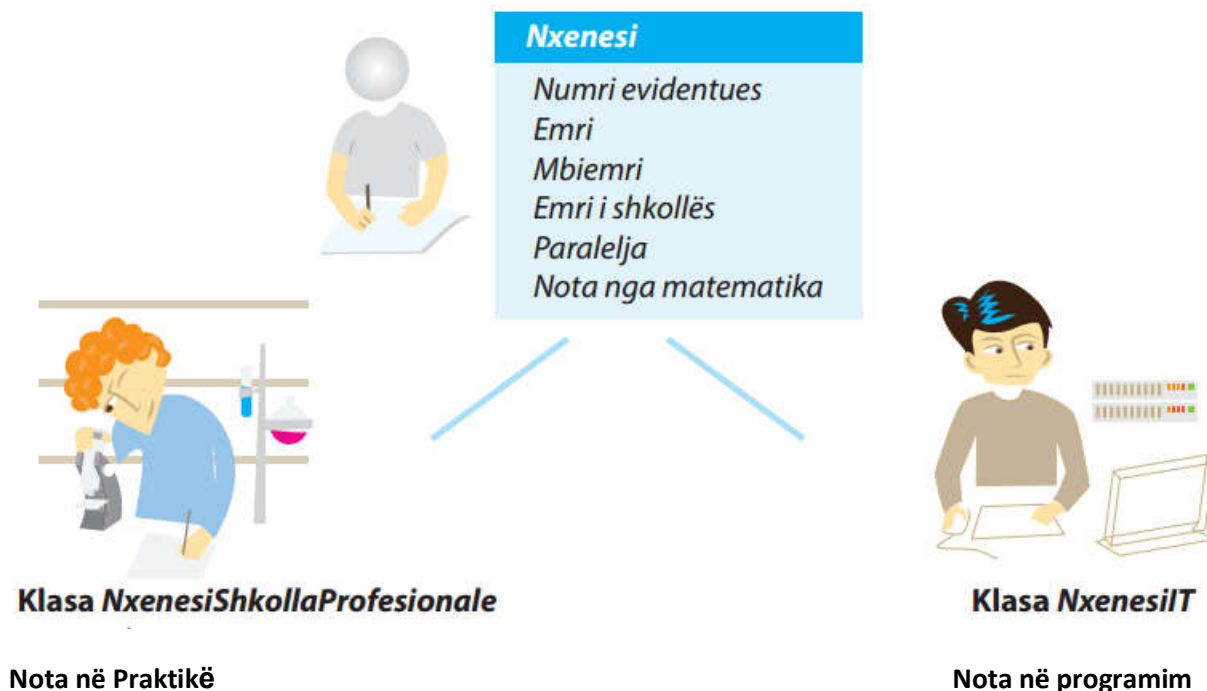


Fig 2. Trashëgimi

Poliformizmi ka kuptimin që shumë objekte mund të kenë të njëjtën metodë, kurse veprimi adekuat ndërmerret nga objekti që e thirr atë metodë. Shembull, programet për përpunimin e teksteve tekstin mund ta bartin në ekran ose shtypës. Secili nga këta dy objektet, më ndihmën e metodës së tregimit ose shtypjes, e lajmërojnë objektin që të shfaq tekstin në vendin e caktuar. Metoda tregon se çfarë duhet të bëjmë në vartësi nga objekti që e thirr metodën.

3.2. Forma e përgjithshme e klasës

Klasa përkufizohet shumë thjesht – duke përkufizuar të gjitha elementet e saj që e përbëjnë: të dhënat, gjegjësisht ndryshoret (*atributet*) dhe funksionet që me ato të dhëna diçka veprojnë (*metodat*). Edhe pse janë të mundshme klasat që përmbajnë vetëm ndryshoret ose vetëm metodat, shumica e klasave përmbajnë zakonisht ndryshoret dhe metodat. Forma e përgjithshme e përkufizimit të klasave bëhet nëpërmjet të fjalës së rezervuar *class* në mënyrën e mëposhtme:

```
class emriKlases {
tipi_atributit emri_atributit1;
tipi_atributit emri_atributit2;
// ...
tipi_atributit emri_atributitN;
```

```

tipi_vleres_kthyeses_metodes emri_metodes1(lista_parametrave) {
// trupi i metodës
}
// ...
tipi_vleres_kthyeses_metodes mri_metodesM(lista_parametrave) {
// trupi i metodës
}
};

```

Është me rëndësi të përmendet se në C++ përdoret rregulla e pashkruar që emrat e klasave të fillojnë me shkronja të mëdha. Kështu klasa dallohet lehtë nga ndryshoret dhe metodat, emrat e të cilave zakonisht shënohen me shkronja të vogla.

Të shkruajmë tani pjesën e kodit për krijimin e klasës Nxenesi: numriEvidences, emri, mbiemri, paralelja, emriShkolles dhe notaNgaMatematika.

```

class Nxenesi {
int numriEvidences;
string emri;
string mbiemri;
int klasa;
string emriShkolles;
int notaNgaMatematika;
};

```

3.2.1. Krijimi i metodave

Metodat përcaktojnë se si disa objekte të caktuara sillen, çfarë ndodh me to gjatë krijimit dhe cilat veprime kryejnë. Metodat përkufizohen në kuadër të trupit të klasës në bazë të sintaksës së mëposhtme:

```

tipi_vleres_kthyeses_metodes emri_metodes(lista_parametrave) {
// trupi i metodës
}

```

Këtu *tipi_vleres_kthyeses_metodes* përcakton tipin e të dhënave që i kthen metoda. Mund të jetë secili tip i të dhënave (bool, char, int, float), duke kështu edhe tipat e klasave që i krijon vetë programuesi. Në qoftë se metoda nuk kthen një vlerë të caktuar, tipi i saj i të dhënave kthyesë duhet të shënohet me *void*. Emri i metodës është përcaktuar me identifikuesin *emri_metodes*. Mund të jetë cilido identifikues i përkufizuar në pajtim me rregullat e përmendura.

Lista e parametrave përmban vargun e çifteve *tipi_parametrit emri_parametrit* të ndarë me presje. Parametrat, janë në esencë, ndryshore që marrin vlerën e argumenteve të dërguar

metodës në momentin e thirrjes së saj. Në qoftë se metoda nuk ka parametra, lista e parametrave do të jetë e zbrazët, prandaj ceken vetëm kllapat e zbrazëta. Kur tipi i të dhënave kthyesë nuk është void, metoda kthen vlerën e cilanë një moment të caktuar, zakonisht në fund të ekzekutimit të metodës, jepet me ndihmën e komandës *return* në formën:

return vlera;

Në shembullin e klasës *Nxenesi* do të njoftohemi me mundësitë e përmendura gjatë përkufizimit të metodave të ndryshme. Klasës *Nxenesi* i shtojmë:

- metodën *peseMatematika*, që nxënësit i evidenton notën 5 nga matematika;
- metodën *shenoNotenMatematika*, që nxënësit i evidenton notën e re nga matematika, që dorëzohet si argument hyrës i metodës;
- metodën *paraqitNotenMatematika*, e cila me një mesazh përkatës paraqet notën nga matematika;
- metodën *ktheNotenMatematika*, që kthen notën aktuale nga matematika.

3.2.2. Krijimi i objekteve dhe puna me objektet

Më parë kemi sqaruar se klasat paraqesin vetëm shabllone për krijimin e objekteve, d.m.th. objekti paraqet shembull (instancë) të një klase përkatëse. Në C++ objektet deklarohen në mënyrë të ngjashme si dhe ndryshoret. Në fillim jepet emri i klasës, pastaj emri i objektit konkret:

emriKlases emriObjektit;

Në shembullin tonë të klasës *Nxenesi*:

```
Nxenesi nxenesi;
// formohet objekti i tipit Nxenes me emrin nxenesi.
```

Mbasi që komanda paraprake të ekzekutohet, objekti *nxenesi* do të jetë një mostër e klasës *Nxenesi*. Fjala është për objektin që do të ekzistojë "fizikisht". Në këtë mënyrë, prej një klase mund të formohet numri i dëshiruar i objekteve, si në shembullin e mëposhtëm.

```
Nxenesi nxenesi1;
Nxenesi nxenesi2;
```

Gjatë secilit krijim të objektit të ri ai i përfton kopjet e veta të attributeve të përkufizuar me klasën. Prandaj, çdo objekt i tipit *Nxenesi*, do të përmbajë kopjet e veta të attributeve *emrin*,

mbiemrin, numriEvidences, emriShkolles, paralelja, notaNgaMatematika. Që t'i qasemi këtyre attributeve përdoret operatori pikë (.), në mënyrën e mëposhtme:

emriObjektit.emriAtributit;

Operatori pika e lidhë emrin e objektit me emrin e atributit të instancës (mostrës). Kështu mund të ndryshoni vlerat e notave nga matematika për të dy nxënësit në mënyrën e mëposhtme:

```
nxenesi1.notaNgaMatematika = 5;
nxenesi2.notaNgaMatematika = 3;
```

Kjo komandë i kumton përkthyesit që kopjes së atributit *notaNgaMatematika*, që ndodhet brenda objektit *nxenesi1*, t'i shoqërohet vlera 5, kurse kopjes së atributit *notaNgaMatematika*, që ndodhet brenda objektit *nxenesi2*, t'i shoqërohet vlera 3. Është me rëndësi të theksohet se ndryshimi i atributit të një objekti nuk ndikon në asnjë mënyrë në atributin me emër të njëjtë të objektit tjetër.

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

class Nxenesi {
public:
    int numriEvidences = 1;
    string emri;
    string mbiemri;
    int klasa = 1;
    string emriShkolles;
    int notaNgaMatematika;

    void peseMatematika () { // metoda për shënimin e notës 5 nga matematika
        notaNgaMatematika = 5;
    }
    void shenimiNotaMatematika (int nota) { // metoda për shënimin e notës nga matematika
        notaNgaMatematika = nota;
    }
    void paraqitNotenMatamatika () {
        cout<<"Nota nga matematika " + notaNgaMatematika<<endl;
    }
    int ktheNotenMatematika() { // metoda për kthimin e notës nga matematika
        return notaNgaMatematika;
    }
};

int main()
{
    Nxenesi nxenesi1 ; // krijimi i instances se objektit
    Nxenesi nxenesi2;
    nxenesi1.emri = "Agron"; //inicializimi i attributeve te nxënësit
    nxenesi1.mbiemri = "Islami";
    nxenesi1.notaNgaMatematika = 5;
```

```

nxenesi2.emri = "Besarta";
nxenesi2.mbiemri = "Kastrati";
nxenesi2.notaNgaMatematika =4;
cout<<"Nxenesi: " + nxenesi1.emri << " " << nxenesi1.mbiemri<<" ka noten nga
matematika: "<<nxenesi1.notaNgaMatematika<<endl;
// printimi i të dhënave për nxënësin
cout<<"Nxenesi: " <<nxenesi2.emri << " " << nxenesi2.mbiemri<<" ka noten nga
matematika: " <<nxenesi2.notaNgaMatematika<<endl;
nxenesi1.shenimiNotaMatematika(4); // thirrja e metodës
nxenesi2.shenimiNotaMatematika(5);
int shuma;
shuma = nxenesi1.ktheNotenMatematika() +nxenesi2.ktheNotenMatematika();
// shuma e notave nga matematika
double mesatarja;
mesatarja = shuma/(2*1.0); // njehsimin e mesatares, printimi
cout<<"Nota mesatare nga matematika eshte "<<mesatarja<<endl;
/* Nota e Besartes (nxenesi2) dhe Agronit (nxenesi1) do t'i shkëmbejne
vlerat si:
1. Ndryshores notaA i shoqërojmë vlerën e notës së Besartes
2. Nota e re të Besartes e merr vlerën e notës së Agronit
3. Nota e re e Agronit e merr vlerën e notës paraprake të Besartes
(të cilën e kemi mbajtur në mend me emrin notaA) */
int notaA;
notaA = nxenesi2.ktheNotenMatematika();
nxenesi2.shenimiNotaMatematika(nxenesi1.ktheNotenMatematika());
nxenesi1.shenimiNotaMatematika(notaA);
cout<<"Nota e re e Besartes eshte: "<<nxenesi2.ktheNotenMatematika();
// printimi i notës
}

```

3.3. Kontrollët i qasjes për atributet dhe metodat

Një prej përparësive themelore të përdorimit të objekteve manifestohet në faktin se objekti nuk ka detyrim të zbulojë të gjitha atributet dhe sjelljet (metodat). Në softuerin e projektuar mirë të OO, objekti duhet të zbulojë vetëm detajet e domosdoshme për komunikim, kurse ato që nuk janë të rëndësishme për përdorim duhet të fshehen nga objektet e tjera. Kjo quhet *enkapsulacion*. Për shembull, objekti që njehson katrorin e një numri duhet të sigurojë metodat për përfitim të rezultatit. Mirëpo, atributet e brendshme dhe algoritmet, që përdoren për njehsimin e katrorit, nuk duhet të jenë në dispozicion të objektit që bën thirrjen. Kjo sigurohet me kontrollin e qasjes së elementeve të klasës që jepen gjatë deklarimit. Specifikuesit e qasjes në C++ janë **public** (publik), **private** (privat) dhe **protected** (i mbrojtur). Specifikimi i **protected** ka kuptim vetëm gjatë trashëgimit.

Niveli i qasjes	Modifikuesi i qasjes	Përshkrimi
Publik	public	Klasa, atributi ose metoda që janë të shënuara në këtë mënyrë janë të dukshme kudo dhe kanë qasje të qartë.
Privat	private	Atributi ose metoda që janë shënuar në këtë mënyrë janë të dukshëm në kuadër të klasës në të cilën janë shënuar.
I nënkuptuar	Nuk shkruhet asgjë	Klasa, atributi ose metoda që janë të shënuar në këtë mënyrë janë të dukshëm vetëm në kuadër të paketës në të cilën janë shënuar.
I mbrojtur	protected	Është i njëjtë si niveli i paketës i qasjes, vetëm që dukshmëria zgjerohet në të gjitha klasat nga paketat e tjera që trashëgojnë klasën e cila është e mbrojtur ose përmban elementin mbrojtës.

Specifikuesi i qasjes i paraprin specifikimit të tipit të anëtarit. Për shembull:

```
private: int i;
protected int j
public: double k;
int metodaIme(int a, char b) { //...
```

3.4. Konstruktorët

Gjatë inicializimit të objektit të ri rezervohet hapësirë në memorie në të cilën do të vendoset objekti i klasës së dhënë dhe vlerat e inicializuara të attributeve të tyre. Në rastin e përgjithshëm, mund të jetë shumë e lodhshme të inicializohen vlerat e të gjitha attributeve në vlerat e dëshiruara, prandaj në C++ lejohet përdorimi i **konstruktorëve**, detyra e të cilëve është pikërisht inicializimi i attributeve të instancës.

Konstruktori është sipas mënyrës së shënimit i ngjashëm me metodën, sepse mund të përmbajë komanda të çfarëdoshme, mirëpo, për dallim nga metoda, nuk mund të thirret drejtpërdrejt, por këtë e bën Java në mënyrë automatike gjatë krijimit të objektit të ri me komandën new.

Konstruktorët në pamje të parë duken shumë të çuditshëm, sepse krahas tyre nuk jepet tipi privat i të dhënave, madje as tipi void. Ai tip i të dhënave nuk është i nevojshëm, sepse konstruktorët në të vërtetë kthejnë tipin e të dhënave që i përgjigjet vetë klasës.

Me fjalë të tjera, *detyra e konstruktorit* është që të inicializojë objektin në procesin e krijimit të tij, menjëherë ekziston objekti i gatshëm për përdorim.

Gjithashtu që C++ të njohë konstruktorin, ai duhet të ketë emrin e njëjtë si edhe vetë klasa. Klasa mund të ketë më shumë konstruktorë dhe ata dallohen vetëm sipas parametrave hyrës (numrit dhe/ose tipit).

Sintaksa e krijimit të konstruktorit është:

```
public Emri_i_klasës (lista_parametrave) {  
    // trupi i konstruktorit  
}
```

Thirrja e konstruktorit brenda funksionit main bëhet:

```
Emri_i_klases instanca(lista me parametra);
```

Thirra e konstruktorit pa parametra:

```
Emri_i_klases instanca;
```

Shembull:

```
#include<iostream>  
using namespace std;  
class Thyesa {  
    int numeruesi,emeruesi;  
public:  
    //konstruktori  
    Thyesa (int n, int e){  
        emeruesi=e;  
        numeruesi=n;  
    }  
    Thyesa ()  
    {  
        numeruesi=1;  
        emeruesi=1;  
    }  
  
    void paraqit()  
    {  
        cout<<numeruesi<<" / "<<emeruesi<<endl;  
    }  
};  
int main(){  
    int e,n;
```



```

//Thirrja e konstruktorit pa parametra
Thyesa x1;
cout<<"Thyesa 1: \n";
x1.paraqit();
cout<<"Thyesa 2: \n";
cout<<"Shkruaj numeruesin: ";
cin>>n;
cout<<"Shkruaj emeruesin: ";
cin>>e;
// Thirrja e konstruktorit me parametra
Thyesa x2(n,e);
x2.paraqit();
return 0;
}

```

3.5. Metodat Get dhe Set

Siç është sqaruar paraprakisht, disa attributeve të klasës u shoqërohet e drejta e qasjes private me qëllim që të "fshehen" dhe të mbrohen nga qasja prej klasave të tjera. Për ato attribute të klasës, të cilave megjithatë dëshirojmë t'u mundësojmë leximin ose ndryshimin e vlerave jashtë klasës konkrete, përdoren metodat *get* dhe *set* ose, më popullore *geterët* dhe *seterët*. **Metoda *get*** përdoret për leximin e vlerave të atributit të dhënë (kthen vlerën e atributit), kurse **metoda *set*** ka për parametër vlerën që e vendos si vlerë të re të atributit. Gjatë krijimit të metodës, e cila do të kthejë vlerën e atributit të dhënë, shumë programues do të përdornin emërtime të ndryshme, të tipit "*merre...*", "*ktheje...*". Mirëpo që të zgjidhet ky problem dhe të realizohet uniteti në emërtimin e metodave të cilat kthejnë vlerën e një atributi. Përfundimi i vetëm janë *get* metodat për atributet *bool*, emërtimi i të cilave është ***isEmriAtributit***. Në mënyrë të ngjashme, *set* metodat kanë emërtimin ***setEmriAtributit***, që vlen për atributet e çdo tipi.

Shembull:

```

#include<iostream>
using namespace std;
class Thyesa {
int numeruesi,emeruesi;
public:
//konstruktori pa parametra
Thyesa ()
{
    numeruesi=1;
    emeruesi=1;
}

//konstruktori me parametra
Thyesa (int n, int e){
    emeruesi=e;
    numeruesi=n;
}

```

```

    }

    int getEmeruesi()
    {
        return emeruesi;
    }
    void setEmeruesi(int e)
    {
        emeruesi=e;
    }
    int getNumeruesi()
    {
        return numeruesi;
    }
    int setNumeruesi(int n)
    {
        numeruesi=n;
    }

    void paraqit()
    {
        cout<<numeruesi<<" / "<<emeruesi<<endl;
    }
    void thjeshto()
    {
        int i=2;
        while(i<=emeruesi && i<=numeruesi)
        {
            if(numeruesi%i==0 && emeruesi % i==0)
            {
                numeruesi=numeruesi/i;
                emeruesi=emeruesi/i;
            }
            else
                i++;
        }
    }

};

int main() {
    int e,n;

    Thyesa x1;
    //Thyesa me konstruktor pa parametra
    cout<<"Thyesa 1: \n";
    x1.paraqit();
    cout<<"Thyesa 2: \n";
    cout<<"Shkruaj numeruesin: ";
    cin>>n;
    cout<<"Shkruaj emeruesin: ";
    cin>>e;
    Thyesa x2(n,e);
    x2.thjeshto();
    // THyesa me konstruktor me parametra
    x2.paraqit();
}

```

```

    return 0;
}

```

3.6. Trashëgimi

Sintaksa themelore për deklarimin e nënklasës që trashëgon mbiklasën është:

```

class A: tipi i trashëgimit B{
//Trupi I klases
}

```

Tipi i trashëgimit mund të jetë: public, protected ose private:

Trashëgimi public: Gjatë trashëgimit nga klasa bazë me public anëtarët public bëhen anëtarë public në klasën e derivuar, dhe anëtarët protected të klasës bazë bëhen protected për klasën e derivuar. Anëtarët private të klasës bazë nuk mund të qasen direct nga klasa e derivuar, por mund të kenë qasje përmes anëtarëve public dhe protected të klasës bazë.

Trashëgimi protected: Gjatë trashëgimit nga klasa bazë me protected anëtarët public dhe protected bëhen anëtarë protected në klasën e derivuar,

Trashëgimi private: Gjatë trashëgimit nga klasa bazë me private anëtarët public dhe protected bëhen anëtarë private në klasën e derivuar.

Ne shembullin në vijim implementohet trashëgimi i klasave Drejtekendeshi dhe Romboidi nga klasa Paralelogrami.

```

#include<iostream>
using namespace std;
class Paralelogrami
{
    protected:
    int a;
    int b;
    public:
    Paralelogrami ()
    {
        a=0;
        b=0;
    }
    Paralelogrami (int _a,int _b)
    {
        a=_a;
        b=_b;
    }
    void setA(int _a) {
        a=_a;
    }
    int getA()
    {

```

```

        return a;
    }
    void setB(int _b) {
        b=_b;
    }
    int getB()
    {
        return b;
    }
    int njehsoPerimetrin()
    {
        return 2*(a+b);
    }
};

class Drejtekesdeshi : public Paralelogrami
{
    public :
        Drejtekesdeshi(int _a,int _b)
        {
            a=_a;
            b=_b;
        }
        int njehsoSyprinen()
        {
            return a*b;
        }
};

class Romboidi: public Paralelogrami{
    private :
        float lartesia_a;
        float lartesia_b;

    public:
        Romboidi (int _a,int _b) {
            a=_a;
            b=_b;
        }
        Romboidi (int _a,int _b,float l_a,float l_b) {
            a=_a;
            b=_b;
            lartesia_a=l_a;
            lartesia_b=l_b;
        }
        void setLartesia_a(int _l)
        {
            lartesia_a=_l;
        }
        float getLartesia_a()
        {
            return lartesia_a;
        }

        void setLartesia_b(int _l)
        {

```

```

        lartesia_b=_l;
    }
    float getLartesia_b()
    {
        return lartesia_b;
    }

    float njehsoSyprinen()
    {
        return a*lartesia_a;
    }
};

int main()
{
    int P;
    float S;
    //krijohet objekti drejtekesdesh me dimensionet a=6 dhe b=4
    Drejtekesdeshi d(6,4);
    cout<<"Drejtekesdeshi i krijuarme dimensionet a="<<d.getA()<<" b="
"<<d.getB()<<endl;
    cout<<"Perimetri i drejtekesdeshit: "<<d.njehsoPerimetrit()<<endl;
    cout<<"Syprina e drejtekesdeshit: " <<d.njehsoSyprinen()<<endl;
    cout<<"-----\n";
    cout<<"-----\n";
    // krijohet objekti Romboid me dimensione a=4 b=3,
    Romboidi r(4,2);
    //vendoset vlere 5 per lartesine
    r.setLartesia_a(5);
    cout<<"Romboidi i krijuarme dimensionet a="<<r.getA()<<" b="
"<<r.getB()<<" lartesia h="<<r.getLartesia_a()<<endl;
    cout<<"Perimetri i romboidit: "<<r.njehsoPerimetrit()<<endl;
    cout<<"Syprina e romboidit: " <<r.njehsoSyprinen()<<endl;
    return 0;
}

```

Shembull: Të shkruhet program në OOP i cili modelon thyesen, mundëson që thyesës ti shtohet, zbritet, të shumezohet ose pjestohet me një thyesë të dhënë.

```

#include<iostream>
using namespace std;
class Thyesa {
int numeruesi,emeruesi;
public:
    //konstruktori pa parametra
    Thyesa ()
    {
        numeruesi=1;
        emeruesi=1;
    }
    //konstruktori me parametra
    Thyesa (int n, int e){
        emeruesi=e;
    }
}

```

```

    numeruesi=n;
    thjeshto();
}

    int getEmeruesi()
    {
        return emeruesi;
    }
    void setEmeruesi(int e)
    {
        emeruesi=e;
    }
    int getNumeruesi()
    {
        return numeruesi;
    }
    int setNumeruesi(int n)
    {
        numeruesi=n;
    }

    void paraqit()
    {
        cout<<numeruesi<<" / "<<emeruesi<<endl;
    }
    void thjeshto()
    {
        int p;
        if(numeruesi<emeruesi)
            p=numeruesi;
        else
            p=emeruesi;
        int i=2;
        while(i<=emeruesi && i<=numeruesi)
        {
            if(numeruesi%i==0 && emeruesi % i==0)
            {
                numeruesi=numeruesi/i;
                emeruesi=emeruesi/i;
            }
            else
                i++;
        }
    }

}

/* algoritmi i cili do te caktoje shumen e thyesave:
a/b +c/d= (a*d+c*b)/b*d */
void shto(Thyesa a)
{
    int n= numeruesi*a.getEmeruesi()+a.getNumeruesi()*emeruesi;
    int e=emeruesi*a.getEmeruesi();
    numeruesi=n;
    emeruesi=e;
    thjeshto();
}

/* algoritmi i cili do te caktoje ndryshimin e thyesave:
a/b +c/d= (a*d-c*b)/b*d */

```

```

void zbrit(Thyesa a)
{
    int n= numeruesi*a.getEmeruesi()-a.getNumeruesi()*emeruesi;
    int e=emeruesi*a.getEmeruesi();
    numeruesi=n;
    emeruesi=e;
    thjeshto();
}
/* algoritmi i cili do te caktojë prodhimin e thyesave:
(a/b) * (c/d) = (a*c) / (b*d) */
void shumezo(Thyesa a)
{
    int n= numeruesi*a.getNumeruesi();
    int e=emeruesi*a.getEmeruesi();
    numeruesi=n;
    emeruesi=e;
    thjeshto();
}
/* algoritmi i cili do te caktojë shumën e herës:
(a/b) / (c/d) = (a*d) / (c*b) */
void pjesto(Thyesa a)
{
    int n= numeruesi*a.getEmeruesi();
    int e=emeruesi*a.getNumeruesi();
    numeruesi=n;
    emeruesi=e;
    thjeshto();
}
};
int main() {
    int e,n;
    Thyesa x1(2,3);
    //Thyesa me konstruktor pa parametra
    cout<<"Thyesa 1: \n";
    x1.paraqit();
    cout<<"Thyesa 2: \n";
    cout<<"Shkruaj numeruesin: ";
    cin>>n;
    cout<<"Shkruaj emeruesin: ";
    cin>>e;
    cout<<"Thyesa e dhënë pas thjeshtimit është: ";
    Thyesa x2(n,e);
    x2.thjeshto();
    x2.paraqit();
    cout<<"Nëse thyesën 2 i shtojmë theysën 1"<<endl;
    x2.shto(x1);
    x2.paraqit();

    cout<<"Nëse nga thyesa 2 e zbresim thyesen 1"<<endl;
    x2.zbrit(x1);
    x2.paraqit();

    cout<<"Nëse thyesen 2 e shumezojmë me theysën 1"<<endl;
    x2.shumezo(x1);
    x2.paraqit();

    cout<<"Nëse thyesen 2 e pjestojmë me theysën 1"<<endl;

```

```
x2.pjesto(x1);  
x2.paraqit();  
return 0;  
}
```


4. PJESËT STRUKTURE DHE KODI NË POO

4.1. Strukturat për degëzim

Gjatë procesit të programimit vijmë në situata në të cilat duhet që programit të varësisht nga kushti i dhënë të zgjedhë nga dy ose më tepër opsione që do ti egzekutojë ashtuqë një apo më tepër urdhëra të egzekutojë apo mos të egzekutojë. Varësisht nga kushti i dhënë të zgjedhë sekuencë të urdhërave ose të mos egzekutojë urdhër ose bashkësi urdhërash të dhënë.

4.1.1. Struktura për degëzim if-else

Shembulli në vijim sqaron qëllimin e urdhërave për kushtëzim:

Nëse fjalëkalimi është i saktë atëherë mundëso qasje në sistem

Fjalinë do ta ndajmë në dy pjesë:

Nëse

FJALËKALIMI ËSHTË I SAKTË

Atëherë

MUNDËSO QASJE NË SISTEM

Kushti **FJALËKALIMI ËSHTË I SAKTË** mund të ketë vetëm njërën nga vlerat ose **i vërtete** ose **i pavërtete** e cila vërtetohet nga sistemi dhe është tregues se a do ta kemi qasje në të ose jo. Mundëso qasje në sistem është urdhër ose bashkësi urdhërash të cilat do ti egzekutohen me qëllim që të mundësojë shfrytëzuesit ti qaset sistemit, nëse fjalëkalimi i dhënë është i saktë.

Në rastin e përgjithshëm kemi kushtin dhe bashkësinë e urdhërave.

Urdhëri për kushtëzim mund të paraqitet edhe grafikisht me bllok diagram.

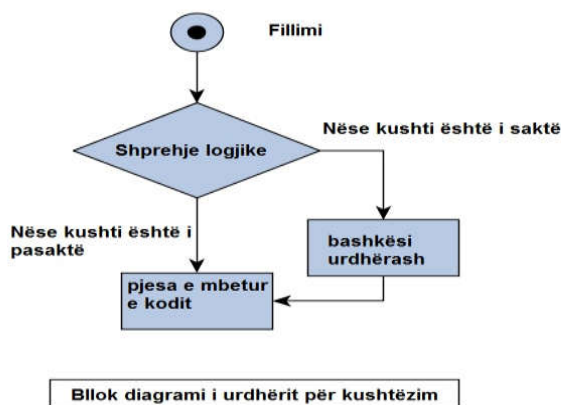


Fig 3. Urdhëri për kushtëzim

Bashkësia e urdhërave e paraqitur në anën e djathtë egzekutohet vetëm nëse kushti është i saktë:

Sintaksa e urdhërit për kushtëzim në C++ është:

```
if (shprehja)
    urdhëri;
```

Nëse shprehja ka vlerë të ndryshueshme nga 0 atëherë egzekutohet urdhëri, nëse ka vlerë 0 atëherë urdhëri nuk egzekutohet.

Nëse kemi bashkësi urdhërash (dy apo më tepër urdhëra) të cilët duhet të egzekutohen atëherë urdhërat duhet të vendosen brenda kllapave.

```
if (shprehja)
{
    Urdhëri1;
    Urdhëri2;
    .....
    UrdhëriN;
}
```

Urdhëri për kushtëzim if mund të shkruhet edhe në formën:

```
if (shprehja)
    urdhëri1;
else
    urdhëri2;
```

Nëse shprehja ka vlerë të ndryshueshme nga 0 atëherë egzekutohet urdhëri1, nëse ka vlerë 0 atëherë egzekutohet urdhëri2.

Nëse kemi bashkësi urdhërash (dy apo më tepër urdhëra) të cilët duhet të egzekutohen atëherë urdhërat duhet të vendosen brenda kllapave.

```
if (shprehja)
{
    Urdhëri_v1;
    Urdhëri_v2;
    .....
    Urdhëri_vM;
}
else
{
    Urdhëri_p1;
```

```
    Urdhëri_p2;  
    .....  
    Urdhëri_pN;  
}
```

Detyra:

- Numri i dhënë x të tregohet a është çift ose tek;
- Numri i dhënë x të tregohet a është pozitiv, negativ ose 0.
- Nëse janë dhënë gjatësitë e segmenteve, trego a mundet të jenë gjatësitë e brinjëve të trekëndëshit.
- Numri katershifror x të gjendet nëse është palindrom (lexohet njësoj si nga ana e djathtë dhe e majtë):

Nëse është dhënë nota e nxënësit të paraqitet përshkrim i saj me fjalë: (5- shkëlqyeshëm, 4-shumë mirë, 3 - mirë, 2- mjaftueshëm, 1- pamjaftueshëm)

4.1.2. Struktura për degëzim switch

if-else struktura shkruhej dhe egzekutohej në rastin me më shumë opsione. Por if-else është jo shumë i përshtatshëm për raste komplekse. C++ mundëson teknikë më të përshtatshme për të zgjidhur raste me më shumë opsione. Struktura switch njehson shprehje dhe duke u bazuar në rezultatin e njehësuar zgjedh urdhërat nga lista të cilat i përgjigjen rezultatit të njehësuar; vlerat e listës quhen raste (ang. cases):

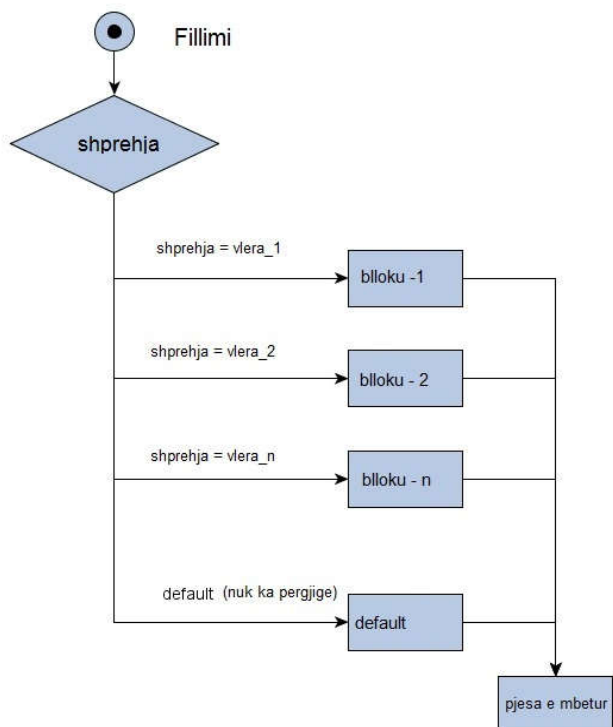


Fig 4. Urdhëri switch

Sintaksa e urdhërit switch në C++ është

```

switch(shprehja){
    case shprehja konstante_1 :
        urdherat(s1);
        break; //opsionale
    case shprehja konstante_2 :
        urdhërat(s2);
        break; //opsionale
    .....
    // mund të keni numër të cfardoshëm të rasteve.
    default : //opsionale
        urdhërat(sN);
}
  
```

Shprehja është e tipit numër i plotë, konstantë ose shenjë.

Vlera e shprehjes krahasohet me konstantat numra të plotë: shprehja `konstante_1`, shprehja `konstante_2`, etj.

Nëse vlera e shprehjes është e barabartë me ndonjërin nga konstantat e dhëna, atëherë do të kryhet blloku i urdhërave që i takon asaj konstante. Pas kryerjes së atij blloku, komanda *break* ndalon ekzekutimin e urdhërit *switch-case* dhe programi vazhdon me komandën e parë pas këtij urdhëri. Nëse nuk kemi komandën *break* atëherë egzekutohet edhe urdhëri i ardhshëm (rasti tjetër). Nëse vlera e shprehjes nuk është e barabartë me asnjërin nga konstantat, atëherë kryhet blloku i urdhërave të *default*.

Detyra:

- Për ndryshoret e dhëna *x* dhe *y* të zgjidhet operacini aritmetik (+, -, *, /, %) i cili do të egzekutohet në ato dy numra.
- Shkruaj program me të cilin përdoruesi do të shtyp numrin rendor të muajit të vitit dhe varësisht nga zgjedhja e përdoruesit, në ekran do të paraqitet cilës stinë të vitit i takon ai muaj! Në rast se kemi të dhënë hyrëse që nuk i përgjigjet muajit të paraqitet mesazh përkatës.
- Të shkruhet program i cili për muajin e dhënë tregon sa ditë ka.

4.2. Strukturat për përsëritje

Shpesh herë paraqitet nevoja që një ose më shumë instruksione të kryhen më shumë here. Për shembull: Derisa nuk e kam kuptuar mësimin lexo, Mbledh çdo mungesë për ta catuar shumën e tyre deri sa të arrish në fund të librit. Për të caktuar vlerën mesatare mbledh të gjithë numrat një nga një. Edhe në programim shumë shpesh paraqitet nevoja për përsëritje të një komande të njëjtë (ose një bllok komandash) më shumë herë. Që të mos përsëritet shkruarja e komandës, në kodin programor përdoret struktura për përsëritje e cila quhet cikël - ciklus (loop). Strukturat për përsëritje mundësojnë që ndonjë bllok i komandave të kryhet disa herë. Gjatë kësaj, numri i përsëritjeve të ciklit është i definuar me numrin e paracaktuar të dhënë natyror ose varet nga ndonjë kusht i cili caktohet kur përsëritja do të mbarojë, dmth. kushti mund të kontrollohet para fillimit të ciklit ose pas mbarimit të tij. Çdo përsëritje e ciklit quhet iteracion.

4.2.1. Struktura për përsëritje *do-while*

Struktura **do-while** mundëson përsëritjen e një apo më tepër urdhërave. Urdhërat brenda urdhërit *do-while* përsëriten derisa shprehja kushti është e saktë. Nëse kushti është i saktë

atëherë urdhërat Brenda bllokut do të përsëriten, nëse jo atëherë vazhdon egzekutimin pjesa tjetër e kodit.

Me bllok diagram urdhëri për përsëritje do- while mund të paraqitet:

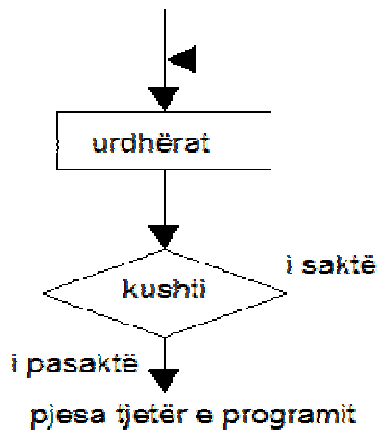


Fig 5. Struktura për përsëritje do-while

Sintaksa e urdhërit për përsëritje në C++ është

```

do
{
    Urdhëri 1;
    Urdhëri 2;
    .....
    Urdhëri N;
}
while (kushti);
  
```

Nëse kushti është i saktë atëherë urdhrat brend bllokut do të përsëriten nëse jo atëherë egzekutimi vazhdon me urdhërat nën bllokun;

Shembull: Të caktohet shuma dhe prodhimi i numrave 1 deri në n;

```

#include<iostream>
using namespace std;
int main()
{
    int n,shuma=0,prodhimi=1,numeruesi=1;
    cout<<"Shkruaj vleren per n ";
    cin>>n;
    do
    {
        suma+=numeruesi;//suma=suma+numeruesi;
        prodhimi*=numeruesi; //prodhimi=prodhimi*numeruesi;
        numeruesi++; //numeruesi=numeruesi+1
    }
    while (numeruesi<=n);
  
```

```

cout<<"Shuma e numrave prej 1 deri ne "<<n<<" eshte "<<shuma<<endl;
cout<<"Prodhiimi i numrave prej 1 deri ne "<<n<<" eshte "<<prodhimi;
return 0;
}

```

4.2.2. Struktura për përsëritje while

Për dallim nga struktura **do-while** tek urdhëri while kushti vendoset në fillim të bllokut, nëse është i saktë atëherë atëherë egzekutohet blloku i urdhërave, nëse jo atëherë program vazhdon egzekutimin pas bllokut. Bllok diagram i urdhërit while do paraqitet ne formën:

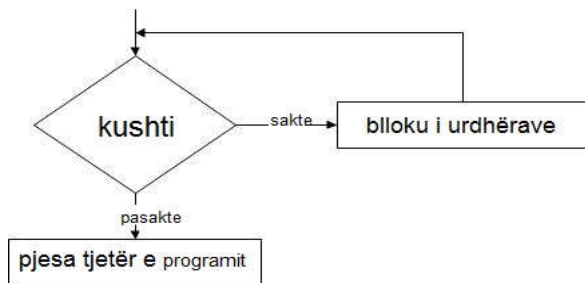


Fig 6. Urdhëri për përsëritje while

Sintaksa në C++ e urdhërit while është:

```

while (kushti)
{
    Urdhëri 1;
    Urdhëri 2;
    .....
    Urdhëri n;
}

```

Shembull: Të caktohet numri dhe shuma e numrave çift në intervalin [m,n].

```

#include<iostream>
using namespace std;
int main()
{
    int n,m,numri,suma,numeruesi;
    cout<<"Shkruaj vleren per fillimin e intervalit m=";
    cin>>m;
    cout<<"Shkruaj vleren per fundin e intervalit n=";
    cin>>n;
    numri=suma=0;
    numeruesi=m;
    while (numeruesi<=n) //kushti qe numeruesi te kete vlere me te vogel se n
    {
        if (numeruesi %2==0) //nese numeruesi ehste numer chift
        {
            numri++;          // numri i numrave rritet per 1

```

```

        shuma+=numeruesi;        // shuma rritet per numeruesin
    }    //fundi i kushit numeruesi%2 ==0
        numeruesi++; //numeruesi rritet per 1
    }//fundi i bllokut while

    cout<<"Ne intervalin ["<<m<<","<<n<<"]"<<" kemi "<<numri<<" numra cift"<<endl;
    cout<<"Shuma e numrave cift ne intervalin ["<<m<<","<<n<<"]"<<" eshte "<<shuma<<endl;

        return 0;
}

```

Nëse blloku programor përbëhet nga një urdhër i vetëm atëherë përdorimi i kllapave {} nuk është i domosdoshëm.

Ushtrime:

- Të shkruhet program i cili cakton shumën e shifrave të një numri;
- Të shkruhet program i cili për numrin e dhënë n do të paraqesë yjet ne ekran në formën:

Shembull per n=5:

```

*
**
***
****
*****

```

- Të shkruhet program i cili gjen PMP dhe SHVP e dy numrave

4.2.3. Struktura për përsëritje **for**

Urdhëri **for** mundëson përsëritjen e disa urdhëra me numërim të cikleve

Forma e përgjithshme e ciklit **for** është:

```
for (inicializimi; kushti; ndryshimi)
{
    blloku_i_urdhërave ;
}
```

Çdo cikël **for** ka variablën e tij kontrolluese, vlera e së cilës ndryshohet me çdo ekzekutim të ciklit dhe ky ndryshim është i përcaktuar me vlerën shtuese (kjo vlerë mund të jetë edhe negative). Kushti duhet të jetë shprehje logjike e cila mund të marrë vlerë të *saktë (true, 1)* ose të *pasaktë (false, 0)*. Blloku i komandave i cili gjendet në cikël ekzekutohet derisa vlera e kushtit është e saktë. Kur kushti do të marrë vlerë të pasaktë, cikli do të ndërpritet (përfundon).

Shembull: Të paraqiten në ekran të gjithë numrat prej 1 deri në n dhe të gjendet shuma e tyre të cilët janë të plottëpjeshtueshëm me 3 ose janë çift.

```
#include<iostream>
using namespace std;
int main()
{
    int n,i,shuma=0;
    cout<<"shkruaj vleren per n:";
    cin>>n;
    for(int i=1;i<=n;i++)
        if(i%3==0 && i %2==0)
        {
            cout<<i<<endl;
            suma+=i;
        }
    cout<<"shuma e numrave cift dhe te plotepjestueshem me 3 eshte "<<suma;

    return 0;
}
```

4.3. Urdhërat **break** dhe **continue**

Me komandën **break** ndalohet ekzekutimi i më tutjeshëm i ciklit në të cilin gjendet kjo komandë. Këtë komandë tanimë e kemi parë edhe në strukturën *switch-case*. Me komandën **continue** ndërprehet vetëm ekzekutimi i iteracionit momental i ciklit në të cilin gjendet kjo komandë (ai iteracion tejkalohet).

Shembull: Të gjendet nëse shuma e numrave në intervalin $[m,n]$ e kalon numrin e dhënë S ; dhe nëse e kalon sa numra duhet të mbledhen deri tek numri $x < n$, që të plotësohet kushti.

Numrat të cilët do ti konsiderojmë janë të gjithë numrat në intervalin $[m,n]$, duke përdorur urdhërin `continue` për numrat të cilët shifrën e fundit e kanë 0 nuk do ti shtojmë në ndryshoren në të cilën do të ruajmë shumën e numrave. Ndërsa përmes urdhërit `break`, nëse shuma e numrave të cilët e plotësojnë kushtin e ka tejkaluar ose është barazuar me numrin e dhënë S atëherë cikli duhet të mbarojë gjegjësisht të përdorim urdhërin `break`;

```
#include<iostream>
using namespace std;
int main()
{
    int m,n,S,numri,shuma;
    bool plotesohet=false; //tregon nese plotesohet kushti qe shuma e numrave
te jet me e
                                //madhe se vlere S
    cout<<"Shkruaj fillimin e intervalit m="; cin>>m;
    cout<<"Shkruaj fundin e intervalit n=";   cin>>n;
    cout<<"Shkruaj kufirin e shumes S=";   cin>>S;
    numri=1;
    shuma=0;
    for(int i=m;i<=n;i++)
    {
        if(i%10==0)
            continue;
        shuma+=i;
        numri+=1;
        if (shuma>S)
        {
            plotesohet=true;
            break;
        }
    }

    if (plotesohet)
        cout<<"Me "<<numri<<" numrat e pare plotesohet shuma qe tejkalon
vleren "<<S;
    else
        cout<<"Shuma e numrave ne intervalin ["<<m<<","<<n<<"] eshte me vogel
se vlere e dhene "<<S;
    return 0;
}
```

Ushtrim: Çfarë do të paraqesë në ekran kodi i dhënë:

```
int count=0;
while (count < 10)
{
    if (count == 5)
        continue;
    cout << count << " ";
    ++count;
}
```

Detyra:

1. Të paraqitet tabela e shumëzimit për numrin e dhënë n
2. Të caktohet numri i dhënë x a është l thjeshtë ose jo (numër l thjeshtë është numri l cili ka vetëm dy pjesues numrin 1 dhe vetveten p.sh numri 5)
3. Të gjendet maksimumi/minimumi nga n numrat e dhënë nga shfrytëzuesi
4. Të caktohet shuma e numrave të dhënë me formulën $\sum_{i=1}^n 4(i + x)$, x dhe n jipen me tastier
5. Te caktohet prodhimi i numrave $\prod_{i=1}^{m+n} \frac{2(i+k)}{3k}$, $m, n, k \neq 0$ jipen me tastier
6. Të caktohen të gjithë numrat e plotë a, b dhe c , ku $c \leq n$, për të cilët vlen: $a^2 + b^2 = c^2$.
7. Të paraqitnen në ekran shifrat e numrit
8. Të gjendet numri i kundërt i numrit x

5. FUNKSIONET, FUNKSIONET ANËTARE TË KLASËS

5.1. Funksionet standarde në C++

Gjatë programimit hasim me probleme të ndryshme në të cilat duhet llogaritje më të komplikua se operacionet themelore aritmetike (+, -, *, /, %). Nëse duhet të llogaritet vlera absolute, katrori, kubi, rrënja katrore dhe funksione tjera të cilat përpunojnë të dhëna; gjuha programuese C++ mundëson shfrytëzimin e funksioneve të definuara të cilat mundësojnë llogaritje të vlerave për funksione të ndryshme.

Funksionet të cilat përdoren më së shpeshti gjatë programimit dhe janë të përfshira në C++ quhen **funksione standard**. Këto funksione janë pjesë përbërëse e Gjuhës programuese C++ dhe përdoren gjatë shkruarjes së programit pa pasur nevojë që ti deklararojmë ose definojmë sjelljen e tyre.

Detyra e funksionit është që të llogarisë dhe kthejë vlerë e cila kërkohet nga programuesi. Në shembullin llogaritjes se shprehjes: $X = A^3 - B^3$ për të shkruar urdhërin në gjuhën programuese C++ e shkruajmë në formën: $X = A * A * A - B * B * B$, edhe pse rezultati gjatë llogaritjes është i saktë kodi i programit përmes funksionit bëhet më i kuptueshëm për lexuesin. Nëse e shkruajmë si $X = \text{pow}(A, 3) - \text{pow}(B, 3)$ do të jetë më e lehtë për programuesit për ta shkruar dhe kuptuar kodin. Gjatë programimit mund të hasim në operacione si rrënja katrore, logaritmit, sinusit, rrumbullaksimi i numrave, kopjimi i stringut etj, do të ishte shumë më e vështirë programimi i operacioneve të tilla.

Funksionet e shfrytëzohen (thirren) përmes emrit të tyre dhe parametrave të funksionit në kllapa. Forma e përgjithshme e thirrjes së një funksioni është:

ndryshorja = EmriFunksionit(arg1, arg2, ..., argN);

Libraria me funksione standarde mundëson përdorimin e funksioneve të cilat llogarisin vlerat e dëshiruara nga programuesi. Shprehjen 4^2 në gjuhën programuese C++ mund ta shkruajmë si: $\text{pow}(4, 2)$; Emri i funksionit është shkurtës **pow**, funksioni ka vetëm një parametër (numër i plotë) 4, llogarit vlerën $4^2 = 16$ dhe kthen vlerë 16 numër i plotë.

Shembull të caktohet sipërfaqja e katrorit, rrethit, nëse janë dhënë brinja dhe rrezja në mënyrë përkatëse. **Për shfrytëzimin e funksionit duhet të përfshihet biblioteka përkatëse në të cilën është definuar funksioni, gjegjësisht emri, lista me tipet e parametrave dhe tipi kthyes i funksionit;**

Shembull për përdorimin e funksioneve matematikore si **abs**, **pow**, **sqrt**, dhe funksionet tjera matematikore duhet të përfshihet biblioteka **cmath** në të cilën janë të definuara funksionet.

```
#include<iostream>
```

```
#include<cmath>
using namespace std;
const double PI =3.141592653589793238463;
int main()
{
    double a=3.2, r=4.4,sk,sr;
    sk=pow(a,2);
    sr=pow(r,2)*PI;
    cout<<"Siperfaqja e katrorit eshte "<<sk<<endl;
    cout<<"Siperfaqja e rrethit eshte "<<sr;
    return 0;
}
```

Funksioni	Prototipi	Qëllimi
abs(x)	int abs(int x);	Kthen vlerën absolute në numrit të plotë
fabs(x)	double fabs(double x);	Kthen vlerën absolute në numrit dhjetor
ceil(x)	double ceil(double x);	Rrumbullakson në numri më të madh cout<<ceil(11.2); (në ekran paraqet numrin 12)
floor(x)	double floor(double x);	Rrumbullakson në numri më të vogël cout<<floor(11.5); (në ekran paraqet numrin 11)
sin(x)	double sin(double x);	Njihëson sinusin e këndët të shprehur në radian.
pow(x,y)	double pow(double x, double y);	Njihson x në fuqi y. Nëse x është negativ y duhet të jetë numër i plotë. Nëse x është 0 y duhet të jetë numër i plotë pozitiv.
pow10(x)	double pow10(int x);	Njihëson 10 në fuqi x.
sqrt(x)	double sqrt(double x);	Njihëson rrënjën katrore të x. (x >=0)
strlen	unsigned strlen (char *x)	Kthen numër të plotë pozitiv gjatësinë e vargut tekstual
strcmp	int strcmp (const char * str1, const char * str2);	Kthen 0 nëse str1 dhe str 2 kanë vlera të njejta,<0 nëse karakteri i parë i cili nuk përputhet është më i vogël në str1 se në str2,
atoi	int atoi (const char * str);	Konverton string në integer
rand	int rand (void);	Gjeneron numër të rastësishëm në intervalin 0..RAND_MAX

Shembulli 1: Të krahasohen gjatësitë e dy vargjeve tekstuale dhe nëse janë të barabarta të krahasohet përmbajtja e tyre a janë të njejta.

```
#include<iostream>
```

```
#include<string.h>
using namespace std;
int main()
{
    char text1[60], text2[60];
    cout<<"shkruaj tekstin e pare: ";
    cin>>text1;
    cout<<"shkruaj tekstin e dyte: ";
    cin>>text2;
    if (strlen(text1)==strlen(text2))
    {
        if(strcmp(text1,text2)==0)
            cout<<"tekstet kane permbajtje te njejte ";
        else
            cout<<"tekstet kane gjatesi te njejte permbajtje te ndryshme ";
    }
    else
        cout<<"tekstet nuk kane gjatesi te njejte ";
    return 0;
}
```

5.2. Definimi i funksioneve jostandarde

C + + nuk ofron të gjitha funksionet e mundshme që i nevojiten përdoruesit, sepse nevojat e secilit përdorues mund të jenë të ndryshme dhe specifike, prandaj për këtë arsye duhet të shkruhen funksionet e caktuara nga përdoruesi. Funksionet e caktuara nga përdoruesi, në C++ klasifikohen në dy kategori:

- Funksione që kthejnë vlerë, këto funksione kanë një tip kthyes dhe duke përdorur deklaratën **return** e kthejnë rezultatin e llojit të caktuar të të dhënave.
- Funksione boshe (që nuk kthejnë vlerë), këto funksione nuk kanë tip kthyes të të dhënave. Këto funksione nuk e përdorin deklaratën **return** për të kthyer rezultat.

Sintaksa e deklarimit të një funksioni të caktuar nga përdoruesi dhe i cili kthen rezultat është si në vijim:

```
tipi emri_funksionit(tipi1 par1, tipi2 par2 ...)
{
    deklarimet dhe urdhërat;
    return shprehja;//opsionale
}
```

- emri_funksionit – secili identifikator valid
- tipi – tipi i të dhënave për funksionin/rezultatit
- tipi1, tipi2 – tipi i të dhënave për parametrat

- par1, par2 – emrat e parametrave

Thirrja e funksionit brenda bllokut kryesor programor main bëhet përmes emrit dhe listës së parametrave.

ndryshorja=emri_funksionit(ndryshorja1,ndryshorja2...); //funksionet me tip

Parametrat e shënuar brenda kllapave në titullin e funksionit gjatë definimit të tij, quhen parametra formalë, sepse përmes tyre tregohet forma e llogaritjeve që kryhen brenda funksionit. Kurse, parametrat me të cilët zëvendësohen ato formalë gjatë thirrjes së nënprogramit, siç u përmend edhe më sipër, quhen parametra aktualë.

Parametrat formalë dhe ato aktualë duhet të përputhen mes vete për nga:

- **numri** - sa ka parametra formalë atë duhet të ketë edhe parametra aktualë;
- **tipi** - tipin e njëjtë duhet ta kenë parametrat formalë dhe parametrat aktualë përkatës;
- **radha e shkruarjes** - parametrat formalë në pozitë të caktuar i përgjigjet parametër aktual me pozitë të njëjtë.

Ushtrim: Të shkruhet dhe thirret në bllokun main funksioni i cili cakton vlerën x^n , ku x dhe n janë parametrat e funksionit.

```
#include<iostream>
using namespace std;
int fuqizimi(int x,int n)
{
    int rezultati=1;
    for(int i=1;i<=n;i++)
        rezultati*=x;
    return rezultati;
}

int main()
{
    int b,e,rezultati;
    cout<<"shkruaj bazen:  ";
    cin>>b;
    cout<<"shkruaj eksponentin:  ";
    cin>>e;
    rezultati=fuqizimi(b,e);
    cout<<b<<" ne fuqi "<<e<<" = "<<rezultati;
    return 0;
}
```

Çfarë ndodhë gjatë thirrjes së funksionit: Parametrat e funksionit marrin vlerën e ndryshoreve nga blloku kryesor, në funksionin **fuqizimi** ndryshorja **x** merr vlerën e ndryshores **b**, ndryshorja **n** merr vlerën e ndryshores **e**, pas kryerjes së operacineve funksioni kthen vlerën e ndryshores (rezultatit të funksionit) me **return rezultati**; ku dhe mbaron funksioni.

Shembulli 2: Të shkruhet funksioni i cili cakton sipërfaqen e trekëndëshi nëse janë dhënë gjatësitë e brinjëve, me formulën e Heronit: $S = \sqrt{s(s-a)(s-b)(s-c)}$ ku a,b,c brinjët e trekëndëshit, dhe s- gjysëmperimetri i trekëndëshit $s=(a+b+c)/2$.

```
#include<iostream>
#include<math.h>
using namespace std;

float siperfaqja(int a,int b, int c)
{
    float s=(a+b+c)/2.0;
    float S=sqrt(s*(s-a)*(s-b)*(s-c));

    return S;
}
int main()
{
    int a,b,c;
    cout<<"shkruaj gjatesine per brinjen a: ";
    cin>>a;
    cout<<"shkruaj gjatesine per brinjen b: ";
    cin>>b;
    cout<<"shkruaj gjatesine per brinjen c ";
    cin>>c;
    cout<<"Siperfaqja e trekendeshit eshte: "<<siperfaqja(a,b,c);
    return 0;
}
```

Funksionet e definuara mund të mos kthejnë vlerë por vetëm të kryejnë detyra të caktuara, sikurse paraqitjen e të dhënave në ekran ose ndryshimin e parametrave. Funksionet të tilla nuk kanë vlerë kthyesë ose tip kthyes të ndryshores dhe quhen funksione pa tip ose void;

```
void emri_funksionit(tipi1 par1, tipi2 par2...)
{
    deklarimet dhe urdhërat;
}
```

- emri_funksionit – secili identifikator valid
- tipi1, tipi2 – tipi i të dhënave për parametrat
- par1, par2 – emrat e parametrave

Thirrja e funksionit brenda bllokut kryesor programor main bëhet përmes emrit dhe listës së parametrave, për dallim nga funksionet të cilat kthejnë vlerë funksioni void nuk i jipet ndryshore.

```
emri_funksionit(ndryshorja1, ndryshorja2...);
```

Shembull të shkruhet funksioni i cili paraqet në ekran shumën, ndryshimin, prodhimin dhe herësin e dy numrave të plotë;

```
#include<iostream>
using namespace std;
int Z;
void operacionetAritmetike(int a,int b)
{
    cout<<"operacionet elemntare aritmetike"<<endl;
    cout<<a<<" + "<<b<<" = "<<a+b<<endl;
    cout<<a<<" - "<<b<<" = "<<a-b<<endl;
    cout<<a<<" * "<<b<<" = "<<a*b<<endl;
    cout<<a<<" / "<<b<<" = "<<a/float(b)<<endl;
}

int main()
{
    int a,b;
    cout<<" shkruaj vleren per operatorin e pare: ";
    cin>>a;
    cout<<" shkruaj vleren per operatorin e dyte: ";
    cin>>b;
    operacionetAritmetike(a,b);
    return 0;
}
```

5.3. Ndryshoret lokale dhe globale

Shtrirja globale dhe lokale çdo gjë që është definuar (deklaruar) në nivel të shtrirjes së programit (p.sh. jashtë funksioneve dhe klasave) është e thënë të ketë një shtrirje globale. Kështu që funksionet e thjeshta që kemi përdorur deri tani të gjithë kanë shtrirje globale. Variablat gjithashtu mund të definohen në shtrirje globale:

```
int viti = 1994; // ndryshore globale
int Max (int, int); // funksion global
int main (void) // funksion global
{
    //...
}
```

Variablat globale të pa inicializuara, automatikisht janë të inicializuara me vlerën zero. Që kur entitetet globale janë të dukshme në nivelin programit, ata gjithashtu duhet të jenë unike (të vetme) në atë nivel të programit. Kjo do të thotë se variabla apo funksione të njëjta nuk mund të definohen më shumë se një herë në nivel global. Entitetet globale janë përgjithësisht të

qasshme kudo në program. Çdo bllok në program përcakton një *shtrirje (fushë) lokale*. Pra trupi i një funksioni prezanton një shtrirje lokale. Parametrat e funksionit kanë shtrirje të njëjtë si trupi i funksionit. Variablat e definuar në fushën lokale janë të qasshme vetëm në atë fushë. Kështu që një ndryshore (variabël) duhet të jetë unike (e vetme) vetëm brenda fushës (shtrirjes) së vetë. Fushat lokale mund të jenë të mbivendosura, në të cilin rast fushat e brendshme i refuzojnë fushat e jashtme. Për shembull:

```
int xyz; // xyz është ndryshore globale
void ABC (int xyz) // xyz është lokale në trupin e funksionit ABC
{
if (xyz > 0)
{
double xyz; // xyz është lokale në këtë bllok
//...
}
}
```

ka tre fusha të ndryshme, dhe secila fushë përmban nga një ndryshore xyz të ndryshme. Në përgjithësi jetëgjatësia e një ndryshore është e kufizuar brenda fushës ku ajo përfshihet. Kështu, për shembull, ndryshoret globale zgjat me kohëzgjatjen e ekzekutimit të programit, kurse ndryshoret lokale janë krijuar kur fusha e tyre ka filluar dhe shkatërrohen (fshihen) kur fusha e tyre të ketë përfunduar. Hapësira e memories për ndryshoret globale është e rezervuar para se të fillon ekzekutimi i programit, kurse hapësira e memories për ndryshoret lokale është e ndarë në fly për gjatë ekzekutimit të programit.

Operatori i fushës (shtrirjes) Nga që fusha lokale e refuzon fushën globale, një variabël lokale me emër të njëjtë si një variabël globale e bën këtë të fundit të pa qasshme në fushën lokale. Për shembull në

```
int error;
void Error (int error)
{
//...
}
```

ndryshorja globale error është e pa qasëshme brenda funksionit Error, sepse është e kufizuar nga ndryshorja lokale error. Ky problem është i zgjidhur duke përdorur operatorin unar të fushës (shtrirjes, hapësirës) :: që merr një entitet global si argument.

```
int error;
void Error (int error)
{
//...
if (::error != 0) // I referohet ndryshores globale
//...
}
```

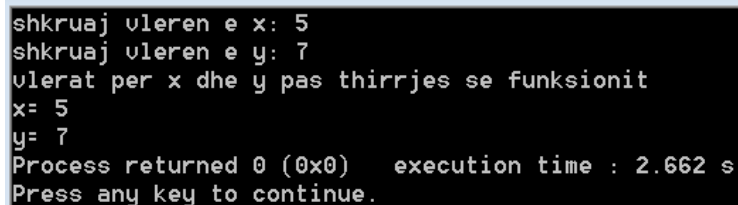
tani më përmes operatorit :: (katër pika) mund ti qasemi ndryshores globale error.

5.4. Parametrat e funksionit sipas vlerës, referencës dhe adresës

Me deklarimin dhe thirrjen e funksioneve të mësipërme, parametrat formal marrin vlerat e parametrave aktual. Në trupin e funksionit parametrat formal mund të ndryshojnë vlerat e tyre por kjo nuk do të thotë se ndryshimi i vlerave vlen edhe për parametrat aktual. Shembull mund të marrim funksioni tip shkembe(int a, int b); tek i cili parametrat a dhe b shkëmbyjnë vlerat mes veti.

```
#include<iostream>
using namespace std;
void shkembe(int a,int b){
    int nd=a;
    a=b;
    b=nd;
}
int main(){
    int x,y;
    cout<<"shkruaj vleren e x: ";
    cin>>x;
    cout<<"shkruaj vleren e y: ";
    cin>>y;
    shkembe(x,y);
    cout<<"x= "<<x<<"\ny= "<<y;
    return 0;
}
```

Programi i egzekutuar nuk do të japë vrezultatit e pritur:



```
shkruaj vleren e x: 5
shkruaj vleren e y: 7
vlerat per x dhe y pas thirrjes se funksionit
x= 5
y= 7
Process returned 0 (0x0)   execution time : 2.662 s
Press any key to continue.
```

5.4.1. Kalimi i parametrave me reference

C++ mundëson që parametrat aktual në funkskon të zëvendësohen me ato aktual dhe ndryshimet në parametrat formal në trupin e funksionit të vlejnë në parametrat aktual.

Te kalimi me reference argumentet formal janë referencë e argumenteve aktual dhe ndryshimet në argumentet formal në fakt janë ndryshimet në argumentet aktual.

Definimi i funksioneve me kalim të parametrave me referencë bëhet ashtu që në mes të tipit dhe para parametrin formal vendoset shenja &.

Shembull

```
#include<iostream>
using namespace std;
void shkembe(int & a,int & b){
    int nd=a;
    a=b;
    b=nd;
}
int main(){
    int x,y;
    cout<<"shkruaj vleren e x: ";
    cin>>x;
    cout<<"shkruaj vleren e y: ";
    cin>>y;
    shkembe(x,y);
    cout<<"x= "<<x<<"\ny= "<<y;
    return 0;
}
```

5.4.2. Kalimi me parametrave me adresë

Kalimi i parametrave me adresë përfshin kalimin e adresës së ndryshores si parametër e jo vetë ndryshores si parametër. Pasi që parametri është adresë, atëherë edhe parametri i funksionit është tregues. Funksioni mund ta dereferencojë treguesin që të ketë qasje ose ndryshojë vlerën të e cila tregohet. Definimi dhe thirrja e funksionit bëhet si në shembullin në vijim:

```
#include<iostream>
using namespace std;
void shkembe(int * a,int * b){ //definimi i funksionit shkembe
    int nd=*a;
    *a=*b;
    *b=nd;
}
int main(){
    int x,y;
    cout<<"shkruaj vleren e x: ";
    cin>>x;
    cout<<"shkruaj vleren e y: ";
    cin>>y;
    shkembe(&x,&y); //thirrja e funksionit shkembe
}
```

```

    cout<<"x= "<<x<<"\ny= "<<y;
    return 0;
}

```

5.5. Mbingarkimi i funksionieve

Brenda një programi mund të definohen njëkohësisht disa funksione me emra të njëjtë, por me parametra të ndryshëm për nga numri ose të ndryshëm për nga tipet e tyre. Funksionet e tilla njihen si funksione të mbingarkuar (ang. overloaded functions). Nuk mund të mbingarkohet funksioni nga tipi i rezultatit kthyes.

Shembull: Mbingarkimi i funksionit shuma, i cili cakton shumën e dy numrave të plotë (int) dhe dy numrave dhjetor (float).

```

#include<iostream>
using namespace std;
int shuma(int a, int b){
    return a+b;
}

float shuma(float a, float b){
    return a+b;
}

int main()
{
    int x=7,y=3;
    float z=3.5,w=5.5;
    //thirrja e funksionit int shuma(int a, int b)
    cout<<" shuma e numrave "<<x<<" dhe "<<y<<" eshte "<<shuma(x,y);
    //thirrja e funksionit float shuma(float a, float b)
    cout<<" shuma e numrave "<<z<<" dhe "<<w<<" eshte "<<shuma(z,w);
}

```

Shembull: Mbingarkimi i funksionit sipërfaqja, i cili cakton shumën e katrorit dhe drejtëkëndëshit.

```

#include<iostream>
using namespace std;

int sipërfaqja(int a, int b){
    return a*b;
}
int sipërfaqja(int a){
    return a*a;
}

int main()
{
    int a,b;
}

```

```

cout<<"shkruaj gjatesine e brinjes se katrorit" ;
cin>>a;
cout<<"siperfaqja e katrorit eshte "<<siperfaqja(a);

cout<<"shkruaj gjatesine e brinjes a te drejtekendeshit: " ;
cin>>a;
cout<<"shkruaj gjatesine e brinjes b te drejtekendeshit: " ;
cin>>b;
cout<<"siperfaqja e drejtekendeshit eshte "<<siperfaqja(a,b);
}

```

Ushtrime

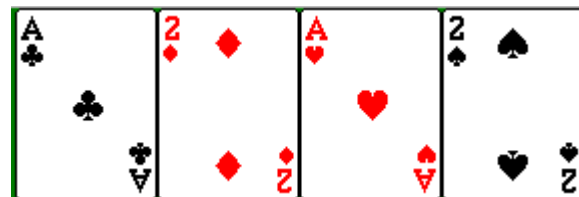
1. Të shkruhet funksioni i cili gjen maksimumin nga tre numra të dhënë
2. Të shkruhet funksioni i cili gjen prodhimin e shifrave të një numri të dhënë x.
3. Të shkruhet funksioni i cili paraqet në ekran numrat çift të plotëpjestueshëm me 3;
4. Të shkruhet funksioni i cili përcakton se parametri i dhënë n është i thjeshtë ose jo, duke e shfrytëzuar këtë funksion të paraqiten në ekran numrat e thjeshtë prej 1 deri ne numrin e dhënë x.
5. Të shkruhet funksioni i cili llogarit SHVP (shumëfishin më të vogël të përbashkët) e dy numrave
6. Të shkruhet funksioni i cili llogarit PMP (pjestuesin më të madh të përbashkët) e dy numrave
7. Te shkruhet funksioni i cili gjeneron vargun e Fibonaçit

6. VARGJET DHE VEKTORËT

6.1. Kuptimi për vargun



Një varg i avionëve



Një varg i letrave

Fig 7. Vargjet

Nëse shikoni gjërat e cilit do grup që janë paraqitur më sipër, ti e kupton se sendet në figurë kanë disa karakteristika të njejta, edhe pse secili prej tyre kanë tipare specifike që e veçojnë atë nga të tjerët. Secili nga elementet në fotografin e parë është një aeroplan, në qoftë se ju vendosni të jenë specifik, atëherë ju mund të thoni se aeroplani i parë i grupit është e gjelbër e ndritshme, ndërsa e dyta është e zezë, aeroplanët e parë dhe të katërt nuk kanë helik edhe pse të gjithë të tjerët bëjnë.

Nëse ju keni luajtur letra ndonjëherë atëherë ju jeni të njohur me fotografin e dytë. Secili nga elementet në fotografin e dytë është një kartë, madhësia e njëjtë, të njëjtin sfond të bardhë, megjithëse ato shfaqin vlera të ndryshme, ngjyra të ndryshme të karakterit (edhe ata do të kenë efekte të ndryshme në varësi se si loja juaj është duke

shkuar). Shembulli i vargut është edhe lista lëndëve por edhe lista e notave të ndara veç e veç;

Gjuhë shqipe	Gjuhë angleze	Matematikë	Fizikë	Informatike	Biologji	Gjeografi	Sport
5	4	5	4	5	3	3	5

Një varg është një grup i artikujve që mund të identifikohen si të ngjashme, sepse ato janë të natyrës së njëjtë. Në programim vargjet janë një seri e objekteve që janë me të njëjtën madhësi dhe lloj.

Çdo objekt në një varg quhet element i vargut. Për shembull, ju mund të keni një grup të numrave të plote (int), ose një grup të karaktereve (char) ose një grup nga çdo gjë që ka një lloj të dhënave të definuar.

Karakteristikat e rëndësishme të një vargu janë:

- Çdo element ka të njëjtin lloj të të dhënave (edhe pse ata mund të kenë vlera të ndryshme)
- I tërë vargu është ruajtur pranë njëri tjetrit në kujtesë (që do të thotë nuk ka boshllëqe në mes të elementeve).

Vargjet mund të ketë më shumë se një dimension. Vargu një dimensional quhet varg.

6.1.1. Inicializimi dhe deklarimi i vargut

Forma e përgjithshme e deklarimit të vargut është:

```
tipi emriVargut [numriElementeve];
```

Me deklarimin e tillëtë vargut, në memorjen e kompjuterit rezervohet hapsirë për `numriElementeve` të tipit `tipi` e emërtuar si `emriVargut`;

Çdo element në varg është përcaktuar me indeksin (pozicionin) e tij në varg. Elementi i parë ka indeksin 0 dhe mund që ti qasemi sikurse ndryshoreve të tipit të dhënë `emriVargut[0]`; elementi i dytë `emriVargut[1]` deri tek elementi i fundit `emriVargut[numriElementeve-1]`.

Vargjet mund të deklarohen dhe të inicializohen njëkohësisht. Kështu, p.sh., deklarimi dhe inicializimi i vargut `vargut` me 5 anëtarë të tipit integer (tip numër i plotë) duket:

```
int vargu[5]={73,62,51,42,41};
```

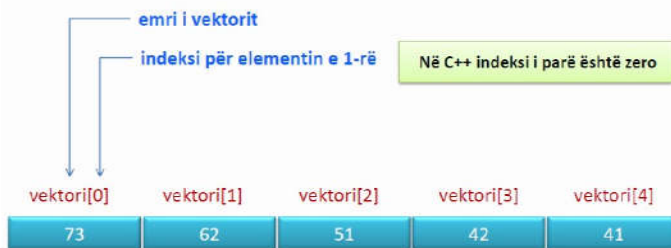


Fig. 48. Deklarimi dhe inicializimi i vargut

Nëse deklarojmë dhe inicializojmë varg ashtuqë gjatësia e definuar e vargut është më e madhe se numri i elementeve atëherë pjesa e mbeture e elementeve do të inicializohen me vlerë 0. Pas deklarimit të vargut, duhet të gjejm një mënyrë për t'ju drejtuar elementeve të tij. Kjo realizohet me ndihmën e indekseve, numrit brenda kllapave katrore `[]` që vjen menjëher pas emrit të vargut. Indeksi i elementit të parë të vargut është 0, kështu që indeksi i elementit të fundit është një më i vogël se numri i elementeve të vargut.

```
int x[6]={1,3,5}
```

vlerat e elementeve të vargut `x` do të jenë: `x[0]=1` `x[1]=3`, `x[2]=5`, `x[3]=0`, `x[4]=0`, `x[5]=0`

Shembull: Te inicializohet vargu `A` me `n` elemente ku çdo element është dyfishi i indeksit të tij dhe të paraqitet në ekran

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



```

{
    int n;
    cout<<"shkruaj dimensionin e vargut <100: ";
    cin>>n;
    int A[n];
    for(int i=0;i<n;i++)
    {
        a[i]=2*i;
        cout<<a[i]<<endl;
    }
    return 0;
}

```

Shembulli 2: Në vargun e dhënë me vlera numër i plotë gjeneruara me funksionin rand të gjendet elementi me vlerë maksimale dhe pozicioni (indeksi) i tij në varg.

```

#include<iostream>
#include <stdlib.h>
using namespace std;
int main()
{
    int a[100],vMax,n,indeksi;
    cout<<"shkruaj gjatesine e vargut:";
    cin>>n;
    for(int i=0;i<n;i++)
    {
        a[i]=rand();
    }
    vMax=a[0];
    indeksi=0;
    for(int i=1;i<n;i++)
    if(a[i]>vMax)
    {
        vMax=a[i];
        indeksi=i;
    }

    cout<<"vargu i gjeneruar "<<endl;
    for(int i=0;i<n;i++)
        cout<<"a["<<i<<"]="<<a[i]<<endl;
    cout<<"vlera maksimale e gjeneruar ne varg eshte "<<vMax<<" ne
    poziten"<<indeksi;
    return 0;
}

```

6.1.2. Vargu si parametër i funksionit

C++ nuk lejon që tërë vargu të kaolohet si parametër i funksionit. Por mund të kalohet treguesi i cili i referohet emrit të vargut.

Kalimi i mund të bëhet në tri mënyra sikurse në shembullin e mëposhtëm , dhe të gjitha deklaratimet e funksionit japin rezultatin e njëjtë.

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

void printo(int *a,int l)
{
    for(int i=0;i<l;i++)
    {
        cout<<a[i]<<" ";
    }
    cout<<"\n";
}

void inicializo(int a[],int n)
{
    for(int i=0;i<n;i++)
    {
        cout<<" a["<<i<<"]=";
        cin>>a[i];
    }
}

float mesatarja (int a[3],int n)
{
    float mesatarja;
    int shuma;
    cout<<"n= "<<n<<endl;
    for (int i=0;i<n;i++)
        shuma+=a[i];
    mesatarja=shuma/(n*1.0);
    return mesatarja;
}

int main()
{
    int n;
    cout<<"shkruaj gjatesine e vargut: ";
    cin>>n;
    int b[n];
    inicializo(b,n);
    cout<<"Vlerat e vargut b\n";
    printo(b,n);
    cout<<"Vlera mesatare e vargut eshte"<<mesatarja(b,n);
    return 0;
}
```

Detyra

1. Të gjendet shuma dhe prodhimi i elementeve të vargut të dhënë a;
2. Të gjendet mesatarja e elementeve të vargut a
3. Të gjendet mesatryja dhe numri i elementeve çift në vargun a
4. Të gjenerohet vargu a me vlerat $a[i] = \frac{2(x-1)}{3k}$, vlerat x dhe k jipen me tastier, $k \neq 0$.
5. Në vargun e dhënë a të caktohet a gjendet vlera e dhënë x, në cilat pozicione dhe sa herës.
6. Në vargun e dhënë a nëse është i radhitur të caktohet a e përmbanë vlerën e dhënë x.
7. Të gjendet prodhimi shuma e vargeve a dhe b.
8. Të krijohet vargu D i cili përbëhet nga elementet e vargëve A, B dhe C.
9. Të gjendet sa herë ndryshon shenja nga + në – ose nga – në + për vlerat e vargut a. shembull në vargun a me vlerat {3,12,-4,-7,2,9,5,-3} shenja ndryshon 3 herë (12,-4), (-7,2) dhe (5,-3).
10. Të gjendet a janë të barbartë vargjest a dhe b(a dhe b kanë të gjitha vlerat e njejta dhe posedojnë numër të njejtë të elementeve);
11. Të gjendet nëse vargu a e përmban vargun b shembull $a=\{3,5,6,7,8,9,12\}$, $b=\{6,7,8\}$ atëherë thuhet se vargu **a** e përmban vargun **b**.

6.2. Kompleksiteti kohor dhe memorues e algoritmeve

Në shkenca kompjuterike, analiza e algoritmeve është caktimi i resurseve (koha dhe hapsira memoruese) e nevojshme për egzekutimin e tyre. Shumica e algoritmeve dizajnohen që të punojnë me gjatësi të papërcaktuar. Zakonisht, efikasiteti i kohës së egzekutimit llogaritet si funksion i numrit të hapave (operacioneve), varësisht prej gjatësisë (sasisës) së të dhënave hyrëse (kompleksiteti kohor) ose memorja për ruajtje (kompleksiteti i memorues).

Kompleksiteti kohor i algoritmeve shprehet me O notacionin (big O notation).

Kompleksiteti konstant: Shpejtësia e egzekutimit të urdhërit (bashkësisë së urdhërave) është konstante dhe nuk varet nga të gjatësia e të dhënave hyrëse;

Urdhëri;

Kompleksiteti linear $O(n)$: Shpejtësia e egzekutimit të urdhërit (bashkësisë së urdhërave) është proporcional me N , nëse n dyfishohet atëherë dyfishohet koha e egzekutimit;

```
for(int i=1;i<n;i++)
    Urdhëri;
```

Kompleksitet katror $O(n^2)$: Koha e egzekutimit të dy cikleve është proporcional me katrorin e N , nëse dyfishohet N koha rritet për $N \cdot N$

```
for( int i=1;i<=n;i++)
    for(j=1;j<=n; j++)
        urdhëri;
```

Kompleksiteti logaritmik $O(\log n)$: koha e egzekutimit të algoritmit është proporcional me numrin me të cilin N mund të pjestohet me 2; arsyeja është sepse algoritmi e ndan sipërfaqen punuese në dy pjesë me çdo iteracion;

```
while ( ulet <= larte ) {
    m = ( ulet + larte ) / 2;

    if ( qellimi < list[mesi] )
        larte= m - 1;
    else if ( qellimi > list[mesi] )
        ulet= m + 1;
    else break;
}
```

Kompleksiteti linear-logaritmik $N \cdot \log(n)$ përbëhet nga N cikle të cilët janë me kompleksitet logaritmik;

Sipas kompleksitetit të algoritmeve dallojmë:

- Konstant $O(1)$
- Logaritmik $O(\log(n))$
- Linear $O(n)$
- Linear-logaritmik $O(n \cdot \log_2 n)$
- Katror $O(n^2)$
- Exponencial $O(2^n)$
- Faktorial $O(n!)$

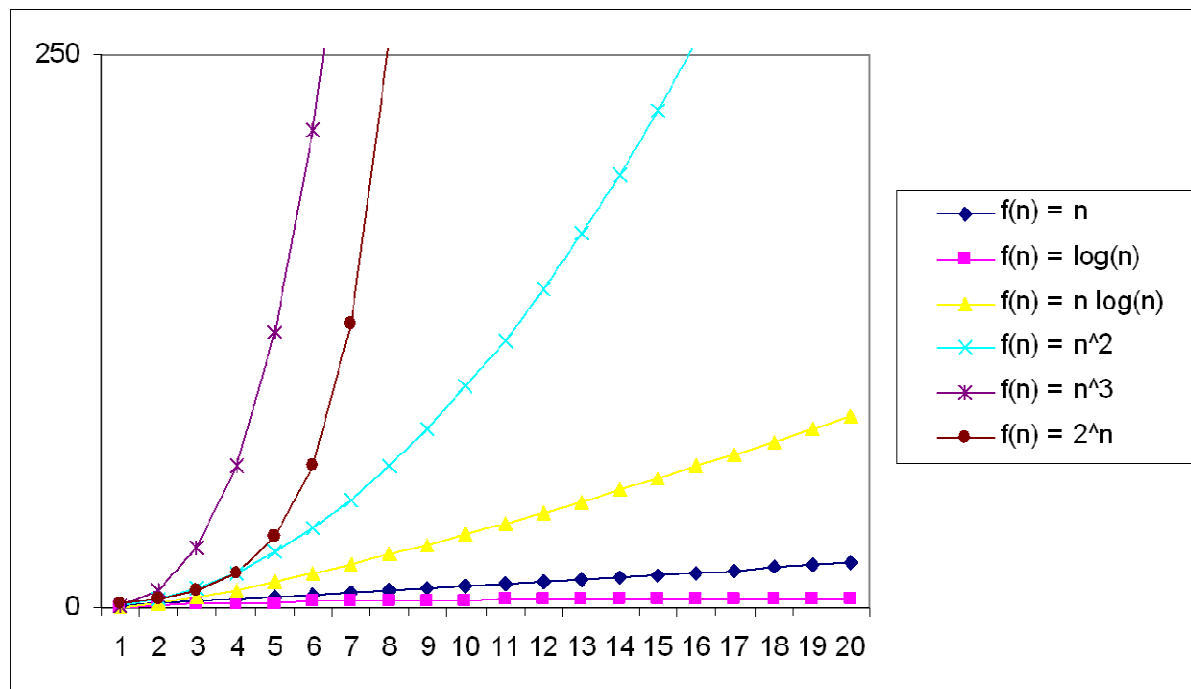


Fig 8. Grafiku i kompleksitetit kohor

6.3. Kërkimi në varg

Procesi i gjetjes së të dhënave të kërkuara në një varg quhet kërkim. Teknikat më të njohura për kërkim në varg janë:

1. Kërkimi linear (*Linear search or sequential search*)
2. Kërkimi binar (*Binary Search*)

6.3.1. Kërkimi linear (Linear search)

Kjo teknikë kërkimi është shumë e thjeshtë, për të kryer këtë teknikë përdoruesi fillon ciklin (loop) nga indeksi zero te një vargu deri në indeksin e fundit. Ajo fillon nga indeksi i parë dhe krahasohet me vlerën e kërkuar me vlerën e parë.

Nëse vlera e kërkuar është gjetur ajo do të tregojë rezultat ndryshe do të krahasojë vlerën e indeksit të ardhshëm dhe kjo do të vazhdojë derisa vlera e kërkuar është gjetur ose cikli përfundon pa gjetur ndonjë vlerë.

Shembull i kërkimit linear:

```
#include<iostream>
using namespace std;
int main() {
    cout<<"Shkruaj gjatesine e vargut:  ";
    int n;
    cin>>n;
    int vargu[n], vlera,i;
    for(int j=0;j<n;j++){
        cout<<"shkruaj elementin " << j << ": ";
        cin>>vargu [j];
    }

    cout<<"Shkruaj elementin i cili duhet te kerkohet ne varg: ";
    cin>>vlera;
    bool gjendet=false;
    for(i=0;i<n;i++){
        if(vlera==vargu[i]){
            cout<<"Elementi gjendet ne indeksin numer:"<<i<<endl;
            gjendet=true;
            break;
        }
    }
    if(!gjendet)
    {
        cout<<"Numri "<<vlera<<" nuk gjendet ne varg ";
    }
    return 0;
}
```

Nëse analizojmë hapat që duhet të egzekutojë algoritmi, varësisht nga gjatësia e vargut të dhënë. Atëherë mund të kemi:

- Rasti më i shpejtë (Elementi i kërkuar ndodhet në indeksin 0): urdhëri për kushtëzim egzekutohet vetëm 1 herë. kompleksiteti kohor $O(1)$.
- Rasti më i ngadalshëm (elementi i kërkuar ndodhet në indeksin e fundit); urdhëri për kushtëzim egzekutohet n herë. Kompleksiteti kohor $O(n)$.
- Rasti mesatar $O(\frac{n}{2})$

Nga vijnë në përfundim se kompleksiteti kohor te algoritmi për kërkim linear është $O(n)$.

6.3.2. Kërkimi binar (Binary Search)

Kërkimit binar aplikohet vetëm në një varge të renditur. Ne nuk mund të aplikojmë në një varg të paradhitur. Kjo është një teknikë shumë e dobishme sepse përdoret për të gjetur shpejte vlerat e kërkuara.

Kjo teknikë kryhet në disa hapa:

1. Së pari gjendet elementi i mesëm i vargut dhe krahasohet me vlerën të cilin përdoruesi dëshiron për ta kërkuar në një varg.
2. Nëse ata janë të njëjta atëherë ajo do të kthen vendndodhjen e vlerës së kërkuar.
3. Nëse ata nuk janë të barabartë, atëherë ajo do ta ndajë vargun në gjysmë.
4. Nëse elementi i mesëm i vargut është më i madh se numri i kërkuar atëherë do të kërkojë gjysmën e parë të vargut.
5. Nëse elementi i mesëm i vargut është më i vogël se numri i kërkuar atëherë ajo do të kërkojë gjysmën e dytë të vargut.

Ky proces do të vazhdojë derisa vlera e kërkuar është gjetur ose derisa cikli (loop) përfundon ose kompletohet pa e gjetur vlerën e kërkuar.

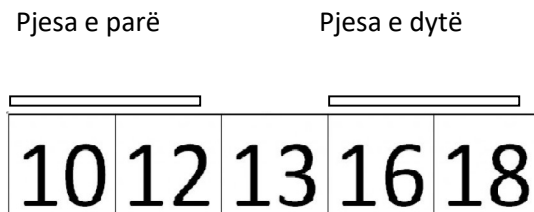


Fig. 49. Kërkimi binar

Kodi në C++ për kërkim binar:

```
#include<iostream>
using namespace std;
int main()
{
    int n,mesi,fillimi,fundi,vlera;
    cout<<"shkruaj gjatesine e vargut:";
    cin>>n;
    int vargu[n];
    cout<<"vargu i gjeneruar eshte \n";
    for(int i=0;i<n;i++)
    {
```

```

        vargu[i]=3*i+4;
        cout<<"vargu ["<<i<<"]= "<<vargu[i]<<endl;
    }

    cout<<"Shkruaj vleren e cila kerkohet: ";
    cin>>vlera;

    fillimi=1;
    fundi=n;
    mesi=(fillimi+fundi)/2;
    while(fillimi<=fundi && vargu[mesi]!=vlera)
    {
        if(vargu[mesi]<vlera)
            fillimi=mesi+1;
        else
            fundi=mesi-1;

        mesi=(fillimi+fundi)/2;
    }

    if(vargu[mesi]==vlera)
    {
        cout<<"Vlera e kerkuar gjendet ne indeksin numer:"<<mesi<<endl;
    }
    else
    {
        cout<<"Vlera e kerkuar "<<vlera<<" nuk gjendet ne varg ";
    }
}

```

Nëse analizojmë hapat që duhet të egzekutojë algoritmi, varësisht nga gjatësia e vargut të dhënë. Atëherë mund të kemi:

- Rasti më i shpejtë (Elementi i kërkuar ndodhet në indeksin $(\text{fillimi} + \text{fundi})/2$): urdhëri për kushtëzim egzekutohet vetëm 1 herë. kompleksiteti kohor $O(1)$.
- Rasti më i ngadalshëm (elementi i kërkuar ndodhet në indeksin e fundit); urdhëri për kushtëzim egzekutohet n herë. Kompleksiteti kohor $O(\log_2(n))$.

Nga vijnë në përfundim se kompleksiteti kohor të algoritmi për kërkim binar është $O(\log(n))$.

6.4. Sortimi (rradhitja) e elementeve të vargut

Shpesh është e nevojshme për ti rregulluar elementet në një varg në mënyrë numerike nga vlera më e lartë deri te vlera më e ulët (të rendit zbrites) ose anasjelltas.

Nëse vargu përmban vlera të tipit string (varg karakteresh), atëherë ndoshta do të jetë e nevojshme që të renditen sipas alfabetit (e cila bëhet duke përdorur vlerat ASCII).

Egzistojnë disa mënyra të rradhitjes së elementeve në varg. Mënyra e dhënë njihet si bubble sort(rradhitja fluska) ;

Mënyra e punës së bubble sort është krahasimi i vlerave fqinje dhe nëse janë në rradhitje të kundërt atëherë elementet ndërrojnë vendet me njëri tjetrin. Lista kalohet derisa nuk ka nevojë që elementet të ndërrojnë vendet me njëri tjetrin.

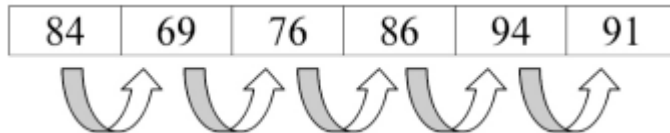


Fig 9. Rradhitja e elementeve

```
#include<iostream>
using namespace std;
inicializoVargun(int vargu[],int n)
{
    for (int i=0;i<n;i++)
    {
        cout<<"Vargu ["<<i<<"] = ";
        cin>>vargu[i];
    }
}
shtypVargun(int vargu[],int n)
{
    for(int i=0;i<n;i++)
        cout<<vargu[i]<<",";
        cout<<endl;
}
sortoVargun(int v[],int n) {
    int nd,i,j;

    for (i=0; i<n-1; i++) {
        for (j=0; j<n-i-1; j++)
            if (v[j] > v[j+1]) {
                nd = v[j];
                v[j] = v[j+1];
                v[j+1] = nd;
            }
    }
}

int main()
{
    int n;
    cout<<"Shkruaj gjatesine e vargut: ";
```

```

    cin>>n;
    int  vargu[n];
    inicializoVargun(vargu,n);
    sortoVargun(vargu,n);
    cout<<"vargu i sortuar"<<endl;
    shtypVargun(vargu,n);
}

```

Nëse vargu i dhënë është përmban vlerat (9,7,8,3,6)

Atëherë radhitja e elementeve do të bëhej si:

7,9,8,3,6

7,8,9,3,6

7,8,3,9,6

7,8,3,6,9

7,3,8,6,9

7,3,6,8,9

3,7,6,8,9

3,6,7,8,9

Nëse analizojmë hapat që duhet të egzekutojë algoritmi, varësisht nga gjatësia e vargut të dhënë. Atëherë mund të kemi:

Nëse gjatësia e vargut është n:

Urdhërat për krahasim dhe shkëmbim për ciklin i jashtëm egzekutohet n-1 here,

Urdhërat për krahasim dhe shkëmbim për ciklin i brendshëm egzekutohen

➤ (n-1) herë

➤ (n-2) herë

.....

➤ 1 herë

Kompleksiteti $O(n^2)$

Grafiku i cili paraqet kompleksitetin kohor të algoritmeve për kërkim linear, binar dhe rradhitje

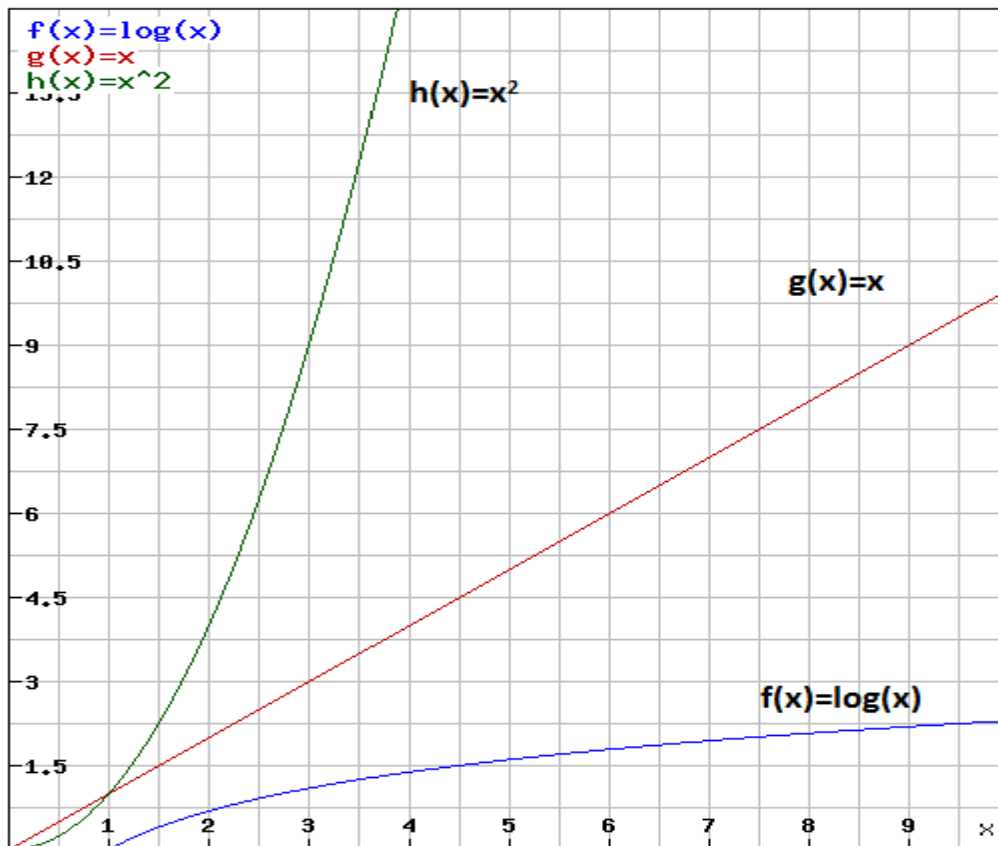


Fig 10. Grafiku për funksionin logaritmik, linear dhe katror

6.5. Krijimi i vargut me objekte

C++ mundëson krijimin e vargut ku si element përbërës janë instanca nga klasa e definuar për objektin.

Shembull: Të shkruhet programi i cili mundëson krijimin e instancave për objektin Nxënës, me atributet nrDitarit, emri, mbiemri, mungesat, nota mesatare.

```
#include<iostream>
using namespace std;
class Nxenesi
{
    short nrDit;
    string emri;
    string mbiemri;
    float notamesatare;
    short mungesat;
public:
    Nxenesi(){}
    Nxenesi(short nr, string em, string mb, float nm, short ms)
```

```

    {
        nrDit=nr;
        emri=em;
        mbiemri=mb;
        notamesatare=nm;
        mungesat=ms;
    }

    void set_emri(string em)
    {
        emri=em;
    }
    string get_emri()
    {
        return emri;
    }

    void set_mbiemri(string mb)
    {
        mbiemri=mb;
    }
    string get_mbiemri()
    {
        return mbiemri;
    }
    void set_notaMesatare(float nm)
    {
        notamesatare=nm;
    }
    float get_notaMesatare()
    {
        return notamesatare;
    }
    void set_mungesat(float nm)
    {
        mungesat=nm;
    }
    short get_mungesat()
    {
        return mungesat;
    }
    void paraqit()
    {
        cout<<" numri i ditarit: "<<nrDit<<endl;
        cout<<" emri: " <<emri<<endl;
        cout<<" mbiemri: " <<mbiemri<<endl;
        cout<<" mungesat: " <<mungesat<<endl;
        cout<<" mesatarja: " <<notamesatare<<endl;
    }
};

int main()
{
    Nxenesi klasa[34]; //krijohet vargu klasa e cili permban 34 objekte te
    tipit nxenes
    int n;
    cout<<"Sa nxenes do te fusni ne sistem ";

```

```

cin>>n;
short nrDit, mungesat;
string emri,mbiemri;
float notaMes;

for (int i=0;i<n;i++)
{
    cout<<"shkruaj te dhenat per nxenesin "<<i+1<<"\n";
    cout<<"numri i ditarit: ";
    cin>>nrDit;
    cout<<"emri: ";
    cin>>emri;
    cout<<"mbiemri: ";
    cin>>mbiemri;
    cout<<"mungesat: ";
    cin>>mungesat;
    cout<<"nota mesatare: ";
    cin>>notaMes;
    klasa[i]=Nxenesi(nrDit, emri, mbiemri,notaMes,mungesat);
}
cout<<"keni futur te dhenat per nxenesit"<<endl;
for(int i=0;i<n;i++)
{
    cout<<i+1<<" ".<<endl;
    klasa[i].paraqit();
}
}

```

6.6. Hyrje në C++ shabllonin vektor të librarisë standarde

Shablloni vector nga libreria standard, paraqet mënyrë më të përshtatshme dhe fuqishme duke poseduar disa mundësi shtesë për dallim nga vargjet.

Shembulli i dhënë demonstroi praktikimin e vektorëve:

Deklarimi i vektorit bëhet me sintaksën:

```
vector <tipi> emri_i_ndryshores (numri_i_elementeve);
```

Numri i elementeve është opsional, vektori mund të deklarohet edhe si:

```
vector <tipi> vemri_i_ndryshores;
```

Argumenti tipi në kllapat <> tregon tipin e elementeve të vektorit Urdhëri i dhënë do të deklarojë vektorë të zbrazët (një vector i cili përmban zero elemente) emri_i_ndryshores është emri i cili i jepet vektorit, dhe numri opsional numri_i_elementeve tregon numrin e elementeve që mund të përmbajë vektori

Shembujt në vijim ilustrojnë deklarimin e vektorëve:

```
vector<int> vlerat (5); // Deklarohet vektor me 5 numra të plotë
```

```
vector<double> notat (20); // Deklarohet vektor me 20 vlera numër
dhjetor(double)
vector<string> emra; // Deklarohet vektor me elemente varg tekstual
//përmban 0 vargje teksuale)
```

Për shfrytëzimin e vektorëve në program, duhet patjetër të të përfshijmë direktivën `#include` në fillim të programit, pasi që vektorët janë një vegël nga libraria standard, dhe jo pjesë thelbësore e gjuhës C++:

```
#include <vector>
```

Pas deklarimit të vektorit dhe specifikim i një numri të elementeve, mund ti referohemi elementeve veç e veç në vector duke përdorur kllapat katrore `[]`:

```
notat[3], emra[6]
```

Shembull i implementimit të vektorit me 5 anëtar dhe itearcini në vektor:

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

int main()
{
    vector<short> notat(5); //krijohet vektori notat me 5 elemente
    for (vector<short>::size_type i = 0; i < 5; i++)
    {
        cout << "Shkruaj noten per nxenesin #" << i+1<< ": ";
        cin >> notat[i];
    }
    short shuma=0;
    for (vector<short>::size_type i = 0; i < 5; i++)
    {
        shuma+=notat[i];
    }
    cout<<"nota mesatare "<<shuma/5.0;
    return 0;
}
```

Operacionet me vektorë

Vektorët mundësojnë operacione të ndryshme si, ndryshimi kapacitetit, krahasimi, shkëmbimi i vlerave, etj

Shembull i për mënyrat e krijimit të vektorëve.

```
#include <iostream>
#include <vector>
using namespace std;
int main ()
```

```

{
    unsigned int i;
    //krijimi i ndryshores vektoril
    vector<int> vektoril; // vektori i zbrazet
    cout<<"Elementet e vektoril:"<<endl;
    for(i=0;i<vektoril.size();i++)
        cout<<vektoril[i]<<" ";
    cout<<endl<<endl;
    //krijimi i ndryshores vektori2
    vector<int> vektori2(4,100); // 4 numra te plote me vlere 100
    cout<<"Elementet e vektori2:"<<endl;
    for(i=0;i<vektori2.size();i++)
        cout<<vektori2[i]<<" ";
    cout<<endl<<endl;
    //krijimi i ndryshores vektori3
    vector<int> vektori3(vektori2.begin(),vektori2.end()); // iteracioni ne
vektori2
    cout<<"Elementet e vektori3:"<<endl;
    for(i=0;i<vektori3.size();i++)
        cout<<vektori3[i]<<" ";
    cout<<endl<<endl;
    //krijimi i ndryshores vektori4
    vector<int> vektori4(vektori3); // kopje e vektori3
    cout<<"Elementet e vektori4:"<<endl;
    for(i=0;i<vektori4.size();i++)
        cout<<vektori4[i]<<" ";
    cout<<endl<<endl;
    int numrat[]={16,2,77,29};
    //krijimi i ndryshores vektori5
    vector<int> vektori5(numrat,numrat+sizeof(numrat)/sizeof(int));
    cout<<"Elementet e vektori5:"<<endl;
    for(i=0;i<vektori5.size();i++)
        cout<<vektori5[i]<<" ";

    return 0;
}

```

Shembull: Nga vektori i dhënë a të krijohen vektorët b dhe c ashtu që vektori b përmban elementet çift të vektorit a, ndërsa vektori c përmban elementet tek të vektorit a.

```

#include <iostream>
#include <vector>
using namespace std;
int main()
{
    int i,n,nr;
    vector<int> a, b, c;
    cout << "Shkruani numrin e elementeve te vektorit"<<endl;
    cout<<"n=";cin>>n;
    cout<<"Shkruani elementet e vektorit:"<<endl;
    for(i=0;i<n;i++)
    {
        cout<<"a["<<i<<"]=";
        cin>>nr;
    }
}

```

```
        a.push_back(nr);
    }
    for (int i=0; i<n; i++)
    {
        if (a[i]%2==0)
            b.push_back(a[i]); // nese numri eshte cift vendose ne vektorin b
        else
            c.push_back(a[i]); //nese numri eshte tek vendose ne vektorin c
    }

    cout<<"Vektori me numra cift:"<<endl;
    for (i=0;i<b.size();i++)
        cout<<b[i]<<"\t";
    cout<<endl;
    cout<<"Vektori me numra tek:"<<endl;
    for (i=0;i<c.size();i++)
        cout<<c[i]<<"\t";
    cout<<endl;
    return 0;
}
```

Detyra

1. Të shkruhet programi i cili gjen vlerën mesatare të elementeve të vektorit
2. Të shkruhet programi i cili gjen maksimumin dhe minimumin e elementeve nga vektori me elemente numra të plotë.
3. Të shkruhet programi i cili largon nga vektori elementet që përsëriten.

6.7. Vargu tekstual

Në çdo gjuhë programore vargu me karaktere (string) është tip i rëndësishëm i të dhënave pasi që paraqiten shpesh në aplikacione. Shembull, vargu “Programimi në C++” është varg me 17 shenja duke përfshirë 13 shkronja, 2 shenja aritmetike, dhe dy hapsira.

C++ standard mbështet dy lloje të ndryshoreve varg karakteresh (string) `std::string` (pjesë nga biblioteka standarde) dhe string i stilit- C.

Në definimin e tipit të të dhënave kemi dy karakteristika: domeni dhe bashkësia e opearcioneve. Pjesa më e rëndësishme është gjetja e bashkësisë së oeracioneve përkatëse.

String i stilit C është një varg i cili përbëhet nga karakterët i cili përdorë një **përfundues null**. Përfunduesi null është karakter special (`'\0'`, kodi ASCII 0) i cili përdoret për të treguar fundin e vargut. Në përgjithësi, stringu i stilit-C quhet edhe si string i **null- përfunduar**.

'C'	'+'	'+'	' '	'l'	'a'	'n'	'g'	'u'	'a'	'g'	'e'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

0x2104ADC0

Index	0	1	2	3	4	5
Variable	H	e	l	l	o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

Deklarimi i ndrshores varg karakteresh me gjatësi 50:

```
char vargu[50];
```

Urdhëri i dhënë deklaron string me gjatësi prej 50 karakterësh. Karakteri i parë fillon në indeksin 0. Përveç kësaj mbaron me null karakterin `'\0'`. Duhet të theksohet se kemi një shenjë shtesë në fund të vargut.

Gjithashtu stringu mund te inicializohet gjatë deklarimit:

```
char mystring[] = "string";
```

Urdhëri i mësipërm deklaron varg karakterësh me gjatësi 7; i cili përmban 6 karakteret s,t,r,i,n,g dhe null-karakterin për përfundim '\0'

```
#include <iostream>
Using namespace std;
int main()
{
    char mystring[] = "string";
    cout << mystring << " ka gjatesi " << sizeof(mystring) << "
karaktere.\n";
    for (int index = 0; index < sizeof(mystring); ++index)
        cout << static_cast<int>(mystring[index]) << " ";
    return 0;
}
```

Rezultati i egzekutimit të programit është:

```
string ka gjatei 7 karaktere.
115 116 114 105 110 103 0
```

Inicializimi i stringut në stilin-C ndjek rregullat e njehta sikurse vargjet. Kjo do të thotë mund të inicializosh stringun gjatë krijimit, por nuk mund të inicializohet pas krijimit të tij duke përdorur operaotrin për dhënien e vlerës (=).

```
char str[] = "string"; // lejohet
str= "teksti"; // nuk lejohet!
```

Ndryshimi i vlerave të vargut tekstual bëhet njësoj sikurse te vargjet numerike; përmes emrit të vargut dhe indeksit (pozitës) se karakterit.

```
char str[] = "string";
str[1]='p';
atëherë vargu str merr vleren spring
```

Inicializimi i vargut tekstual mund të behet përmes gets (nga biblioteka cstdlib), ndërsa vlerat e ndryshoreve mund të paraqiten në ekran përmes urdhërave cout dhe puts; Funkzioni **sizeof(vargu)** kthen vlerë numerike numër i plotë që është numri i karaktereve të vargut:

Shembulli 1: Te inicializohet dhe paraqitet në ekran ndryshorja e tipit varg karakteresh, të paraqitet në ekran edhe gjatësia e tekstit të shkruar.

```
#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
int main()
```

```

{
    char vargu[50]; //deklarimi i vargut tekstal me gjatesi 50
    karakter
    cout<<"Shkruaj tekstin e deshruar:";
    gets(vargu); //inicializimi i vargut

    cout<<"Keni shkruar tekstin \n";
    cout<<vargu;
    cout<<"\nGjatesia e teksit te shkruar eshte: "<<strlen(vargu)<<endl;
    cout<<"\nPo e paraqesim edhe nje here tekstin e shkruar\n";
    puts(vargu);
    return 0;
}

```

Vargu me karaktere mundëson kalimin e elementeve një nga një përmes urdhërave për përsëritje.

Shembull:

```

char vargu[]="Programim ne C++";
for(int i=0; i<sizeof(vargu);i++)
{
    cout<<vargu[i]<<endl;
}

```

Kodi i dhënë vargun “Programimi ne C++” e paraqet në ekran vertikalisht. Gjithashtu e njejta paraqet vargun vertikalisht duke lparaqitur edhe pozitën e çdo shkronje të vargut.

```

int j=0;
while(vargu[j]!='\0')
{
    cout<<j<<"    "<<vargu[j]<<endl;
    j++;
}

```

Shembulli 2: Teksti I përmes tastieres të paraqitet nga ana e kundërt.

```

#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;
int main()
{
    char tekst[100];
    int i,gjatesia;
    cout << "Shkruani tekstin: ";
    gets(tekst);
    cout<<"Keni shkruar tekstin: "<<endl;
    puts(tekst);
}

```

```

gjatesia=strlen(tekst);
cout<<"Teksti i shkruar nga ana kundert paraqitet si:"<<endl;
for(i=gjatesia-1;i>=0;i--)
    cout<<tekst[i];
return 0;
}

```

Funksionet të cilat mundësojnë manipulimin me vargun e karaktereve janë të definuara në bibliotekën cstring dhe përfshihen në program përmes urdhërit paraprocesorik

```
#include<cstring>
```

Disa nga funksionet janë

strlen(name) – kthen gjatësinë e vargut

strcat() –l shton një varg vargut tjetër

strncat() –l shton një varg vargut tjetër (duke llogaritur gjatësinë e vargut)

strcmp() –krahason dy vargje tekstuale (kthen vlerë 0 nëse janë të barabartë)

strncmp() – krahason dy vargje tekstuale deri te karakteri l specifikuar (kthen vlerë 0 nëse janë të barabartë)

Shembulli 3. Përdorimi l funksioneve nga biblioteka cstring:

```

#include<iostream>
#include<cstring>
using namespace std;
int main()
{
    char x[]="Hello";
    char y[]="Heddo";
    cout<<"Vargu " <<x<<"perbehet nga " <<strlen(x)<<"karakteret"<<endl;
    strcat(x,y);
    cout<<x;
    cout<<strcmp(x,y);
    if (strcmp(x,y))
        cout<<" te ndryshme";
    else
        cout<<"Te njejta ";
    return 0;
}

```

Shembulli 4: Të shkruhet programi i cili e gjen nëse vargu i dhënë është simetrik (lexohet njësoj nga ana e djathtë dhe e majtë)

```

#include<iostream>
#include<cstring>
#include<cstdio>
using namespace std;

```

```

int main(){
char teksti[100],teksti_mbrapsht[100];
bool simetrik;
int i,gjatesia;
cout << "Shkruani tekstin: ";
gets(teksti);

gjatesia=strlen(teksti);
cout<<"Keni shkruar tekstin:"<<endl;
for (int i=0;i<=gjatesia-1;i++)
    cout<<teksti[i];
cout<<endl;
//puts(teksti);

for(int i=gjatesia-1;i>=0;i--)                //gjen tekstin nga
mbrapa
    teksti_mbrapsht[i]=teksti[gjatesia-i-1];
cout<<"Teksti nga mbrapsht eshte:"<<endl;
for (int i=0;i<=gjatesia-1;i++)
    cout<<teksti_mbrapsht[i];
cout<<endl;
simetrik=true;
for(int i=0;i<=gjatesia-1;i++)//krahasohen te gjitha elmentet e vargut
{
    if (teksti[i]!=teksti_mbrapsht[i])
    {
        simetrik=false;
        break;
    }
}
if (simetrik==true)
    cout<<"Teksti i dhene eshte simetrik"<<endl;
else cout<<"Teksti i dhene nuk eshte simetrik"<<endl;
getchar();
return 0;
}

```

Detyra:

1. Të shkruhet programi i cili gjenë numrin e zanoreve në tekstin e dhënë përmes tastieres
2. Të gjendet fjala më e gjatë në vargun tekstual
3. Vargu i dhënë të çvendoset në mënyrë ciklike për k vende në anën e djathtë.

6.8. Klasa string

Versionet e para të C++ dhe gjuha paraardhëse C nuk kanë mbështetur shumë manipulimin me të dhënat string. Posedonin vetëm një numër të vogël të funksioneve në nivel të ulët. C++ standard posedon librarinë standard të tipit **string** e cila paraqet një qasje abstrakte ndaj string dhe një bashkësi të pasur me vegla për manipulim me string. Kjo mundëson manipulim me string përmes ndërmjetësimit (interfejsit) të definuar, pa shqetësimin se si ruhen në memorjen kompjuterike.

C++ shfrytëzon klasën standard string që të paraqesë dhe manipulojë me vargjet tekstual duke mundësuar manipulim të lehtë dhe të sigurtë. Gjatë manipulimit të operacioneve me string memorja e nevojshme automatikisht rezervohet dhe ndryshohet. Programuesi nuk ka nevojë të mendojë për rezervimin e memorjes.

Në klasën string janë të definuar disa operatorë të cilët mundësojnë kopjim ngritje dhe krahasim lehtësisht. Gjithashtu mundësojnë operacione shtesë si futja e tekstit në tekst, fshirja, kërkimi dhe zëvendësimi.

Në C++ ky tip i të dhënave është i definuar në interfejsin **<string>**, dhe duhet patjetër të të përfshihet në kodin burimor të fajllit i cili manipulon me të dhëna të tipit string.

Deklarimi dhe inicializimi i ndryshres string;

```
string ndryshorja1="Dardan Isaku";

string ndryshorja2("Vargu 2");
```

Operatori cin për inicializimin e vargut nuk mund të përdoret nëse vargu përmban hapsirë.

Shembull:

```
#include <string>
#include <iostream>
using namespace std;
int main()
{
    cout << "shkruani emrin e plote ";
    string emriPlote;
    cin >> emriPlote;
    cout << "Shkruani moshen: ";
    string mosha;
    cin >> mosha;
    cout << "Emri juaj eshte " << emriPlote << " dhe mosha juaj eshte " << mosha;
}
```

Për ta lexuar vargun e plotë në vijë përdoren urdhëri **getline(cin, ndryshorja);**

```
#include <string>
#include <iostream>
using namespace std;
int main()
{
    cout << "shkruani emrin e plote ";
    string emri;
    getline(cin, emri);
    cout << "Shkruani moshen: ";
    string mosha;
    getline(cin, mosha);
    cout << "Emri juaj eshte " << emri << " dhe mosha juaj eshte "
<<mosha;
}
```

Operatorët relacional

Operatorët relacional që përdoren për krahasimin e ndryshoreve string janë: =,!=, <,<=,>,>=

x==y	Vargjet x dhe y janë të barabarta
x!=y	Vargjet janë të ndryshme
x<y	Vargu është x është më i vogël se vargu y në radhitje alfabetike
x<=y	Vargu është x është më i vogël ose i barabartë me vargun y në radhitje alfabetike
x>y	Vargu është x është më i madh se vargu y në radhitje alfabetike
x>=y	Vargu është x është më i madh ose i barabartë me vargun y në rradhitje alfabetike

Shembull:

```
#include <string>
#include <iostream>
using namespace std;
int main()
{
    string st1 = "alfa";
    string st2 = "beta";
    if (st1==st2) cout << "st1 dhe st2 jane te barabarte\n";
    if (st1!=st2) cout << "st1 dhe st2 nuk jane te barabarte\n";
    if (st1< st2) cout << "st1 eshte me I vogel se st2\n";
    if (st1> st2) cout << "st1 eshte me I madh se st2\n";
    if (st1<=st2) cout << "st1 eshte me I vogel ose I barabrte me
st2\n";
    if (st1>=st2) cout << "st1 eshte me I madh ose I barabarte me
st2\n";
}
```

6.8.1. Operacionet me string

Operatori + mundëson bashkangjitjen e një vargu vargut tjetër:

```
string str="Programimi";

str=str+" ne C++.";

cout<<str;
```

vlera e ndryshore str është **Programim ne C++** .

Klasa string mundëson një numër të madh të operacioneve me ndryshore të tipit string. Në tabelë do të përshkruhen disa nga operacionet e klasës string.

length/size	Metodë e cila kthen numrin e karaktereve që i përmban vargu
str.at(pos) / str[pos]	Shprehjet kthejnë vlerën e karakterit në pozicionin pos në vargun str.
str.substr(pos, x)	Metoda kthen varg i cili përbëhet deri në x karaktere, duke filluar nga pozicioni pos të kopjuar nga vargu str. Parametri I dytë është opsional, nëse nuk shkruhet atëherë kthehet vargu nga pozicioni pos deri në fund të vargut. Vargu str mbetet I pandryshuar
str.find(x, pos)	Kërkon nënvargun x ne vargun str duke filluar nga pozicioni pos në vargun str.
str.insert(pos,txt)	Metodë e cila kopjon karakteret nga ndryshorja txt dhe I vendos në vargun str duke filluar nga pozicioni pos

```
#include <string>

#include <iostream>

using namespace std;

int main()

{

    string vargul="Programim",vargu2("ne C++.");

    cout<<"gjatesia e vargut "<<vargul<<" eshte "<<vargul.length()<<endl;

    cout<<"gjatesia e vargut "<<vargu2<<" eshte "<<vargu2.size()<<endl;

    int pos=3;
```



```
cout<<"Ne vargun "<<vargul<<" ne pozicionin "<<pos<<" gjendet
karakteri: "<<vargul.at(pos)<<endl;

cout<<"Ne vargun "<<vargul<<" ne pozicionin "<<pos<<" gjendet
karakteri: "<<vargu2[pos]<<endl;

cout<<"Nenvarug i vargul nga pozicioni 3 deri ne 5
"<<vargul.substr(3,4)<<endl;

cout<<"Nenvarug i vargu2 nga pozicioni 3 deri ne 5
"<<vargu2.substr(3,3)<<endl;

cout<<"Paraqitja e pare e karakterit m ne vargun "<<vargul<<" eshte
ne poziten "<<vargul.find("m",0)<<endl;

cout<<"Paraqitja e pare e karakterit m ne vargun"<<vargu2<<"eshte ne
poziten "<<vargu2.find("m",0)<<endl;

vargul.insert(0,"Mesojme ");

cout<<vargul<<" "<<vargu2;

}
```

7. MATRICAT

7.1. Deklarimi dhe manipulimi me matrica

Për paraqitjen sa më të qartë të të dhënave të cilat varen prej dy parametrave, përdoren fushat dydimensionale gjegjësisht matricat. Këtu për përcaktimin e pozicioneve të numrave përdoren dy madhësi. Në matematikë, matricat paraqiten si fusha në të cilat vizatohen rreshtat dhe kolonat.

Matrica është fushë dydimensionale e cila përbëhet nga m rreshta dhe n kolona ($m \times n$). m dhe n njihen si dimensionet e matricës. Në C++ numërimi i rreshtave dhe kolonave fillon me 0.

Çdo element i matricës është i përcaktuar me indeksin e tij i cili përbëhet nga numri i rreshti dhe kolonës.

	Column 0	Column 1	Column 2	Column 3	
Row 0	$a[0][0]$	$a[0][1]$	$a[0][2]$	$a[0][3]$...
Row 1	$a[1][0]$	$a[1][1]$	$a[1][2]$	$a[1][3]$...

Row Index
Column Index

Deklarimi i matricës A me 3 rreshta dhe 3 kolona me elemente numër i plotë në C++ bëhet me urdhërin:

```
int A[3][3];
```

Në matricën A mund të ruhen $3 \times 3 = 9$ elemente (numra të plotë) në 3 rreshta dhe 3 kolona.

Çdo element përcaktohet nga emri i matricës rreshti dhe kolona.

Rreshti \ Kolona	1	2	3
1	$A[0][0]$	$A[0][1]$	$A[0][2]$
2	$A[1][0]$	$A[1][1]$	$A[1][2]$
3	$A[2][0]$	$A[2][1]$	$A[2][2]$

Një nga mënyrat e inicilizimit të matricës është:

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

Elementet e matricës A do të kenë vlerat:

Rreshti \ Kolona	1	2	3
1	4	3	9
2	5	6	1
3	2	7	2

Elementi A[0][0] ka vlerë 4

Elementi A[0][1] ka vlerë 3

Elementi A[0][2] ka vlerë 9

Elementi A[2][2] ka vlerë 2

Përvec inicializimit gjatë deklarimit matrica mund të edhe në mënyrën vijuese;

Ngjajshëm me shtypjen e vektorëve shtypen edhe matricat por këtu duhet pasur kujdes që të fitohet forma tabelare e matricës e cila përdoret në matematikë. Gjatë punës me matrica, për t'i shfrytëzuar anëtarët e vendosur në fushat e veçanta të tyre, përdoren indekset e rreshtave dhe kolonave përkatëse.

Shembull: Të shkruhet program inicializohet matrica A[3][3] ashtu që çdo elementin I matricës është shumë e indekseve përkatës. A[0][0]=0+0=0, A[1][1]=2, A[1][2]=3, Elementet e matricës A të paraqiten në ekran:

```
#include<iostream>
using namespace std;
int main()
{
    int A[3][3];
    for(int rreshti=0; rreshti<3; rreshti++)
        for(int kolona=0; kolona<3; kolona++)
            A[rreshti][kolona]=rreshti+kolona;
    for(int rreshti=0; rreshti<3; rreshti++)
    {
        for(int kolona=0; kolona<3; kolona++)
            cout<<A[rreshti][kolona]<<"\t";
        cout<<endl;
    }
}
```

Shembulli 2: Të shkruhet program i cili gjen mesataren e elementeve të matricës A[m][n], m=numri I rreshtave, n- numri I kolonave:

```

#include<iostream>
using namespace std;
int main()
{
    int m,n;
    cout<<"Shkruaj numrin e rreshtave per matiricen A:";
    cin>>m;
    cout<<"Shkruaj numrin e kolonave per matiricen A:";
    cin>>n;
    int A[m][n];
    cout<<"Shkruaj elementet per matricen A \n";
    for(int i=0;i<m;i++)
        for(int j=0; j<n;j++)
        {
            cout<<"A["<<i<<"] ["<<j<<"]=";
            cin>>A[i][j];
        }

    int shuma=0;
    for(int i=0;i<m;i++)
        for(int j=0; j<n;j++)
            shuma+=A[i][j];
    float mesatarja=shuma*1.0/(m*n);
    cout<<"Vlera mesatare e elementeve te matrices eshte "<<mesatarja;
}

```

Shembulli 3. Të shkruhet program I cili njehson shumën e matricave $A[m][n]$ dhe $B[m][n]$ në matricën $C[m][n]$ ashtuqë $C[i][j]=A[i][j]+B[i][j]$ për $i=0..n-1$ dhe $j=0..m-1$, elementet e matricës jipen me tastier.

```

#include<iostream>
using namespace std;
int main()
{
    int m,n;
    cout<<"Shkruaj numrin e rreshtave per matiricat:";
    cin>>m;
    cout<<"Shkruaj numrin e kolonave per matiricat:";
    cin>>n;
    int A[m][n],B[m][n], C[m][n];

    cout<<"Shkruaj elementet per matricen A \n";
    for(int i=0;i<m;i++)
        for(int j=0; j<n;j++)
        {
            cout<<"A["<<i<<"] ["<<j<<"]=";
            cin>>A[i][j];
        }

    cout<<"Shkruaj elementet per matricen B\n";

```

```

for(int i=0;i<m;i++)
    for(int j=0; j<n;j++)
    {
        cout<<"B["<<i<<"] ["<<j<<"]=";
        cin>>B[i][j];
    }

for(int i=0;i<m;i++)
    for(int j=0; j<n;j++)
        C[i][j]=A[i][j]+B[i][j];

cout<<"Shuma e matricave te dhena eshte \n";
for(int i=0;i<m;i++)
{
    for(int j=0; j<n;j++)
        cout<<C[i][j]<<"\t";
    cout<<endl;
}
}

```

Shembulli 4: Të shkruhet program i cili matricën katrore $A[m][m]$ elementeve mbi diagonalen kryesre i shton numrin 2, elementet nën diagonalen kryesore i dyfishon, ndërsa elementet e digonales nëse janë pozitiv i vendos vlerë 1 nëse janë negativ i vendos vlerë -1.

```

#include<iostream>
using namespace std;
int main()
{
    int m,n;
    cout<<"Shkruaj dimensionin per matiricen A:";
    cin>>m;

    int A[m][m];

    cout<<"Shkruaj elementet per matricen A \n";
    for(int i=0;i<m;i++)
        for(int j=0; j<m;j++)
        {
            cout<<"A["<<i<<"] ["<<j<<"]=";
            cin>>A[i][j];
        }

    cout<<"Matrica e dhene eshte \n";
    for(int i=0;i<m;i++)
    {
        for(int j=0; j<m;j++)
            cout<<A[i][j]<<"\t";
        cout<<endl;
    }
}

```

```

cout<<endl;
for(int i=0;i<m;i++)
  for(int j=0; j<m;j++)
    if(i<j)
      A[i][j]+=2;
    else
      if(i<j)
        A[i][j]*=2;
      else
      {
        if(A[i][j]>=0)
          A[i][j]=1;
        else
          A[i][j]=-1;
      }

    cout<<"Matrica e fituar eshte \n";
for(int i=0;i<m;i++)
{
  for(int j=0; j<m;j++)
    cout<<A[i][j]<<"\t";
  cout<<endl;
}
}

```

Detyra

1. Të shkruhet program i cili gjenë vlerën minimale dhe vlerën maksimale në matricën e dhënë.
2. Të gjenerohet matrica $A[m][n]$ ku $a[i][j]=\begin{cases} 3i-j, & i > j \\ i+2j, & i < j \\ i+j, & i = j \end{cases}$
3. Nëse është dhënë vektori $V[n]$ $n>10$, të krijohet matirca $A[k][m]$ $k,m>3$.
4. Nga matrica e dhënë $A[m][n]$ të krijohet matrica $B[m][n]$, vargjet nga e matricës A do të jenë kolonë për matricën B

LITERATURA

1. <https://www.learncpp.com/>
2. www.functionx.com
3. <http://cforbeginners.com/>
4. <https://www.tutorialcup.com>
5. <https://www.tutorialspoint.com/>
6. Pol Dejtrel, Harvi Dejtrel: **C++: SI PROGRAMOHET**
7. Bazat e programimit në C++, Agni Dika
8. Informatika për vitin e parë, Danijella Gjorgjeviq
9. Algoritmet dhe programimi, Teksti mësimor për vitin e tretë ose të katërt të gjimnazit, Ivana Ognjanoviq, Ramo Shendel