

Gjuhët programuese (Gjuha programuese C++)

Tabela e përmbajtjes

PROGRAMIMI DHE GJUHËT PROGRAMUESE.....	5
GJUHËT PROGRAMUESE	5
SISTEMET NUMERIKE.....	7
SISTEMI NUMERIK DECIMAL.....	7
SISTEMI NUMERIK BINAR	7
SISTEMI NUMERIK OKTAL	7
SISTEMI NUMERIK HEKSADECIMAL	7
SHNDËRRIMI DECIMAL-BINAR	8
SHNDËRRIMI BINAR-DECIMAL	8
ARITMETIKA BINARE - MBLEDHJA	9
ARITMETIKA BINARE – ZBRITJA METODA KLASIKE	9
ARITMETIKA BINARE – ZBRITJA METODA KOMPLEMENTI I DYTË.....	9
SHNDËRRIMI DECIMAL – OKTAL.....	10
SHNDËRRIMI OKTAL - DECIMAL	10
ARITMETIKA OKTALE – <i>MBLEDHJA</i>	10
SHNDËRRIMI DECIMAL -> HEKSADECIMAL	10
SHNDËRRIMI HEKSADECIMAL -> DECIMAL	11
MBLEDHJA E NUMRAVE HEKSADECIMAL.....	11
SHNDËRRIMET NGA SISTEMI NUMERIK BINAR NË OKTAL DHE HEKSADECIMAL ME METODËN E GRUPIMIT	11
PARAQITJA E NUMRAVE NË KOMPJUTER	12
- <i>Paraqitja e numrave të plotë pozitiv</i>	12
- <i>Paraqitja e numrave të plotë negativ</i>	13
C++ MJEDISI DHE ZHVILLIMI.....	15
ALGORITMET	15
AKTIVIZIMI I CODE::BLOCKS.....	17
<i>HAPËSIRA PUNUESE E PROGRAMIT CODE::BLOCKS</i>	19
SHKRUARJA E PROGRAMIT	21
PËR GJUHËN PROGRAMORE C++	23
ELEMENTET E GJUHËS PROGRAMORE C++	23
SHENJAT, EMRAT DHE FJALET E REZERVUARA	23
STRUKTURA E PËRGJITHËSHME E PROGRAMIT NË C++	24
TIPET E THJESHTA TË TË DHËNAVE NE C++	28
DEKLARIMI I VARIABLAVE	29
REALIZIMI I STRUKTURAVE LINEARE	30
URDHËRESAT PËR DALJE TË TË DHËNAVE	30
KARAKTERI PËR TABELIM HORIZONTAL	30
SHTYPJA ME PRECIZITET TË ÇAKTUAR	31
URDHËRESA PËR SHOQËRIM	31
URDHËRESA PËR HYRJE TË TË DHËNAVE	33
PROGRAMIMI I ORIENTUAR NE OBJEKTE (KLASAT)	34
NUMËRIMET.....	34
<i>PËRCAKTIMI I GRUPIT</i>	34
<i>SHFRYTËZIMI I GRUPIT</i>	35
<i>DEGËZIMI PËRMES VLERA VE TË NUMËRUARA</i>	36
<i>DEGËZIMI ME KOMANDËN IF</i>	36

<i>DEGËZIMI ME KOMANDËN SWITCH</i>	37
<i>DISA VARIABLA TË NUMËRUARA TË TIPIT TË NJËTË</i>	40
<i>PËRCAKTIMI DHE DEKLARIMI NJËKOHËSISHT</i>	41
<i>SHOQËRIMI DIREKT I VLERAVE</i>	43
KLASAT	44
FORMA E PËRGJITHSHME E KLASAVE	47
KONSTRUKTORËT	55
STRUKTURË ZGJEDHJE	60
<i>ZGJEDHJA NGA DY MUNDËSI</i>	60
<i>ZGJEDHJE NGA MË SHUMË MUNDËSI</i>	61
<i>OPERATORET E KRAHASIMIT</i>	62
REALIZIMI I STRUKTURAVE PËR ZGJEDHJE	62
<i>DEGËZIMET E SHUMËFISHTA - SWITCH()</i>	65
<i>KAPËRCIMI PA KUSHT GOTO</i>	67
STRUKTURË PËRSËRITJE	68
<i>PËRSËRITJE ME DALJE NË FILLIM TË CIKLIT</i>	69
<i>PËRSËRITJE ME DALJE NË FUND TË CIKLIT</i>	69
<i>PËRSËRITJE DUKE NUMËRUAR CIKLAT</i>	69
REALIZIMI I STRUKTURAVE CIKLIKE	70
<i>URDHËRESA WHILE</i>	70
<i>URDHËRESA DO WHILE</i>	71
<i>URDHËRESA FOR</i>	71
FUNKSIONET	73
FUNKSIONET VOID	75
REKURZIONI	82
FUNKSIONET MATEMATIKORE TË BIBLIOTEKËS “CMATH” NË GJUHËN PROGRAMORE C++	87
TIPET E STRUKTURUARA TË TË DHËNAVE	91
VARG SHENJASH – STRING.....	95
FUNKSIONET PËR PUNË ME STRINGJE	99

PROGRAMIMI DHE GJUHËT PROGRAMUESE

Kompjuteri nuk është i aftë për asnjë punë kreative. Për t'u realizuar ndonjë përpunim, është e nevojshme dhënia e urdhëresës me të cilën aktivizohet dhe ekzekutohet programi adekuat. Programi paraprakisht inçizohet në memorjen e njehsorit pas së cilës mund të ekzekutohet urdhër për urdhër. Nëse programi është i shkruar saktë atëherë kompjuteri do të kryen programin dhe do të jep rezultatet.

Pra me një fjalë mund të themi se puna e një kompjuteri zhvillohet nën kontrollë të programeve të vendosur në memorje të tij. Të gjitha programet janë të shkruara nga ana e njeriut, ku më të ndërlikuarit nga ana e ekipeve të eksperteve të kësaj lëmie.

Programimi është proces i shkruarjes së programit.

Persona të cilët shkruajnë program quhen programues.

Procesi i shkruarjes program në vehte përfshin disa faza dhe atë:

- **parashtrimi i detyrës,**
- **definimi i veprimeve për zgjedhjen e detyrës,**
- **shkruarja e programit,**
- **testimi i programit.**

❖ **faza 1: Parashtrimi i detyrës**

Gjatë parashtrimit të detyrës duhet saktë të definohen dhe të përcaktohen saktë kushtet nën të cilat do të zgjidhet detyra. Pra detyra duhet të kuptohet plotësisht dhe saktë. Kjo bëhet duke analizuar atë mirë dhe nëse ka nevojë të ndërhyhet në atë lëmi të cilës i përket ajo detyrë.

❖ **faza 2: Definimi për zgjedhjen e detyrës**

Pas analizës së detyrës nevojitet gjetja dhe definimi i veprimeve për zgjedhjen e saj.

Zakonisht veprimet rrjedhin nga vet analiza e detyrës dhe normalisht nga konsultimet e literaturës për atë lëmi në të cilën janë të përfshirë spjegime për zgjedhjen e detyrës. Duhet pasur kujdes për veprimet që definohen të jenë të aplikueshme në kompjuter. Çdo operacion duhet të jetë i njëkuptimtë dhe rënditja e operacioneve duhet të jetë saktë i dhënë. Normalisht e gjithë kjo duhet të ketë fillimin dhe fundin pra numër të caktuar të veprimeve. Veprimet kështu të definuara për kryerjen e ndonjë detyreje quhen algoritëm.

❖ **faza 3: Shkruarja e programit**

Shkruarja e programit nënkupton shkruarjen (të shprehurit) e këtyre veprimeve me elementet e një gjuhe programore.

❖ **faza 4: Testimi i programit**

Në këtë fazë bëhet testimi i plotë i programit duke marrë vlera të ndryshme por për të cilat i dimë rezultatet, për të vërtetuar saktësinë e programit.

GJUHËT PROGRAMUESE

Sot njerëzit mes veti meren vesh dhe komunikojnë me ndihmën e ndonjë gjuhe. Pra gjuha paraqet një mjet themelor për komunikim. Gjuhët mund të jenë:

-natyror dhe

-artificial

Për komunikim ndërmjet njerzve, njerëzit shfrytëzojnë gjuhët natyrore si :gj. shqipe, gj. angleze etj. Për komunikim e njerzve dhe makinave shfrytëzohen gjuhët artificial.

Lloji i veçantë i gjuhëve artificial janë gjuhët programore, të cilat janë të zhvilluar për komunikimin e njerzve dhe kompjuterëve. Me këto gjuhë përshkruhen veprimet (algoritmi) për zgjedhjen e ndonjë detyreje, shkruar në ndonjë formë tekstuale, quhet program.

Gjuhët programuese, sipas ndërlíkueshmërisë ndahen në :

- gjuhët makinerike,
- gjuhët simbolike dhe
- gjuhët të larta programuese.

Gjuhët makinerike janë gjuhë të kuptueshme vetëm për kompjuterët. Urdhëresat janë të shprehura me ndihmën e sistemit numerik binar (pra 0 dhe 1). Janë gjuhë të lidhura ngushtë me tipin e kompjuterit.

Zëvendësimi i këtyre kombinacioneve të zerove dhe njëshave me simbole, paraqet grupin e dytë të gjuhëve të ashtuquajtura gjuhë simbolike. Këtu tash kemi moskuptim nga ana e kompjuterit të programit dhe paraqitet nevoja e përkthimit të këtyre programeve në gjuhë makinerike të kuptueshme nga ana e kompjuterit. Këto shpesh njihen edhe si assembler ashtu quhen edhe përkthyesit e tyre.

Gjuhët programuese të nivelit të lartë janë gjuhët në të cilat mendohet kur flasim për programim. Janë zhvilluar nga fundi i viteve 50. Pyetja kryesore për një zhvillim në këtë kahje ka qene : Si të afrohen kompjuterët tek njerëzit të cilët nuk i njohin gjuhët makinerike ose simbolike:.

Qëllimi kryesor ka qenë që programet të mund të shkruhen në ndonjë gjuhë që do të jetë e afërt me ndonjë gjuhë natyrore që shfrytëzohet nga njerëzit.

Edhe pse këto gjuhë janë gjuhë artificial, ato kanë alfabetin të përbërë prej shkronjave (shkronjat e alfabetit të gjuhës angleze), numrave dhe shenjave special. Me kombinimin e këtyre shenjave formohen konstrukcionet elementar të quajtur fjalë, të cilët përbëjnë alfabetin e gjuhës. Me kombinimin e këtyre konstrukcioneve elementare në tërësi më të gjëra fitohen konstrukcionet të ashtuquajtur fjali ose urdhëresa. Sipas kësaj mund të themi se edhe gjuhët artificiale kanë gramatikën e vet të përbër nga dy degët kryesore : sintaksa- degë që përfshin rregullat për drejtshkrimin e konstrukcioneve si elementare ashtu edhe të përbëra, dhe semantika- degë që përfshin rregullat për kuptueshmërinë e këtyre konstrukcioneve.

Programet e shkruara në këto gjuhë patjetër duhet përkthyer për tu ekzekutuar nga ana e kompjuterit. Kjo bëhet me përkthyes. Çdo gjuhë programore i ka përkthyesit e vet të quajtur sipas emrit të gjuhës dhe duke e shtuar fjalën përkthyes.

Program i shkruajtur në këto gjuhë quhet program burimor (angl. source program), ndërsa program i përkthyer quhet program ekzekutues (angl. executive program). Përkthimi skematikisht mund të paraqitet kështu :



Nuk janë rrallë rastet kur kemi nevojë për interpretimin e programit pa përkthyes, kjo bëhet me ndihmën e interpreterëve. Këto nuk japin program të përkthyer por vetëm, pasi që bëjnë analiza të nevojshme, e interpretojnë rreshtin programor dhe kalojnë në rreshtin e ardhshëm programor e ashtu me radhë prej fillimi e deri në fund të programit.

Gjuhët programuese të nivelit të lartë poashtu mund të ndahen në grupe të ndryshme dhe atë sipas lëmisë të aplikimit të tyre dhe atë :

- të përgjithëm (Java, C++, PASCAL, FORTRAN, BASIC etj.)
- për intelegjencë artificial (PROLOG, LIPS etj.)
- programet sistemor (ADA, MODULA1 etj.)
- special (JCL, Apt etj.).

SISTEMET NUMERIKE

- Në matematikë sistemi numerik paraqitet si koleksion i numrave dhe operacioneve.
- Operacionet elementare janë: mbledhja (+), zbritja (-), shumëzimi (*) dhe pjesëtimi (/).
- Baza e sistemit numerik paraqitet nga numri i shifrave që e përbëjnë sistemin. Sistemet numerike janë: sistemi **binar**, **oktal**, **decimal** dhe **heksadecimal**.

SISTEMI NUMERIK DECIMAL

- Është sistem i cili përdoret në jetën e përditshme.
- Sistemi decimal ose dekad përbëhet nga dhjetë shifrat 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Baza e këtij sistemi është dhjetë.

Shembull: numri 327

$$\begin{aligned}
 & 3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 \\
 & 3 \cdot 100 + 2 \cdot 10 + 7 \cdot 1 \\
 & 300 + 20 + 7
 \end{aligned}$$

0 1 2 3 4 5 6 7 8 9 B A Z A 10

SISTEMI NUMERIK BINAR

- Është sistem i përbërë vetëm prej dy shifrave 1 dhe 0.
- Kanë bazë 2.
- Të gjitha pjesët e kompjuterit komunikojnë në mes vete me kodin e numrave binar.

0 dhe 1 B A Z A 2

SISTEMI NUMERIK OKTAL

- Është sistem që përbëhet nga tetë shifra.
- Ka bazë 8.
- Përdoret për të programuar disa tipe / lloje të kompjuterëve.

0 1 2 3 4 5 6 7 B A Z A 8

SISTEMI NUMERIK HEKSADECIMAL

- Është sistem që përbëhet nga gjashtëmbëdhjetë shifra.
- Ka bazë 16.
- Në vend të numrave 10 11 12 13 14 15 janë përdorur shkronjat A B C D E F

**0 1 2 3 4 5 6 7 8 9
A B C D E F B A Z A 16**

SHNDËRRIMI DECIMAL-BINAR

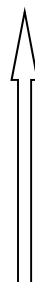
Numri :

$(101)_{10} = (1100101)_2$

 101 : 2 = 50 dhe mbetja 1
 50 : 2 = 25 dhe mbetja 0
 25 : 2 = 12 dhe mbetja 1
 12 : 2 = 6 dhe mbetja 0
 6 : 2 = 3 dhe mbetja 0
 3 : 2 = 1 dhe mbetja 1
 1 : 2 = 0 dhe mbetja 1

$(79)_2 = (1001111)_2$

 79 : 2 = 39 dhe mbetja 1
 39 : 2 = 19 dhe mbetja 1
 19 : 2 = 9 dhe mbetja 1
 9 : 2 = 4 dhe mbetja 1
 4 : 2 = 2 dhe mbetja 0
 2 : 2 = 1 dhe mbetja 0
 1 : 2 = 0 dhe mbetja 1

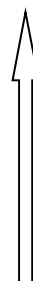


Numri me pjesë decimale :

$(125.875)_{10} = (1111101.111)_2$

 125 : 2 = 62 dhe mbetja 1
 62 : 2 = 31 dhe mbetja 0
 31 : 2 = 15 dhe mbetja 1
 15 : 2 = 7 dhe mbetja 1
 7 : 2 = 3 dhe mbetja 1
 3 : 2 = 1 dhe mbetja 1
 1 : 2 = 0 dhe mbetja 1

0.875 * 2 = 1.750 tepron 1
 0.750 * 2 = 1.50 tepron 1
 0.500 * 2 = 1.000 tepron 1

**SHNDËRRIMI BINAR-DECIMAL**

- Shndërrimi i numrit 10011 nga sistemi binar në decimal:

$$N = \sum_{i=1}^n x_i \cdot 2^{n-i}$$

1 0 0 1 1 ka 5 shifra, n=5

$$N = \sum_{i=1}^5 x_i \cdot 2^{5-i} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 0 + 0 + 2 + 1 = 19$$

- Shndërrimi i numrit 1110.11 nga sistemi binar në decimal:

$$N = \sum_{i=1}^n x_i \cdot 2^{n-i}$$

$$N = \sum_{i=1}^4 x_i \cdot 2^{4-i} + \sum_{j=1}^2 y_j \cdot 2^{-j} = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 8 + 4 + 2 + 0.5 + 0.25 = 14$$

ARITMETIKA BINARE - MBLEDHJA

Në sistemin dekad kemi shembull : 17

$+25$ kemi pra $7 + 5$ janë 12 minusojmë 10 dhe shkruajmë 2
42 dhe bartim një bazë në të majtë pra shkruajmë 1

dhe në sistemin binar kemi :

$1+0=1$

$1+1=0$ dhe bartet 1

$1+1=0$ bartet 1 në kolonën e ardhëshme

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1 \\ 10110101 \\ +\ 10001101 \\ \hline 101000010 \end{array}$$
ARITMETIKA BINARE – ZBRITJA METODA KLASIKE

Në sistemin dekad kemi shembull : 27

-15 kemi pra $5 - 7$, huazojmë nga ana e majte pra marim 1
9 dhe bëhet 17 ndërsa majtas mbetën 1 dhe kemi $17-5=9$

$0-0=0$

$0-1=1$ meret hua 1

$1-0=1$

$1-1=0$

$$\begin{array}{r} 2\ 2\ 1\ 1\ 1 \\ 0\ 0\ 0\ 2\ 2\ 2\ 2 \\ 1110000 \\ -\ 111011 \\ \hline 110101 \end{array}$$

-pra pasi që kemi $0-1$ marim hua nga majtas 1 pra në këtë rast kemi deri te njëshi i parë majtas i cili bëhet 0 pasi marim 1 dhe ashtu me radhe deri tek zeroja e parë djathtas.

ARITMETIKA BINARE – ZBRITJA METODA KOMPLEMENTI I DYTË

Numrat : 1110000

$-\ 111011$
110101

Atëher për metodën e komplementit të dytë kemi :

Komplementi i parë i zbritësit fitohet kur zëvendësojmë shifrat 0 me 1 dhe shifrat 1 me 0 të numrit pra :

Prej 111011 fitojmë 000100 që paraqet komplementin e 1-rë të numrit.

Komplementi i dytë fitohet kur komplementit të parë i shtojmë vlerën 1 pra :

000100

$+\ 1$

000101 që është komplementi i dytë i numrit

Tani pasi që fituam komplementin e dytë atë e mbledhim me numrin nga i cili duhet të zbritet pra 1110000 dhe kemi :

$$\begin{array}{r} 1110000 \\ +\ 000101 \\ \hline 1110101 \end{array}$$

dhe numrin e fundit majtas e fshijmë dhe fitojmë numrin : 110101

SHNDËRRIMI DECIMAL – OKTAL

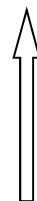
Numri:

$$(1981)_{10} = (3675)_8$$

$$(872)_{10} = (1550)_8$$

$$\begin{array}{l} 1981 : 8 = 247 \text{ dhe mbetja } 5 \\ 247 : 8 = 30 \text{ dhe mbetja } 7 \\ 30 : 8 = 3 \text{ dhe mbetja } 6 \\ 3 : 8 = 0 \text{ dhe mbetja } 3 \end{array}$$

$$\begin{array}{l} 872 : 8 = 109 \text{ dhe mbetja } 0 \\ 109 : 8 = 13 \text{ dhe mbetja } 5 \\ 13 : 8 = 1 \text{ dhe mbetja } 5 \\ 1 : 8 = 0 \text{ dhe mbetja } 1 \end{array}$$



SHNDËRRIMI OKTAL - DECIMAL

- Shndërrimi i numrit 534 nga sistemi oktal në decimal:

$$N = \sum_{i=1}^n x_i \cdot 8^{n-i}$$

5 3 4 ka 3 shifra, n=3

$$N = \sum_{i=1}^3 x_i \cdot 8^{3-i} = 5 \cdot 8^2 + 3 \cdot 8^1 + 4 \cdot 8^0 = 5 \cdot 64 + 3 \cdot 8 + 4 \cdot 1 = 320 + 24 + 4 = 348_{10}$$

ARITMETIKA OKTALE – MBLEDHJA

Në sistemin dekad kemi shembull :

$$\begin{array}{r} 17 \\ +25 \\ \hline 42 \end{array}$$

kemi pra 7 + 5 janë 12 minusojmë 10 dhe shkruajmë 2 dhe bartim një bazë në të majtë pra shkruajmë 1

Dhe në sistemin oktal kemi :

$$\begin{array}{r} 1 \quad 1 \\ 1737 \quad (991)_{10} \\ 423 \quad (275)_{10} \\ \hline 2362 \quad (1266)_{10} \end{array}$$

pra 7+3=10 minus baza 8 mbesin 2 që shkruhen dhe një bazë bartet majtas

SHNDËRRIMI DECIMAL -> HEKSADECIMAL

Numri:

$$(927143)_{10} = (E25A7)_{16}$$

$$(1872)_{10} = (750)_{16}$$

$$\begin{array}{l} 927143 : 16 = 57946 \text{ dhe mbetja } 7 \\ 57946 : 16 = 3621 \text{ dhe mbetja } (10) \text{ A} \\ 3621 : 16 = 226 \text{ dhe mbetja } 5 \\ 26 : 16 = 1 \text{ dhe mbetja } 10 \text{ A} \\ 1 : 16 = 0 \text{ dhe mbetja } (1) \text{ E} \end{array}$$

$$\begin{array}{l} 1872 : 16 = 117 \text{ dhe mbetja } 0 \\ 117 : 16 = 7 \text{ dhe mbetja } 5 \\ 7 : 16 = 0 \text{ dhe mbetja } 7 \end{array}$$



SHNDRRIMI HEKSADECIMAL -> DECIMAL

Shndërimi i numrit 7BD nga heksadecimal në decimal:

$$N = \sum_{i=1}^n x * 16^{n-i}$$

7 B D ka 3 shifra, n=3

$$N = \sum_{i=1}^3 x * 16^{3-i} = 7 * 16^2 + 11 * 16^1 + 13 * 16^0 = 7 * 256 + 11 * 16 + 13 * 1 = 1792 + 176 + 13 = 1981_{10}$$

MBLEDHJA E NUMRAVE HEKSADECIMAL

Në sistemin dekad kemi shembull :

$\begin{array}{r} +25 \\ 42 \end{array}$ kemi pra 7 + 5 janë 12 minusojmë 10 dhe shkruajmë 2
42 dhe bartim një bazë në të majtë pra shkruajmë 1

$$\begin{array}{r} 1 \quad 1 \\ 1DDC \\ + \quad 159 \\ \hline 1F35 \end{array}$$

Sepse :

$$\begin{array}{r} 1 \\ C \\ + 9 \\ \hline (21)_{10} = (15)_{16} \end{array} \quad \begin{array}{r} 1 \\ D \\ + 5 \\ \hline (19)_{10} = (13)_{16} \end{array} \quad \begin{array}{r} 1 \\ D \\ + 1 \\ \hline (15)_{10} = (F)_{16} \end{array}$$

Pra : 21-16 (baza) kemi 5 - shkruajmë, 19 -16 (baza) kemi 3 –shkruajmë dhe 15 pra F dhe 1 që do të thotë kemi : **1F35** ₍₁₆₎

SHNDËRRIMET NGA SISTEMI NUMERIK BINAR NË OKTAL DHE HEKSADECIMAL ME METODËN E GRUPIMIT

Për shembull numri :

110111101 do ta grupojmë sipas sistemit numerik në të cilin dëshirojmë të konvertojmë (shndërrojmë) pra në grupe nga 3 bit për në oktal dhe në grupe nga 4 bit për në heksadecimal .

Dhe kemi :

$$\begin{array}{ccc} 110 & 111 & 101 \\ \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} \\ 6 & 7 & 5 \end{array} \text{ tani nëse veçmas i njehsojmë kemi në grupe nga 3 bit :}$$

pra numrin në oktal e kemi **675₈**

$$\begin{array}{ccc} 0001 & 1011 & 1101 \\ \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} & \underbrace{\hspace{1cm}} \\ 1 & 11-B & 13-D \end{array} \text{ tani nëse veçmas i njehsojmë kemi në grupe nga 4 bit:}$$

pra numrin në heksadecimal e kemi **1BD₁₆**

PARAQITJA E NUMRAVE NË KOMPJUTER

- Paraqitja e numrave të plotë pozitiv

Numrat natyrorë dhe zeroja janë numra të plotë pozitiv. Ato në kompjuter paraqiten me 8, 16, 32 dhe më shumë bite, e pasiqë janë numra pozitiv shenja para tyre nuk shënohet.

Me n-bite mund të paraqiten 2^n numra: **prej 0 deri 2^n-1**

Pra:

- Me 8 bite mund të paraqiten $2^8 = 256$ numra dhe atë prej **0 deri $2^8-1=255$**

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
0 – numri më i vogël i plotë pozitiv							

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
0 – numri më i vogël i plotë pozitiv							

- Me 16 bite mund të paraqiten $2^{16} = 65536$ numra , pra numrat që mund të paraqiten : prej **0 deri $2^{16}-1=65535$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 – numri më i vogël pozitiv															

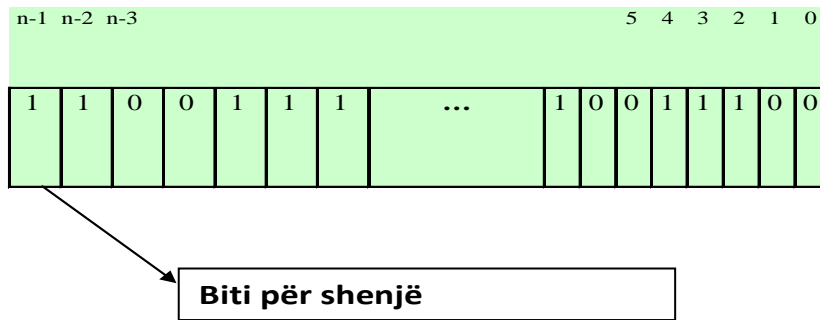
1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
65535 – numri më i madh i plotë pozitiv															

- **Paraqitja e numrave të plotë negativ**

- Për paraqitjen e numrave të plotë negativ përdoret i ashtuquajti sistemi 2 - komplementar, ku numri negativ N paraqitet si:

$$-N = 2^n - N$$

ku biti me rëndësi më të madhe quhet **bit për shenjë**

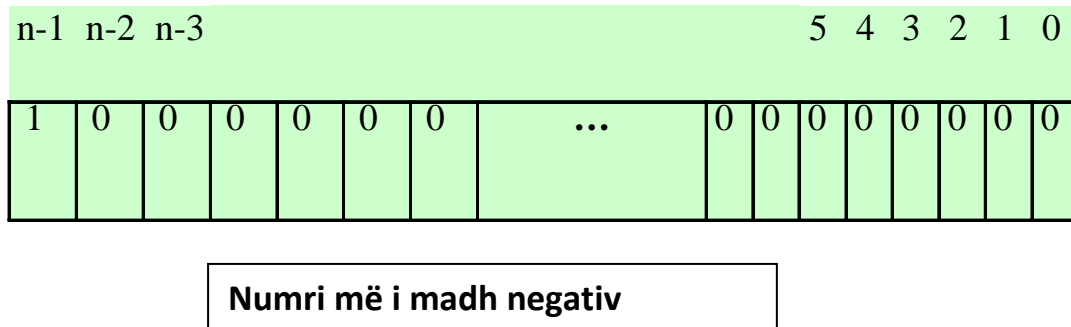


Pra nëse numri është pozitiv biti i parë shkruhet “0” ndërsa nëse numri është negativ atëhere shkruhet “1”

Pasiqë biti i n-të përdoret për shenjën e numrit, mbeten **(n-1)** – bite për paraqitjen e vlerës së numrit.

Numri më i madh që mund të paraqiten është kur të gjithë bitet janë njëshe, e ai është numri $2^{n-1}-1$.

P.sh për **n=8**, numri më i madh pozitiv është numri $2^{8-1}-1 = 127$, kurse për **n=16** është numri $2^{16-1}-1 = 32767$.



P.sh për **n=8**, numri më i madh negati është numri $-2^{8-1} = -128$, kurse për **n=16**, do të jetë numri $-2^{16-1} = -32768$

Intervali i numrave të plotë të paraqitura me n-bite mund të shënohet kështu

prej -2^{n-1} deri $2^{n-1}-1$

në të cilin ka : 2^{n-1} (numra negativ)

1 (numri zero)

$2^{n-1}-1$ (numra pozitiv) ose gjithsej :

2^n (numra)

Gjatë operacioneve aritmetike me numrat e kufizuar në ndonjë nga intervalet prej 2^{n-1} deri $2^{n-1}-1$ ose nga 0 deri 2^n-1 mund të vijë deri te gabimi i ashtuquajtur si stërbushje (overflow) kur rezultati është jashtë intervalit.

numri dekad	numri binar	numri dekad	numri binar
	16-bit		16-bit
0	00...00000	-16	11...10000
1	00...00001	-15	11...10001
2	00...00010	-14	11...10010
3	00...00011	-13	11...10011
4	00...00100	-12	11...10100
5	00...00101	-11	11...10101
6	00...00110	-10	11...10110
7	00...00111	-9	11...10111
8	00...01000	-8	11...11000
9	00...01001	-7	11...11001
10	00...01010	-6	11...11010
11	00...01011	-5	11...11011
12	00...01100	-4	11...11100
13	00...01101	-3	11...11101
14	00...01110	-2	11...11110
15	00...01111	-1	11...11111

C++ MJEDISI DHE ZHVILLIMI

ALGORITMET

Çka është algoritëm ?

Algoritëm është bashkësi e fundme e rregullave (veprimeve) për kryerjen e ndonjë detyreje me radhitje të definuar të kryerjes se tyre.

Në jetën e përditshme njerëzit kryejnë algoritme të ndryshme si p.sh. gatimi i ndonjë specialiteti etj.

Të marrim një shembull :

Prej tre numrave të dhënë të gjejmë se cili nga ato numra është më i madh.

Nëse marrim vlerat konkrete: 3, 7 dhe 4.

-nëse i krahasojmë 3 dhe 7, më i madh është 7; nëse i krahasojmë 7 dhe 4, më i madh është 7; pra numri më i madh është 7.

-nëse i krahasojmë 3 dhe 4, më i madh është 4; nëse i krahasojmë 4 dhe 7, më i madh është 7; pra numri më i madh është 7

-nëse i krahasojmë 4 dhe 7, më i madh është 7; nëse i krahasojmë 7 dhe 3, më i madh është 7; pra numri më i madh është 7.

Pra shohim se kemi vetëm dy veprime të definuar dhe atë: krahasimi i cilado dy numrave dhe gjetja e më të madhit, krahasimi i më të madhit nga krahasimi i parë me numrin e tretë dhe gjetja e numrit më të madh.

Për kryerjen e kësaj detyre duhet të kemi të definuar vlerat dhe si rezultat të veprimeve e kemi numrin më të madh. **Pra me algoritëm do të nënkuptojmë bashkësi e fundëshme veprimesh mbi të dhënat hyrëse sipas një radhitjeje paraprakisht të caktuar që sjell deri te rezultatet dalëse.**

Hapat algoritmik

Veprimet prej të cilëve përbëhet një algoritëm quhen hapa algoritmik. Mvarësisht nga ajo se a janë hapat të përgjithshme ose detale kemi algoritëm të përgjithshëm ose detal.

psh. algoritëm detal:

Paraqitja e algoritmeve

Algoritmet mund të paraqiten në dy mënyra :

- tekstualisht
- grafikisht

Paraqitja tekstuale nënkupton përshkrimin e algoritmit në formë tekstuale duke shfrytëzuar ndonjë pseudogjuhë shembull :

ALGORITËM MëiMadh

FILLIM

LEXO a,b,c;

NËSE a>b

ATËHERË

p<-a

PËRNDRYSHË

P<-b

FUND-NËSE (a>b)

NËSE p>c

ATËHER



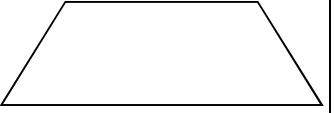

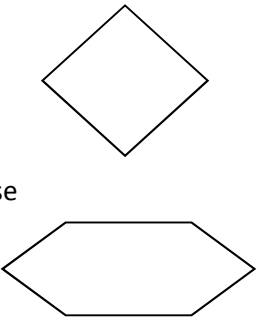

n<-p

PËRNDRYSHË

N<-C

FUND-NËSE (p>c)**PARAQIT N;****FUND (MËIMADH).**

Paraqitja grafike ndryshe quhet **BLOKDIJAGRAM (flowchart)** ku shfrytëzohen simbole grafike për çdo hap algoritmik.

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

AKTIVIZIMI I CODE::BLOCKS

Aktivizimi Code::Blocks bëhet duke klikuar Start → All Programs → CodeBlocks → CodeBlocks. Pas aktivizimit do të hapet dritarja përkatëse si në Fig.1

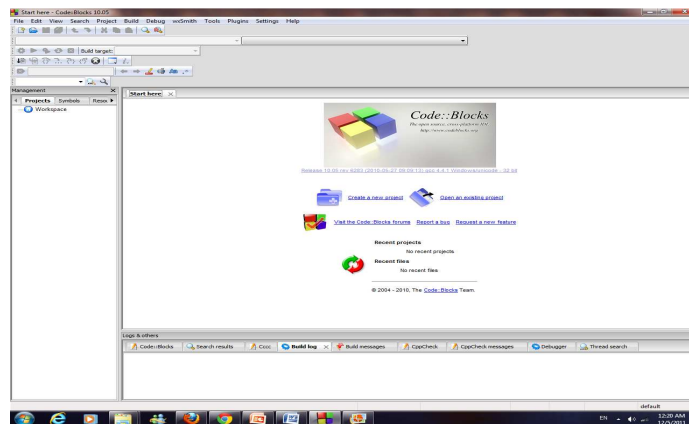


Fig. 1

Krijimi i një projekti të ri

Programimi në gjuhën C++ fillon me hapjen e një projekti të ri. Për këtë qëllim, në menynë kryesore zgjedhet opcioni File → New → Project, nga do tju hapet dritarja dialoguese si në Fig.2

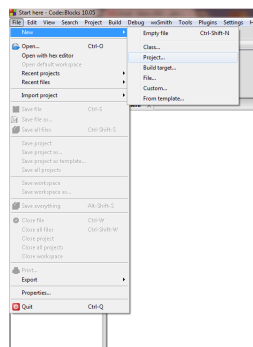


Fig.2

Në dritarën e mëposhtme (Fig.3) meqë duam të shkruajmë programe të zakonshme në C++, e zgjedhim opcionin **Console Application**, pasi të kemi zgjedhur vazhdojmë duke klikuar në **Go** dhe pastaj **Next** si në Fig. 4.

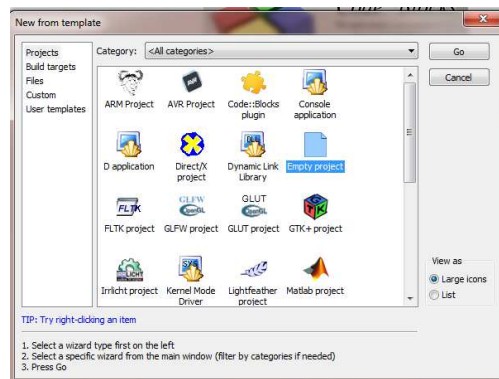


Fig.3



fig. 4

Në hapin tjetër duhet të përcaktohet emri i projektit, duke e shkruar atë në kornizën para së cilës është shënuar teksti **Project title:** . Njëkohësisht në kornizën **Folder to create project in:** zgjedhet edhe folderi i diskut në të cilin do të ruhet projekti. Këtu si shembull, proekti është quajtur me emrin program1 si në Fig. 5.

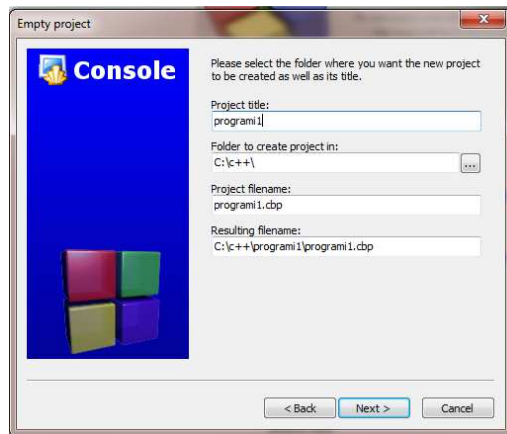


fig. 5

Pasi të zgjedhet emri i projektit dhe folder-i ku do të ruhet, procedura e hapjes së një projekti do të vazhdojë, nëse klikohet pulla Finish si në Fig. 6.

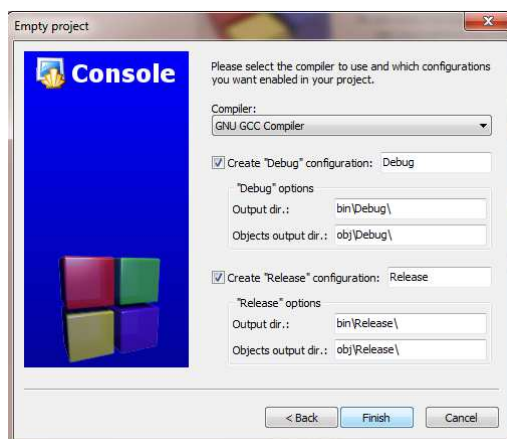


fig. 6

Si rezultat, aktivizohet dritarja e projektit shihet në Fig.7

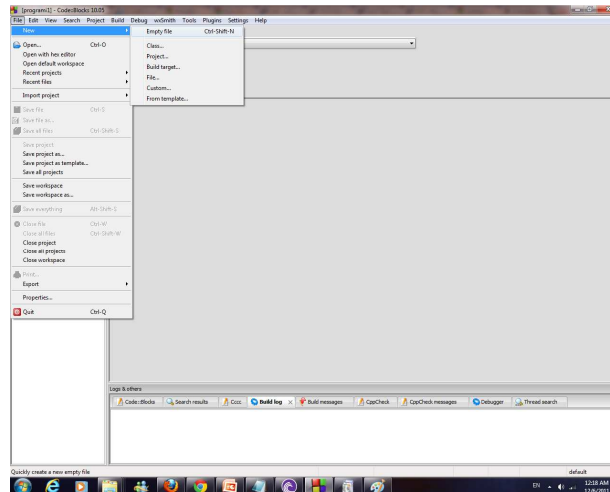


Fig. 7

Hapësira punuese e programit CODE::BLOCKS

Shkruarja e një program në kuadër të projektit fillon me hapjen e hapësirës punuese përkatëse. Për këtë qëllim shtohet fajll i ri me urdhëresën File → New pastaj zgjidhet opcioni Empty File si në Fig.8 dhe pastaj aktivizimi i hapësirës punuese si në Fig. 9

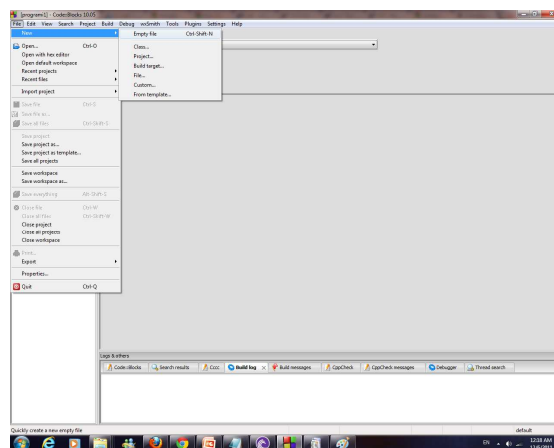


Fig. 8

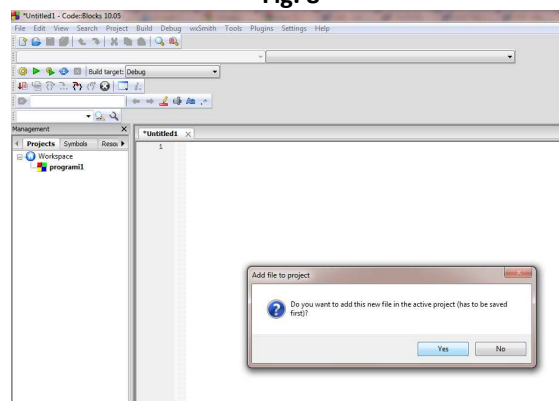


Fig. 9

Në figurën 9 me anë të dritares dialoguese përcaktohet lejimi i fajllit c++ në projekt dhe ruajtja e tij si në Fig.10.

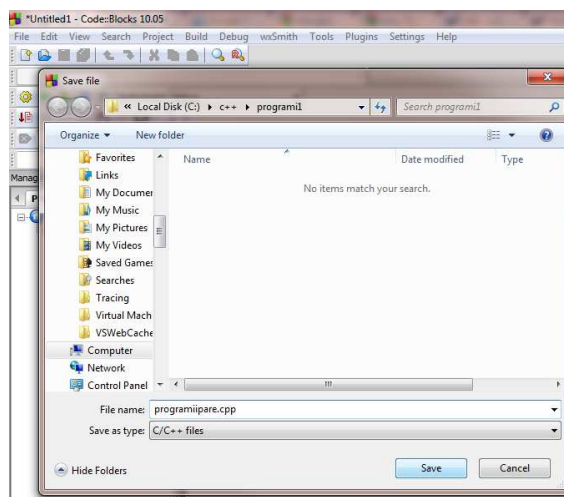


Fig. 10

Ruajtja e fajllit bëhet duke shënuar **emrinefajllit.cpp**. Në shembullin tonë emri është **programiipare.cpp**. Emri i fajllit shënohet me **.cpp** për shkak se programi që ne do të krijojmë është i gjuhës programore C++. Pasi kemi shtuar fajllin c++ në projekt na hapet dritarja dialoguese si në Fig. 11, ku për të vazhduar duhet të klikohet në ok .

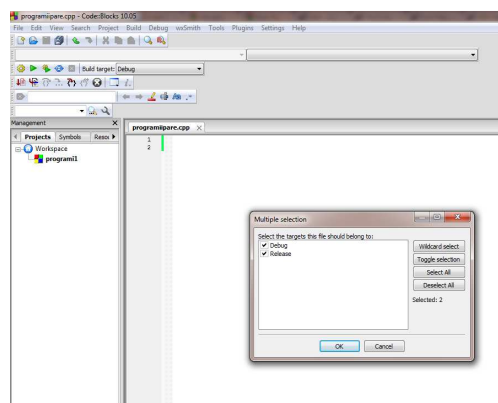


Fig. 11

Me veprimin e fundit përfundon krijimi i projektit të ri si dhe krijimi i hapësirës punuese në gjuhën programore C++ si në Fig. 12

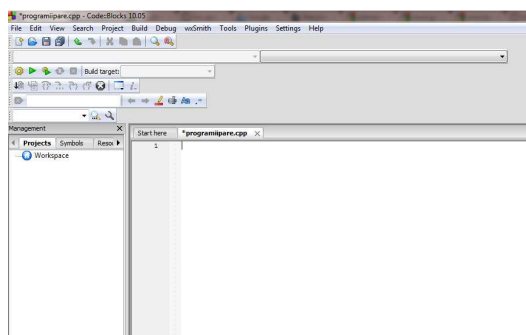


Fig. 12

SHKRUARJA E PROGRAMIT

Programi përmes së cilit shtypet teksti "Mire se erdhet ne programim ne gjuhen programore C++" si në Fig. 10

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Mire se erdhet ne programim ne gjuhen programore C++" << endl;
    return 0;
}
```

shënohet në vendin përkatës si në Fig.13

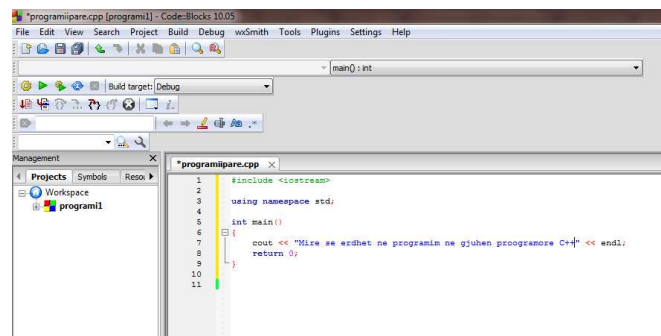


Fig. 13

Për ta kompajluar programin e shkruar, në menynë kryesore duhet të zgjidhet opcioni Build dhe pastaj nënopcionin Compile current file (shtypen tastet Ctrl-Shift-F9) ashtu siç shihet në Fig. 14

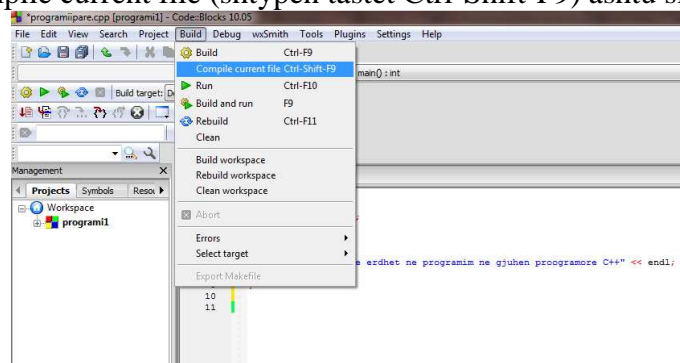


Fig. 14

Rezultati i kompajlimit shihet në dritaren Output si në Fig.15, e cila gjendet në fund të ekranit. Nëse kompajlimi është pa asnjë gabim, në këtë dritare do të paraqitet mesazhi:

*Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings*

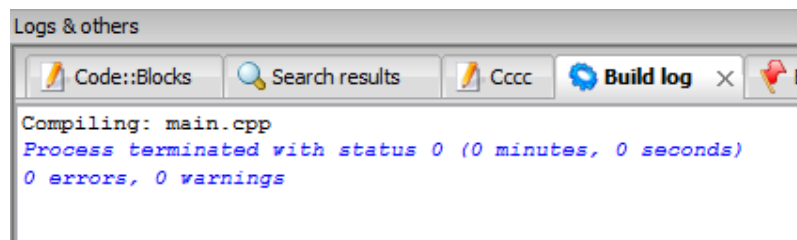


Fig.15

Ekzekutimi i programit bëhet duke e zgjedhur opcionin Build nga menyja kryesore dhe nënopcionin Run (Ctrl-F10) si në Fig.16

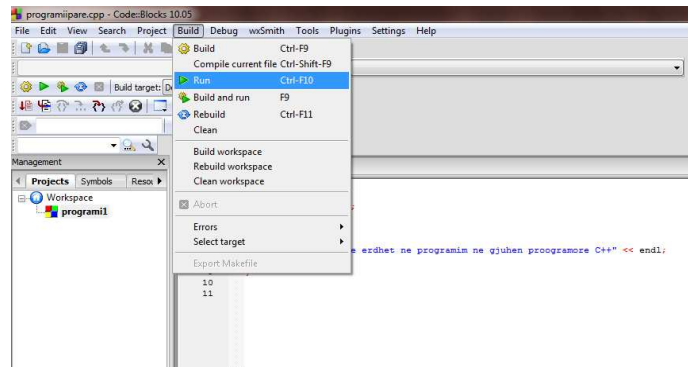


Fig.16

Nëse programi ekzekutohet për herë të parë do të na paraqitet dritarja dialoguese si më poshtë e cila na pyet se a duam të lejojm krijimin e projektit të ri nga ne, duhet të zgjedhim Yes si në fig.17

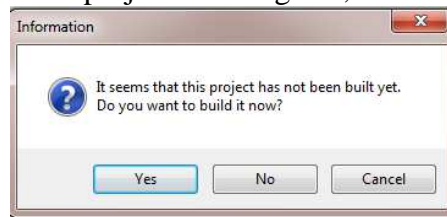


fig. 17

Nga pastaj do të shohim tabelën e rezultateve si në Fig.18

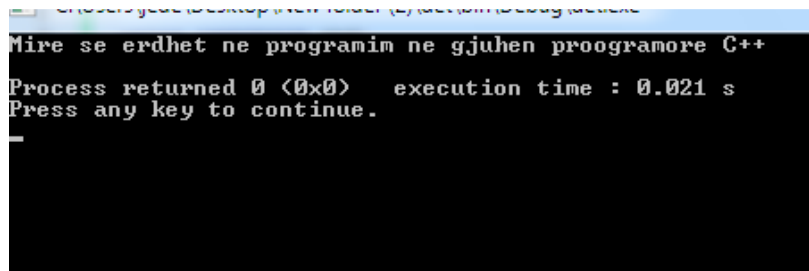


Fig.18

Nëse në kodin e programit të njëjtë bëjmë ndryshime atëher duhet të përsëriten veprimet : Build → Compile the current file dhe veprimi tjetër Build dhe Build and Run

menyra me e thjesht

file new --> console aplication -- >c++--> pastaj emri i projektit dhe finish

ne sources eshte fajlli i gatshem ne c++

PËR GJUHËN PROGRAMORE C++

Gjuha programore është shkruar në vitin 1983 nga Bjarne Stroustrup si verzion i gjuhës programore C, duke trashëguar efikasitetin e gjuhës C dhe duke e kombinuar me metodat objekt-orientuese të gjuhës Simula. Në gjuhën C++ i njëjti program mund të shkruhet në disa stile. Ta zëmë programi mund të shkruhet me metodën procedurale si në C, Fortran dhe Pascal, apo në metodën objekt-orientuese, metodë që në fillim është përdorë nga gjuhët si Simula dhe Smalltalk. Programimi objekt-orientues kërkon që programeri ta shohë problemin duke i menduar pjesët e programit si "objekte" që sajojnë problemin për sjelljet dhe tiparet e këtyre "objekteve" si dhe relacionet në mes tyre.

Gjuha programore C++ ka aplikim të gjërë. Egziston një listë e gjatë e sistemeve dhe aplikacioneve të cilat kryesisht janë të shkruara në gjuhën programore C++ si p.sh : Photoshop & ImageReady, Acrobat, InDesign, Mozilla, Maya 3D, disa aplikacione të Google, Winamp Media Player, Bloomberg, Windows 95, 98, Me, 200 dhe XP gjithashtu janë të shkruar në C++ etj.

ELEMENTET E GJUHËS PROGRAMORE C++

SHENJAT, EMRAT DHE FJALET E REZERVUARA

Gjuha programore C++ është e përbër nga një bashkësi shenjash (simbolesh) të cilat janë:

-germat e vogla dhe të mëdhaja të alfabetit anglez:(a,b,.....z,A,A,.....,Z),

-numrave :0,1,2,3,4,5,6,7,8,9

-shenjat special (+-*/=^< >:;()[]{}'#\$%&_)

me këto shenja shkruhen të gjitha emrat dhe urdhëresat në gjuhën programore C++

Formulimi i emrave në C++ ka disa rregulla. Emrat që i formulon programeri quhen identifikatorë. Janë të përbër prej shkronjave numrave dhe shenjës special _ (nënvizuar) ku si shenjë e parë nuk guxon të jetë shifër(numër). Shkronjat mund të jenë të vogla ose të mëdhaja. Në gjuhën C++ gjatësia e identifikatorëve mund të shkruhet nga 1 deri në 255 karaktere. Por përdorimi i identifikatorëve shumë të thejstë është jo praktik e ndonjëherë edhe me probleme gjatë kompajllimit.

Shembuji të emrave të rregullt:

dita	TemperaturaDitore
Koha5	_Fillimi
distance_mes_rreshtave	a2565b47

Shembuji te emrave jo të rregullt:

4muaji	Fillon me numer
nata+dita	E Permban simbolin +
ab Jeta	Permban shprazetire
U&&	E permban simbolin &
Libri#Rend	E permban simbolin #
%fitimi	E permban simbolin %

Në C++ egzistojë emra të rezervuar të cilët nuk mund të shfrytëzohen si **identifikatorë** sic janë:

asm	else	new	this
auto	enum	operator	throë
break	explicit	private	true
case	export	protected	try
catch	extern	public	typedef

char	false	register	typeid
class	float	reinterpret_cast	typename
const	for	return	true
const friend	const friend	short unsigned	short unsigned
const_cast goto	const_cast goto	signed using	signed using
continue if	continue if	sizeof virtual	sizeof virtual
default inline	default inline	static void	static void
delete	int	static_cast	volatile
do long struct echar_t	do long struct echar_t	do long struct echar_t	do long struct echar_t
double mutable	double mutable	sëitch ehile	sëitch ehile
dynamic_cast	namespace template	namespace template	

STRUKTURA E PËRGJITHËSHME E PROGRAMIT NË C++

Struktura e përgjithëshme e programit në C++ është

```
#include <iostream>           // #-simboli për direktivat paraprocesorike
                               /
                               /*(përfshijë, përmbajë) – C++ i ka libraritë e urdhërave, të
                               cilat i përfshinë në programe, përmes direktivës #include*/
                               /*Input Output Stream (Rrjedha hyrëse, dalëse) – përmbanë
                               urdhërat për hyrje/dalje (cout, cin)*/

using namespace std;
int main ()                   //funskioni kryesor
{                             //Fillimi i programit
return 0;
}                             //Fund i programit
```

Një nga rrugët më të mira për të mësuar një gjuhë programimi është duke filluar me një program. Ja edhe program i jonë i parë:

Pershendetje te gjithëve!

```
//program im i pare në C++
#include <iostream>
using namespace std;
int main()
{
cout<<"Persendetje te gjithëve !";
return 0;
}
```

Zona majtas paraqet kodin e programit tonë të parë, të cilin mund ta emërojmë, p.sh./pershendetje.cpp. Zona djathtas paraqet rezultatin e programit pasi ai kompilohet dhe ekzekutohet. Mënyra se si shkruhet dhe kompilohet një program varet nga kompilatori që do të përdoret.

Program i mësipërm ka si rezultat paraqitjen në ekran të mesazhit “Pershendetje te gjithëve!”. Ai është një nga programet më të thjeshtë që mund të shkruhen në C++, por përfshin komponentët bazë të cilët i ka çdo program në C++. Le ti shohim ato një nga një.

//program im i parë ne C++

Ky rresht është një koment. Të gjithë rreshtat që fillojnë me dy simbole slash (//) konsiderohen komente dhe nuk kanë asnjë efekt në program. Komentët mund të përdoren nga programuesit për të dhënë sqarime të shkurtra brenda kodit. Në rastin konkret, është dhënë një përshkrim i shkurtër i programit.

#include<iostream>

Rreshtat që fillojë me simbolin (#) janë direktiva të preprocesorit. Ato nuk janë instruksione të ekzekutueshme por të dhëna (informacione) për kompilatorin. Në këtë rast rreshti `#include <iostream>` i tregon preprocesorit të kompilatorit që të përfshijë edhe fajllin (skedarin) kokë standard `iostream`. Ky skedar përmban deklaratimet e librarisë standard të C++ për hyrje-daljet (input – output), dhe është përfshirë në programin tonë sepse funksionalitetet e tij përdoren më poshtë.

int main ()

Ky rresht i korrespondon fillimit të deklarimit të funksion `main`. Funksioni `main` është pika nga ku të gjithë programet C++ fillojnë ekzekutimin e tyre. Pavarësisht nga pozicioni që zë në kodin e programit, në fillim, në mes apo në fund, përmbajtja e tij ekzekutohet e para kur fillon ekzekutimi i programit. Prandaj është një kërkesë themelore që të gjithë programet në C++ të kenë një funksion `main`.

`main` ndiqet nga një çift kllapash () sepse është një funksion. Në C++ të gjithë funksionet ndiqen nga një çift kllapash () të cilët, mund të përmbajnë brenda tyre argument.

Përmbajtja e funksionit `main` ndjek menjëherë deklarimin formal dhe përfshihet ndërmjet kllapave të mëdha ({ }), si në shembullin tonë.

cout<<"Pershendetje te gjitheve !";

Ky instruksion kryen detyrën më të rëndësishme në këtë program. `cout` kanali standard i daljes (standard output stream) në C++ (që zakonisht është ekrani), dhe e gjithë fjalia fut një radhë karakteresh (mesazh në këtë rast "Pershendetje te gjitheve!") të jetë kanal (ekrani). `cout` është deklaruar në skedarin kokë `iostream`, prandaj për ta përdorur duhet që përfshijmë këtë skedar në program. (`include <iostream>`)

Duhet vënë në dukje se instruksioni përfundon me simbolin pikëpresje (;). Ky simbol përfaqëson fundin e një instruksioni dhe duhet të vendoset pas çdo instruksioni program në C++. (një nga gabimet më të zakonshme të programuesve në C++ është harrimi i vendosjes së pikëpresjes në fund të çdo instruksioni).

return 0;

Instruksioni `return` bën që funksioni `main()` të përfundojë dhe të kthejë kodin nga i cili ndiqet ky instruksion, në këtë rast 0. Kjo është mënyra më e zakonshme për të përfunduar një program i cili nuk ka ndeshur në gabime gjatë ekzekutimit të tij. Siç do të shohim shohim, të gjithë programet C++ përfundojnë me një instruksion të ngjashëm me këtë.

Deri këtu, mund të vihet re se jo të gjithë rreshtat e këtij program kryejnë një veprim. Ka rreshta të cilët përmbajnë vetëm komente (ato që fillojnë me //), rreshta me instruksione për preprocesorin e kompilatorit (ato që fillojnë me //), rreshta me instruksione për preprocesorin e kompilatorit (ato që fillojnë me #), pastaj ka rreshta që fillojnë me deklarimin e një funksioni (në këtë rast, funksionin `main`) dhe së fundi rreshta me instruksione (si p. sh `cout<<`) të cilët të gjithë përfshihen në një bllok të kufizuar nga kllapat të mëdha ({ }) të funksionit `main`.

Programi është shkruar në disa rreshta për të qënë sa më i lexueshëm, por kjo nuk është e detyrueshme. P.sh., në vend të :

```
int main ()
{
cout<<"Pershendetje te gjithëve!";
return 0;
}
mund të kishim shkruar:
int main () {cout<<"Pershendetje te gjithëve!"; return 0; }
pra në një rresht dhe kjo do të kishte të njëjtin kuptim.
```

Në C++ ndarja ndërmjet instruksioneve realizohet nga pikëpresja (;) pas secilit prej tyre. Ndarja e kodit në disa rreshta shërben vetëm për ta bërë kodin më të lexueshëm dhe me skematik për njerzit që mund ta lexojnë.

Më poshtë jepet një program me më shumë instruksione : **Pershendetje te gjithëve! Program ne C++**

```
//program im i dytë
#include <iostream>
using namespace std;
int main ()
{
cout<<"Pershendetje te gjithëve! ";
cout<<"Program ne C++";
return 0;
}
```

Në këtë rast metoda cout << është përdorur në dy instruksione të ndryshme. Theksojmë sërish se ndarja në rreshta të ndryshëm është bërë për të rritur lexueshmërinë e programit, pasi funksioni main mund të ishte përcaktuar edhe kështu:

```
int main ()
{
cout <<
"Pershendetje te gjithëve! ";
cout
<<"Program ne C++";
}
```

Rezultati edhe në këtë rast do të ishte i njëjtë me shembujt e mëparshëm

Direktivat e preprocesorit (ato që fillojnë me #) përjashtohen nga ky rregull pasi ato nuk janë instruksione të vërteta. Ato që fillojnë që lexohen edhe pastaj fshihen nga preprocesori dhe nuk prodhojnë kod. Këto direktiva duhet të vednosin secila në një rresht më vehte dhe nuk kërkojnë pikëpresje (;) pas tyre.

Komentet

Komentet janë pjesë të kodit të cilat nuk trajtohen nga kompilatori. Ato nuk kryejnë asnjë veprim. Qëllimi i tyre është vetëm që të lejojnë programuesit të vendosin shënimet dhe përshkrimet e tyre në kodin e programit (kodin burim).

C++ lejon dy mënyra për futjen e komenteve:

```
//komentet rresht  
/* komentet bllok */
```

Mënyr e parë, komentet rresht, trajton si koment çdo gjë që vjen pas çifti të slasheve (//) deri në fund të të njëjtit rresht. Mënyra e dytë, komentet bllok, trajton si koment çdo gjë ndërmjet karaktereve /* dhe vendodhjes së parë të karaktereve */ , duke lejuar mundësinë e përfshirjes në koment të disa rreshtave.

Le të shtojmë disa komente në programin e dytë:

```
/* program im i dyte ne C++          Pershendetje te gjithëve! Program ne C++  
me me shume komente */
```

```
#include <iostream>  
int main ()  
{  
    //shkruaj Pershendetje. !  
    cout<<"Pershendetje te gjithëve! ";  
    //shkruaj Program ne C++  
    cout<<"Program ne C++";  
    return 0;  
}
```

Në qoftë se në komentet që vendosen në kodin e programit pa përdorur kombinimet e karaktereve të //, /* ose */ , atëherë kompilatori do ti trajtojë ato si të ishin instruksione në C++ dhe, normalisht do të jepte disa mesazhe gabimi.

Ndryshoret (variablat)

Vlera e programeve të paraqitura në shembullin e sipërm duhet që të merren mirë në konsideratë. Për të shkruar një fjali në ekran, duhet që të shkruajmë më parë disa rreshta program, pastaj ta kompajlojmë dhe së fundi të ekzekutojmë programin e marrë nga kompajllimi. Në këtë rast do të ishte me e lehtë që fjalia e dhënë të shkruhej direkt pa patur nevojën e programit. Por programimi nuk kufizohet vetëm në shtypjen e tekstit në ekran. Në mënyrë që të shkruajmë programe që na e lehtësojnë me të vërtetë punën tonë, duhet të shqyrtojmë konceptin e variablës

Le të supozojmë se ju kërkohet të mbani mend numrin 5, pastaj që të mbani mend edhe numrin 2. Ju keni tani të ruajtur në kujtesën tuaj dy vlera. Tani, në qoftë se ju kërkohet që ti shtoni 1 numrit të parë, atëherë në kujtesën tuaj do të mbani të ruajtur numrat 6 (që është 5+1) dhe 2. Me këto vlera mund të kryejmë një zbritje nga e cila marrim si rezultat vlerën 4.

I gjithë ky process i zhvilluar në mendjen tuaj është i ngjashëm me atë që një kompjuter mund të bëjë me dy variabla. I njëjti proces mund të shërbehet në C++ me radhën e mëposhtme të instruksioneve:

```
a = 5;  
b = 2;  
a = a + 1;  
rezultati = a - b;
```

Është e qartë se ky është një shembull shumë i thjeshtë pasi përdorëm dy numra të plotë të vegjël por mendoni se kompjuteri të ruajë miliona të tillë dhe në të njëjtën kohë të kryejë veprime të vështira matematike me to.

Duket qartë pra, se mund ti përcaktojmë ndryshoret si pjesë të kujtesës që shërbejnë për të ruajtur vlera të caktuara.

Çdo ndryshore ka nevojë për një identifikues për ta dalluar nga të tjerët, p.sh. në kodin e mësipërm identifikuesit ishin a,b dhe rezultati, por ndryshoret mund të emërohen me çdo emër (rradhë karakteresh) që është një identifikues i pranueshëm.

TIPET E THJESHTA TË TË DHËNAVE NE C++

Mvarësisht nga operacionet në të cilat mund të kryhen mbi ndonjë bashkësi të të dhënave dallojmë më shumë tipe të të dhënave. Me termin tip i të dhënave nënkuptojmë bashkësinë e të dhënave $T=(t_1, t_2, \dots, t_n)$ mbi të cilat mund të kryhen cila do operacionet e bashkësisë së operacioneve $O=(o_1, o_2, \dots, o_m)$.

Në C++ bashkësia e numrave të plotë është $T=(-2,147,483,648, \dots, -1, 0, 1, 2, \dots, 2,147,483,647)$ ndërsa bashkësia e operacioneve është $O=(+, -, *, /, \%)$.

Të dhënat që do të shfrytëzohen në program paraprakisht duhet definuar dhe deklaruar, shembull për dy të dhëna x dhe y si numra të plotë në C++ do të kemi definimin e tyre si : ***int x,y;***

Tipet e të dhënave në C++ kanë emrat e veta, si tipe të thjeshtë i kemi:

-int (numrat e plotë)

-float (numrat real)

-char (karakterë)

-string (tekstual)

-bool (logjik)

Tipi	Rangu i vlerave	Madhësia në bajta
unsigned short int	0..65535	2 = 16 bita
short	-32768..32767	2 = 16 bita
unsigned long int	0..4294967295	4 = 32 bita
long int	-2147483648..2147483647	4 = 32 bita
int - 16 bit	-32768..32767	2 = 16 bita
Int - 32 bit	-2147483648..2147483647	4 = 32 bita
unsigned int -16 bit	0..65535	2 = 16 bita
unsigned int - 32 bit	0..4294967295	4 = 32 bita
float	1.2E-38..3.4E38	4 = 32 bita
double	2.2E-308..1.8E308	8 = 64 bita
char	256 simbole	1= 8 bita

Në tipin integer (të plotë) bëjnë pjesë numrat e plotë dhe atë prej numrit $-2,147,483,648$ deri tek numri $2,147,483,647$ pra të gjitha numrat e plotë që mund të shprehen me 32 bita.

Në tipin real bëjnë pjesë numrat real.

Në tipin char (shenjë) bëjnë pjesë të gjitha shenjat që mund të paraqiten me një bajt(8 bita) si p.sh “A”, “#”, “1”, “:”, “g”, “/” etj.

Vargjet e simboleve të ndryshme (shkronjave, numrave dhe simboleve special), përkatësisht tekstet e çfarëdoshme të shkruara brenda thonjzave, quhen stringje dhe llogariten si të dhëna të tipit string. Kështu p.sh të dhëna të tipit string janë:

“Gjuha programuese C++”

“Jeta”

“Y=2*x+10”

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

DEKLARIMI I VARIABLAVE

Në mënyrë që të përdoret një variabël (ndryshore) në C++, duhet që më parë ajo të deklarohet duke treguar se cilit nga tipet e mësipërme i përket. Sintaksa për deklarimin e një variabli të ri është: të shkruhet emri i tipit që duam (si int, short, float...) të ndjekur nga një identifikues i pranueshëm si variabël pra tipi i ndryshores identifikuesi ;

P.sh.: int a;
 float numri;

Këto dy rreshta janë deklarime të sakta variablash. I pari deklaron një variabël të tipit int që ka identifikuesin a. I dyti deklaron një variabël të tipit float me identifikuesin numri. Pasi deklarohen, ndryshoret a dhe numri mund të përdoren në pjesën në vazhdim të zonës (scope) së tyre në program.

Në qoftë se nevojitet të deklarohen disa variabla të të njëjtit tip dhe duam të shkruajmë sa më pak, atëherë ato mund të deklarohen të gjithë në të njëjtin rresht duke i ndarë identifikuesin me presje (,).

P.sh. rreshti:

int a, b, c;

deklaron tre variabla (a, b dhe c) të tipit int dhe ka të njëjtin kuptim sikur të kishim shkruar:

int a;

int b;

int c;

Për të parë se si duket deklarimi i variablave në një program, le të shohim kodin C++ të shembullit të mësipërm mbi kujtesën

```
#include <iostream>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
//deklarimi i variablave
```

```
int a, b;
```

```
float c;
```

```
return 0;
```

```
}
```

REALIZIMI I STRUKTURAVE LINEARE

Strukturat lineare realizohen me anë të tre urdhëresave themelore:

- urdhëresa për dalje,
- urdhëresa për shoqërim,
- urdhëresa për hyrje,

URDHËRESAT PËR DALJE TË TË DHËNAVE

Urdhëresat për paraqitjen (dalje, shtypjen) e të dhënave (konstantave, vlerat e ndryshoreve dhe shprehjeve) në ekran, është :

cout<<shprehja;

cout<<"shprehja tekstuale"

Në listën dalëse mundet të shkruhen konstanta, ndryshore ose shprehje aritmetike të ndara me <<. Gjatë realizimit të urdhëresës në ekran do të paraqiten konstantat e shkruajtur në listën, vlerat e ndryshoreve dhe shprehjeve. Në rastet kur kemi shprehje aritmetike atëherë në fillim njehsohet vlera e shprehjes e mandej ajo vlerë paraqitet. Të dhënat paraqiten në ekran në rreshta. Pasi që të mbushet njëri vazhdohet me rreshtin tjetër.

Urdhëri për kalim në rresht të ri përmes karakterit për rresht të ri \n (angl . new line character) ose po duke përdor endl (end line) .

si në shembullin 1.

cout<<"Mire se erdhet \n";

cout<<"ne c++";

rezultati do te jetë

```
Mire se erdhet
ne c++
```

Të njëjtin rezultat e fitojm edhe duke përdor **endl**

cout<<"Mire se erdhet<<endl;

cout<<ne c++;

shembulli 2.

cout<<2+7;

rezultati do të jetë

```
9
```

dhe

cout<<"2+7";

rezultati do të jetë

```
2+7
```

KARAKTERI PËR TABELIM HORIZONTAL

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

cout << "\t"

P.sh

```
cout<<"pershendetje\t";
rezultati do te jetë
_____pershendetje
```

Shtypja me precizitet të caktuar

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

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

cout.precision(k);

ose manipulatori : ***setprecision(k)***

ku K është numri i shifrave dhjetore.

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

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

Rrumbullaksimi nuk i ndryshon vlerat numerike përkatëse në memorjen e kompjuterit.

URDHËRESA PËR SHOQËRIM

Urdhëresa për shoqërim shfrytëzohet për njehsimin e vlerave të shprehjeve të ndryshme dhe shoqërimin e tyre ndonjë ndryshoreje. Forma e përgjithëshme e kësaj urdhërese është:

ndryshore=shprehje;

ku shenja “=” ka kuptimin e shoqërimit të vlerës së shprehjes nga ana e djathtë e barazimiti ndryshore në anën e majtë të barazimit.

Në këto raste duhet patur kujdes për tipin e ndryshores në anën e majtë të barazimiti dhe vlerës së mundshme të shprehjes në anën e djathtë të barazimit.

sh. Shoqëimi vlerë i ndryshores mund të bëhet gjatë deklarimit si

```
int piket=99;
```

ose po shoqërimi vlerës së ndryshores mund të bëhet pasit ta kemi deklaruar atë

```
int piket;
```

```
.....
```

```
piket=99;
```

sh. me urdhëresën $C=2*A+B$, nëse vlerat A dhe B janë $A=12$, $B=3$ fitojmë $24+3$ pra ndryshore C i shoqërohet vlera 33.

sh. Me $K=I/J$ dhe nëse ndryshoret janë të deklaruar si ndryshore int (integer) atëher kjo urdhëresë nuk do të realizohet pasi që vlera e shprehjes është vlerë reale pra nuk përputhet me tipin e ndryshores në anën e majtë të barazimit.

sh. Shprehjet $I=I+1$ në aspektin matematikor nuk kanë kuptim por në programim shpesh përdoren ku vlera e ndryshores I do ta zëvendësohet me vlerë të re e cila vlerë është vlera e vjetër e saj e zmadhuar për (në këtë rast) 1, pra nëse ka qenë 15 mbas kësaj urdhërese do të jetë 16.

Detyra

1. Të shkruhet program i cili do të paraqes (bën daljen) e mesazhit Mire se erdhet ne C++

```
//Zgjidhja
//Ushtrimi i pare ne C++
//operatori per shtypje "cout"
// kalimi ne rresht te ri "endl"
#include <iostream>
using namespace std;
int main()
{
    cout<< "Mire se erdhet ne C++" <<endl;
    return 0;
}
```

2. Të shkruhet program i cili do të paraqes tekstin “SH.M.K gjimnazi Dr. Ibrahim Temo”

3. Të shkruhet program i cili do të bën mbledhjen e dy numrave dhe të shtyp rezultatin

```
//Mbledhja e dy numrave
// "\n" kalimi ne rresht te ri, njejte si endl
#include <iostream>
using namespace std;
int main()
{
    int a, b, c; // deklarimi i vaiablave
    a = 23; //shoqerimi vlere variables a
    b = 7; // shoqerimi vlere variables b
    c = a + b; // shoqerimi vlere variables c
    cout<< "Shuma e numrave " << a <<" dhe " << b <<" eshte : " << c <<"\n";
    return 0;
}
```

Zgjidhja sipas mënyrës së dytë

```
#include <iostream>
using namespace std;
int main()
{
    int a, b; // deklarimi i vaiablave
    a = 23; //shoqerimi vlere variables a
    b = 7; // shoqerimi vlere variables b
    cout<< "Shuma e numrave " << a <<" dhe " << b <<" eshte : " << a+b<<"\n";
    return 0;
}
```

4. Të shkruhet program i cili bën mbledhjen e tre numrave.

5. Të shkruhet program i cili do të njehson syprinen dhe perimerin e drejtkëndëshit

//Zgjidhja :

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int brinjaA, brinjaB, Syprina, Perimetri;
```

```
    brinjaA=5;
```

```
    brinjaB=10;
```

```
    Syprina=brinjaA*brinjaB;
```

```
    Perimetri=(2*brinjaA)+(2*brinjaB);
```

```
    cout<<"Syprina e drejtkendeshit eshte\t"<<Syprina<<"\tkurse perimetri eshte\t"<<Perimetri;
```

```
    return 0;
```

```
}
```

6. Të shkruhet program i cili njehson notën mesatare (Udhëzim tipi i ndryshore se notes mesatare duhet të deklarohet si float se rezultati mund të jetë numër me presje dhjetore)

URDHËRESA PËR HYRJE TË TË DHËNAVE

Forma e përgjithshme e urdhëresës për hyrje (shfrytëzohet edhe termi urdhëresë për lexim është

cin>>shprehja;

Në listën hyrëse janë një ose më shumë ndryshore të ndarë me presje, të cilave me anë të kësaj urdhërese u shoqërohet vlerë konkrete.

sh1. cin>>v; pas ekzekutimit të kësaj urdhërese, kompjuteri ndalet dhe pret që përmes tastierës t'i jepet vlera ndryshore v.

sh2. Nëse ndryshoret I,J, dhe K janë deklaruar si ndryshore të tipit int, me urdhëresën

```
cin>>I;
```

```
cin>>J;
```

```
cin>>K;
```

paralajmërohet shoqërimi i vlerave për këto ndryshore. Nëse gjatë realizimit të kësaj urdhërese nga tastiera futen këto konstanta: 12 58 99

atëherë ndryshoreve I,J dhe K u shoqërohen vlerat dhe atë I=12, J=58 dhe K=99.

Detyra

1: Të shkruhet program i cili lexon një numer nga tastiera dhe i cili bën daljen e të njëjtit p.sh nëse shkruajm numrin 478 pas porosis Numri qe lexuat nga tastiera është 478

```
#include <iostream>
```

```
using namespace std;
```

```
int main ()
```

```
{
```

```
    int numri; // deklarimi i variables numri
```

```
    cout<<"Lexo nje numer nga tastiera"<<endl; //Shtypet porosis qe e njofton perdoruesin te lexon nje numer
```

```
    cin>>numri; //ketu program ndalon dhe perdoruesin te lexon vleren e ndryshores numri
```

```
    cout<<"Numri qe lexuat nga tastiera eshte \t"<<numri;
```

```
    return 0; }
```

2. Të shkruhet program që bën mbledhjen e dy numrave të lexuar nga tastiera

3. Të shkruhet program i cili njehson syprinen dhe perimetrin e drejtkëndëshit ku gjatesia e brinjëve lexohet nga tastier.

PROGRAMIMI I ORIENTUAR NE OBJEKTE (KLASAT)

Kur flitet për programimin e orientuar në objekte nënkuptojmë klasat si dhe strukturat që deklarohen me shfrytëzimin e tyre. Klasat paraqesin një tip tjetër të strukturave, në të cilat bashkarisht vendosen të dhënat.

Strukturat dhe klasat në gjuhën C++ kanë një dallim të vogël. Derisa qasja e nënkuptuar (ang. default access) te strukturat është publike (ang. public), te klasat kjo qasje e nënkuptuar është private (ang. private).

NUMËRIMET

Në gjuhën programuese C++ të dhënat e caktuara mund të grupohen, siç thuhet përmes *numërimit* (ang. enumeration) të tyre. Kështu, p.sh., mund të grupohen ditët e javës, muajt e vitit, ngjyrat, vlerat logjike etj. Në këtë mënyrë krijohen tipe të reja të të dhënave, të cilat njihen si *tipe të numëruara* (ang. enumerated type). Pastaj, këto të dhëna mund të shfrytëzohen për deklarimin e variablave të tipeve përkatëse, me qëllim të shfrytëzimit të tyre gjatë shkruarjes së programeve të ndryshme.

PËRCAKTIMI I GRUPIT

Grupi i të dhënave përcaktohet duke e shfrytëzuar komandën **enum**, e cila në formë të përgjithshme shkruhet:

```
enum e
{
```

```
    a0,
    a1,
    ...
    an
```

```
};
```

ku janë:

e - emri i grupit.

a0, a1, ..., an - anëtarët e grupit.

Anëtarët e grupit quhen edhe *numërues* (ang. enumerator) dhe në fakt paraqesin *konstante të emëruara*. Gjatë grupimit, çdo anëtar i të grupit kompjuteri automatikisht i shoqëron një *vlerë të nënkuptuar* (ang. default value) në formë të numrit, duke filluar me vlerën **0**, e cila pastaj për çdo anëtar vijues rritet për **1**. Emri i grupit (**e**) si dhe emrat e anëtarëve që përfshihen në grup (**a0, a1, ..., an**) formohen në bazë të rregullave që vlejnë për identifikatorët.

Shembull 1. Programi **enum1a**, në të cilin definohet grupi **java**, i përbërë prej ditëve të javës.

```
// Programi enum1a
#include <iostream>
using namespace std;
enum java
{
    hene,
    marte,
    merkure,
    enjte,
    prente,
    shtune,
    diel
};
```

```
int main()
{
}
```

Me komandën **enum** këtu definohet grupi **java**, në të cilin përfshihen identifikatorë që kanë të bëjnë me ditët e javës. Gjatë përcaktimit të emrave të ditëve të javës, nuk janë shfrytëzuar shkronjat e alfabetit shqip, meqë rregullat për krijimin e identifikatorëve e ndalojnë. Çdo identifikatori që përfshihet në grup, ashtu siç u tha edhe më sipër, kompjuteri i shoqëron një vlerë numerike. Kështu, identifikatorit **hene** ia shoqëronë vlerën **0**, identifikatorit **marte** - vlerën **1** dhe kështu me radhë. Për këtë arsye, *anëtarët e grupit paraqesin konstante*, dhe, kur flitet për grupime përmes komandës **enum**, duhet nënkuptuar *grupimin e konstanteve*. Këtu, pjesa e programit është e zbrazët dhe grupi i përcaktuar nuk është përdorur për asgjë. Grupi mund të përcaktohet duke e vendosur edhe brenda programit. Gjatë kësaj, programi i dhënë më sipër do të duket si në vijim.

```
// Programi enum1b
#include <iostream>
using namespace std;
int main()
{
enum java
{
    hene,
    marte,
    merkure,
    enjte,
    premte,
    shtune,
    diel
};
}
```

Në këtë rast, grupi ka karakter lokal dhe mund të shfrytëzohet vetëm brenda programit, por jo edhe jashtë tij.

SHFRYTËZIMI I GRUPIT

Pasi të jetë përcaktuar grupi, përmes emrit të tij mund të deklarohen *variabla të tipit të grupit*. Forma e deklarimit të variablave është plotësisht e njëjtë me deklarimin e variablave të tipeve standarde dhe në rast të përgjithshëm duket:

e v;

ku janë: **e** - emri i grupit.

v - variabla që deklarohet e tipit të grupit të përcaktuar.

Shembull 2. Programi **enum2**, në të cilin definohet dhe shfrytëzohet grupi **java**, i përbërë prej ditëve të javës.

```
// Programi enum2
#include <iostream>
using namespace std;
enum java
{
    hene,
    marte,
```

```

    merkure,
    enjte,
    premte,
    shtune,
    diel
};
int main()
{
    java dita; // deklarohet ndryshorja dita e tipit java
    dita=marte;
    cout << "\nDitës së martë i shoqërohet numri: " << dita << "\n\n";
    return 0;
}

```

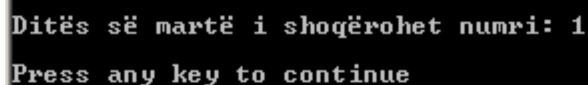
Në program, përmes deklarimit: **java dita;**

variabla **dita** deklarohet e tipit **java** dhe në të mund të ruhen të dhënat e përfshira brenda kllapave të grupit në fjalë. Nëse ekzekutohet programi i dhënë, meqë përmes komandës:

```
dita=marte;
```

variablës **dita** të tipit **java** i është shoqëruar anëtari i dytë në grup, të cilit i korrespondon vlera numerike **1**, rezultati në ekran do të duket si në *Fig.1*.

Fig. 1.
Pamja e ekranit pas
ekzekutimit të programit
enum2



```

Ditës së martë i shoqërohet numri: 1
Press any key to continue

```

Variablës **dita** mund t'i shoqërohen vetëm anëtarët e grupit. Kështu, pavarësisht nga vlerat që u përkasin anëtarëve të grupit, shoqërimi direkt i vlerave nuk lejohet, përkatësisht kompjuteri do të lajmërojë gabim, nëse p.sh., në program shkruhet shprehja:

```
dita=1;
```

Anëtarët e grupit, gjatë përcaktimit të tij, mund të shkruhen në disa rreshta (ashtu siç u dha më sipër), ose edhe në vetëm një rresht, p.sh.:

```
enum java {hene,marte,merkure,enjte,premt,e,shtune,die};
```

DEGËZIMI PËRMES VLERAVE TË NUMËRUARA

Vlerat e numëruara mund të shfrytëzohen për realizimin e degëzimeve të ndryshme edhe atë duke e përdorur komandën **if**, ose komandën **switch**.

DEGËZIMI ME KOMANDËN IF

Variablat, të cilat deklarohen si variabla të tipit të numëruar në kombinim me operatorët relationalë, mund të shfrytëzohen për realizimin e degëzimeve të ndryshme përmes komandës **if**.

Shembull 3. Programi **enum3**, në të cilin tregohet degëzimi përmes komandës **if**, duke e krahasuar variablën e numëruar **koha**.

```
// Programi enum3
#include <iostream>
```

```
using namespace std;
enum rezultati
{
    Po,
    Jo
};
int main()
{
    rezultati koha;
    koha=Po;
    cout << "\nRezultati është ";
    if (koha==Po)
        cout << "pozitiv";
    else
        cout << "negativ";
    cout << "\n\n";
    return 0;
}
```

Në program, fillimisht është përcaktuar grupi **rezultati**, me dy anëtarët e tij **Po** dhe **Jo**. Pastaj, pas deklarimit të variablës **koha** të tipit rezultati, me shprehjen:
koha=Po;

variablës i shoqërohet vlera **Po** e grupit. Në fund, përmes komandës **if**, kontrollohet se a është barazi vlera e variablës **koha** me vlerën **Po** dhe varësisht nga raporti i tyre shtypet fjala **pozitiv** ose **negativ**. Në këtë rast, meqë variablës **koha** i është ndarë vlera **Po**, në ekran do të shtypet fjalia:

Rezultati është pozitiv

ku me ngjyrë të zezë është theksuar fjala që shtypet, duke kaluar në degën e parë të komandës **if**.

DEGËZIMI ME KOMANDËN SWITCH

Anëtarët e grupeve zakonisht shfrytëzohen për realizimin e degëzimeve të shumëfishta përmes komandës **switch**.

Shembull 4. Programi **enum4a**, përmes së cilit tregohet shfrytëzimi i komandës **switch** për degëzim të shumëfishtë në bazë të vlerave të anëtarëve të grupit **java**.

```
// Programi enum4a
#include <iostream>
using namespace std;
enum java
{
    hene,
    marte,
    merkure,
    enjte,
    premtë,
    shtunë,
    diel
}
```

```
};
int main()
{
    java dita;
    dita=marte;
    cout << "\nDita që u zgjodh është dita e ";
    switch (dita)
    {
        case hene: cout << "hënë";break;
        case marte: cout << "martë";break;
        case merkure: cout << "mërkurë";break;
        case enjte: cout << "enjte";break;
        case premte: cout << "premtë";break;
        case shtune: cout << "shtunë";break;
        case diel: cout << "diel";break;
    }
    cout << "\n\n";
    return 0;
}
```

Këtu, me komandën **enum** para programit, definohet grupi **java**. Pastaj, duke e shfrytëzuar komandën **switch**, realizohet degëzimi në bazë të vlerave që u shoqërohen anëtarëve që përfshihen në grup. Kështu, meqë përmes shoqërimit:

```
dita=marte;
```

variablës **dita** të tipit **java** i shoqërohet numri rendor **1**, i cili i përket ditës së martë, me komandën për degëzim zgjedhet dega e dytë:

```
case marte: cout << "martë";break;
```

me çka pjesës së fjalisë:

Dita që u zgjodh është dita e

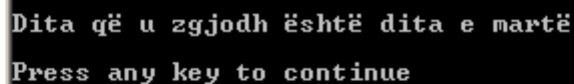
e cila shtypet para komandës **switch**, i shtohet edhe fjala **martë**, për ta fituar formën definitive të fjalisë:

Dita që u zgjodh është dita e **martë**

gjë që shihet edhe në rezultatin që me atë rast shtypet në ekran (shih *Fig.2*).

Fig.2.

Pamja e ekranit pas ekzekutimit të programit enum4a



```
Dita që u zgjodh është dita e martë
Press any key to continue
```

Rezultati do të jetë i njëjtë nëse programi **enum4a** shkruhet ashtu siç është dhënë në vijim, tek i cili shfrytëzohet variabla **t** e tipit **string**.

```
// Programi enum4b
#include <iostream>
#include <string>
```

```

using namespace std;
enum java
{
    hene,
    marte,
    merkure,
    enjte,
    premte,
    shtune,
    diel
};
int main()
{
    java dita;
    string t;
    dita=marte;
    switch (dita)
    {
        case hene: t= "hënë";break;
        case marte: t= "martë";break;
        case merkure: t= "mërkurë"; break;
        case enjte: t= "enjte";break;
        case premte: t= "premtë";break;
        case shtune: t= "shtunë";break;
        case diel: t= "diel";break;
    }
    cout << "\nDita që u zgjodh është dita e " << t << "\n\n";
    return 0;
}

```

Në program, fillimisht, është deklaruar variabla **t** e tipit **string**, në të cilën ruhen emrat e ditëve të javës. Meqë deklarimi **string** nuk është standard, për ta shfrytëzuar këtë deklaram në ballinën e programit është vendosur komanda paraprocesorike **#include <string>**, sepse në modulën **string** është përcaktuar deklarimi në fjalë. Edhe në këtë rast, pas ekzekutimit të programit të dhënë, rezultati në ekran do të duket si ai që u dha në *Fig.2.2*. Rezultati i programit të dhënë më sipër nuk ndryshon nëse degët e veçanta të komandës **switch** shkruhen edhe kështu:

```

switch (dita)
{
    case 0: t= "hënë";break;
    case 1: t= "martë";break;
    case 2: t= "mërkurë";break;
    case 3: t= "enjte";break;
    case 4: t= "premtë";break;
    case 5: t= "shtunë";break;
    case 6: t= "diel";break;
}

```

Në këtë rast, në vend të emrave të ditëve të javës, meqë kemi të bëjmë me konstante, janë shfrytëzuar vlerat e nënkuptuara të tyre.

DISA VARIABLA TË NUMËRUARA TË TIPIT TË NJËJTË

Sikurse te variablat e tipeve standarde, brenda një programi mund të deklarohen edhe disa variabla të numëruara të tipit të njëjtë. Gjatë kësaj, komanda përkatëse për deklarim në formë të përgjithshme do të duket:

e v1,v2,...,vn;

ku janë:

e - emri i grupit.

v1,v2,...,vn - variablat që deklarohen të tipit të grupit të përcaktuar.

Shembull 5. Programi **enum5**, përmes së cilit tregohet deklarimi i dy variablave të numëruara të tipit **java**.

```
// Programi enum5
#include <iostream>
using namespace std;
enum java
{
    hene,
    marte,
    merkure,
    enjte,
    premtë,
    shtunë,
    diel
};
int main()
{
    java punuese;
    java pushuese;
    punuese=hene;
    pushuese=shtunë;
    cout << "\nDitët e punës fillojnë me numrin: " << punuese
    << "\n\nDitët e pushimit fillojnë me numrin: " << pushuese
    << "\n\n";
    return 0;
}
```

Këtu, përmes komandave:

```
java punuese;
java pushuese;
```

variablat **punuese** dhe **pushuese** janë deklaruar si variabla të numëruara të tipit **java**. Në këto variabla mund të ruhen të gjitha vlerat e mundshme që përfshihen në grupin në fjalë, pavarësisht se për cilën ditë të javës kemi të bëjmë. Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket si në *Fig.3*.

Fig.3.

Pamja e ekranit pas

ekzekutimit të programit enum5

```
Ditët e punës fillojnë me numrin: 0
Ditët e pushimit fillojnë me numrin: 5
Press any key to continue
```


Nënkuptohet se edhe variablat e numëruara mund të deklarohen duke e shfrytëzuar vetëm një komandë:

```
java punuese, pushuese;
```

plotësisht njëloj siç deklarohen variablat standarde.

PËRCAKTIMI DHE DEKLARIMI NJËKOHËSISHT

Gjatë përcaktimit të grupit me të dhëna të numëruara, njëkohësisht mund të bëhet edhe deklarimi i variablës përkatëse. Për këtë qëllim, komanda **enum** shkruhet:

```
enum e  
{  
    a0,  
    a1,  
    ...  
    an  
}
```

```
v;
```

ku janë:

e - emri i grupit.

a0, a1, ..., an - anëtarët e grupit.

v - variabla e tipit të grupit që përcaktohet.

Shembull 6. Programi **enum6**, përmes së cilit tregohet përcaktimi i grupit **viti**, si dhe njëkohësisht deklarohet variabla **stina** e tipit që definohet.

```
// Programi enum6
```

```
#include <iostream>
```

```
using namespace std;
```

```
enum viti
```

```
{  
    pranvera,  
    vera,  
    vjeshta,  
    dimri  
}
```

```
stina;
```

```
int main()
```

```
{
```

```
    stina=vera;
```

```
    if (stina==pranvera)
```

```
        cout << "\nStina e pranverës";
```

```
    else
```

```
        if (stina==vera)
```

```
            cout << "\nStina e verës";
```

```
    else
```

```
        if (stina==vjeshta)
```

```
            cout << "\nStina e vjeshtës";
```

```

        else
            cout << "\nStina e dimrit";
    cout << "\n\n";
    return 0;
}

```

Siç shihet më sipër, para programit është përcaktuar grupi **viti**, duke e shënuar në fund edhe variablën **stina**. Në këtë mënyrë kompjuteri njoftohet se variabla **stina** është deklaruar variabël e tipit **viti** dhe si e tillë shfrytëzohet në program.

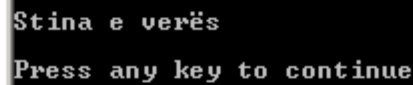
Nëse ekzekutohet programi në fjalë, meqë në fillim të tij paraqitet shprehja:

```
stina=vera;
```

rezultati që fitohet në ekran do të duket si në *Fig.2.4*

Fig.2.4

*Pamja e ekranit pas ekzekutimit të programit
enum6*



```

Stina e verës
Press any key to continue

```

Gjatë përcaktimit të grupit, njëkohësisht mund të deklarohen edhe më shumë variabla të atij tipit, duke e shkruar komandën **enum** në formë të përgjithshme:

```

enum e
{
    a0,
    a1,
    ...
    an
}
v1,v2,...,vm;

```

ku janë:

e - emri i grupit.

a0, a1, ..., an - anëtarët e grupit.

v1,v2,...,vm - variablat e tipit të grupit që përcaktohet.

Shembull 7. Programi **enum7**, përmes së cilit tregohet përcaktimi i grupit **viti**, si dhe njëkohësisht deklarohen variablat **stina** dhe **koha** të atij tipi.

```

// Programi enum7
#include <iostream>
using namespace std;
enum viti
{
    pranvera,
    vera,
    vjeshta,
    dimri
}
stina,koha;
int main()
{
    stina=vera;
    koha=dimri;
}

```

```

if (stina==koha)
    cout << "\nPerioda e njëjtë";
else
    cout << "\nPerioda ndryshon";
cout << "\n\n";
return 0;
}

```

Këtu, menjëherë pas komandës paraprocesorike është përcaktuar grupi **viti**, si dhe njëkohësisht janë deklaruar të tipit **viti** variablat **stina** dhe **koha**. Pastaj, në fillim të tij janë marrë vlerat:

```

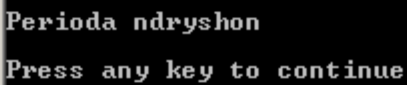
stina=vera;
koha=dimri;

```

Pas ekzekutimit të komandës **if**, për vlerat e marra të variablave **stina** dhe **koha**, rezultati që shtypet në ekran do të duket si në *Fig.2.5*.

Fig.2.5

*Pamja e ekranit pas ekzekutimit të programit
enum7*



```

Perioda ndryshon
Press any key to continue

```

SHOQËRIMI DIREKT I VLERAVE

Vlerat e nënkuptuara të anëtarëve të grupit mund edhe të ndryshohen, duke u shoqëruar direkt vlerat numerike. Ky shoqërim në formë të përgjithshme duket:

```

enum e
{
    a0=k0,
    a1=k1,
    ...
    an=kn
};

```

ku janë:

e - emri i grupit.

a0, a1, ..., an - anëtarët e grupit.

k1,k2,...,kn - konstantet që u shoqërohen anëtarëve të grupit.

Gjatë këtij shoqërimi vlerat e konstanteve zgjedhen sipas dëshirës, por mund të merren edhe të barabarta me vlerat e nënkuptuara të tyre.

Shembull 9. Programi **enum9**, përmes së cilit tregohet krahasimi i vlerave të anëtarëve të grupit.

```

// Programi enum9
#include <iostream>
using namespace std;
enum nota
{
    pese=5,
    gjashte=6,
    shtate=7,
    tete=8,
    nente=9,
    dhjete=10
};

```

```

int main()
{
nota programim,vizatim;
programim=tete;
vizatim=nente;
cout << "\nNota në programim është " << programim << "\n";
cout << "Nota në vizatim është " << vizatim << "\n";
if (programim < vizatim)
    cout << "\nNota në vizatim është më e madhe";
else
    cout << "\nNota në vizatim s'është më e madhe";
cout << "\n\n";
return 0;
}

```

Në program fillimisht është deklaruar grupi **nota**, në të cilin janë përfshirë emrat e notave të mundshme dhe vlerat e tyre. Pastaj, përmes komandave:

```

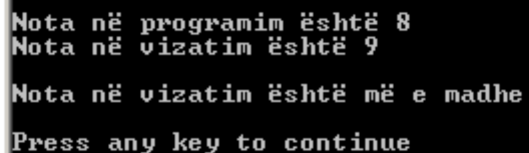
nota programim,vizatim;
    programim=tete;
    vizatim=nente;

```

së pari deklarohen variablat **programim** e **vizatim** të tipit **nota**, dhe pastaj atyre u shoqërohen vlerat **tete** dhe **nente**. Në fund, pasi shtypen vlerat e variablave në fjalë, përmes komandës **if** ato krahasohen dhe, duke e pasur parasysh raportin e tyre, shtypet rezultati i cili shihet në Fig.2.9.

Fig.2.9

Pamja e ekranit pas ekzekutimit të programit enum9



```

Nota në programim është 8
Nota në vizatim është 9

Nota në vizatim është më e madhe
Press any key to continue

```

KLASAT

Kur flitet për *programimin e orientuar në objekte* (ang. object-oriented programming), ose shkurt - *programimin me objekte*, gjithnjë mendohet në *klasat* si dhe në *objektet* që deklarohen me shfrytëzimin e tyre. Klasat paraqesin një tip tjetër të strukturave, në të cilat bashkërisht vendosen të dhënat dhe funksionet që i shfrytëzojnë ato të dhëna. Por, në gjuhën C++, *strukturat dhe klasat kanë një dallim të vogël*. Derisa *qasja e nënkuptuar* (ang. default access) te strukturat është *publike* (ang. public), te klasat kjo qasje e nënkuptuar është *private* (ang. private), gjë që do të shpjegohet në pjesët vijuese.

Përmes klasave jepet një mundësi e shkruarjes së programeve, të cilët sipas nevojës, lehtë ndryshohen, duke i ndryshuar vetëm klasat. Gjatë kësaj, problemi që zgjidhet copëtohet në klasa dhe në deklarimin e objekteve përkatëse, gjë që ka një rëndësi të veçantë kur kemi të bëjmë me programe komplekse, me çka zvogëlohet mundësia e gabimeve.

Programimi i zakonshëm, në të cilin shfrytëzohen vetëm funksionet, ndryshe quhet edhe *programim procedural*. Përmes programimit me objekte, më lehtë modelohet bota reale, krahasuar me programimin procedural, meqë brenda objekteve përfshihen funksionet dhe të dhënat të cilat ato i shfrytëzojnë.

Në fillim të pjesës vijuese, me qëllim të krahasimit të strukturave dhe të klasave, do të shfrytëzohen shembujt e programeve elementare, të cilët janë marrë gjatë shpjegimit të strukturave.

Definimi i klasave te zakonshme

Te gjitha format e definimit te strukturave vlejne edhe per definimin e klasave te zakonshme.

Nje strukture ne formen e pergjithshme duket si ne vijim:

```
struct e
{
    t1 x1;
    t2 x2;
    .....
    tn xn;
}
```

e – emri i struktures.

t1, t2,, t3 – tipet e te dhenave ne komponentet e struktures.

x1, x2,, xn – variablat ne komponentet e struktures.

Definimi i struktures se dhene si klase do te behet:

```
class e
{
    public:

    t1 x1;
    t2 x2;
    .....
    tn xn;
}
```

e – emri i klases.

t1, t2,, t3 – tipet e te dhenave ne komponentet e klases.

x1, x2,, xn – variablat ne komponentet e klases.

Siç e shohim me siper, ne vend te fjales struct eshte perdorur fjala class, si dhe perpara komponenteve te klases eshte perdorur fjala public.

Fjala public eshte perdur tek klasa e mesiperme sepse klasa eshte me qasje private, ndersa struktura eshte me qasje publike. Andaj nese nuk shkruajm shkruajme tek struktura public ateher ajo do te nenkuptohet vet nga kompjuteri si publike dhe klasa nenkuptohet nga vet kompjuteri si private.

Shembull 1.: Programi class1, ne te cilin eshte definuar klasa person, ku perfshihen te dhenat e tipeve te ndryshme te nje personi, siç jane: emri, viti i lindjes dhe qyteti i lindjes.

```
//Programi class1
#include <iostream>
using namespace std;
class person
{
    public:
    char emri[8];
    char qyteti[10];
    int viti;
};
int main()
{
}
```

Pas ekzekutimit të këtij programi nuk do të shfaqet asgjë në ekran, sepse nuk përfshihet asnjë komandë brenda trupit të programit.

Deklarimi i objekteve

Pas definimit të klases kompjuteri nuk rezervon vende në memorie për komponentet që përfshihen brenda klases .

Por me klasën krijohet një tip i ri i, i cili pastaj mund të shfrytëzohet për deklarin e objekteve të asaj klase.

Deklarimi i objekteve të klases bëhet njësoj siç deklarohen variablat e strukturave, ose edhe variablat e tipeve të zakonshme. Por këtu në vend të variablen deklarohet një objekt, ku do të duket si në vijim:

```
e o;
```

ku janë:

e – emri i klases;

o – objekti i tipit të klases e.

Shembull 2.: Programi class2, në të cilin shihet definimi i klases person dhe shfrytëzimi i saj për deklarin e objektit nxenesi të tipit të klases person.

```
//Programi class2
#include <iostream>
using namespace std;
class person
{
    public:
    char emri[8];
    char qyteti[10];
    int viti;
};
int main()
{
    person nxenesi;
```

}
Ketu, përmes shprehjes:

person nxenesi;

deklarohet objekti nxenesi i klases person, i cili në fakt paraqet një kopje të klases që është definuar më parë.

Pas këtij deklarimi në memorien e kompjuterit rezervohen vende për variablat të cilat paraqiten në komponentet e klases.

Qasja të komponentet e klases

Komponenteve të klases mund t'u qasemi pasi të jete deklaruar objekti perkates. Për qasje në komponente të klases shfrytëzohen shprehjet e formës:

```
o.x
```

ku janë:

o – objekti i deklaruar i klases.

x – variabla ose funksioni në komponenten e klases.

. – operatori pike (*ang. dot operator*) për qasje të variablen ose funksionit të komponentes së klases.

Shembull 3.: Programi class3

```
//Programi class3
#include <iostream>
using namespace std;
class person //deklarimi i klases 'person'
{
    public: //deklarimi publik ose privat
    char emri[10];
    char qyteti[10];
    int viti;
};
int main() //funksioni main()
{
    person nxenesi; //klasa person, objekti nxenesi
    cout<<"Emri.....: " <<endl; //japim emrin
    cin>> nxenesi.emri;
    cout<<"Qyteti.....: " <<endl; //japim qytetin
    cin>> nxenesi.qyteti;
    cout<<"Viti.....: " <<endl; //japim vitin
    cin>> nxenesi.viti;
    //shfaqim ne ekran personin e rregjistruar
    cout<<"\nPersoni i rregjistruar eshte: ";
    cout<<"\nEmri.....: " <<nxenesi.emri;
    cout<<"\nQyteti.....: " <<nxenesi.qyteti;
    cout<<"\nViti.....: " <<nxenesi.viti;
    << "\n\n";
    return 0;
}
```

Në program, si edhe te strukturat, tri variabla të përfshira në objektin **nxenesi**, i cili është deklaruar i klasës **person**, u qasemi duke i shënuar ato në format:

```
nxenesi.emri
nxenesi.qyteti
nxenesi.viti
```

Nëse ekzekutohet programi i dhënë dhe pas mesazheve përkatëse, përmes tastierës kompjuterit i jepen vlerat hyrëse **Valmira, Ohri dhe 1983**, në ekran do ta kemi pamjen e cila shihet në *Fig.4.1*.

Fig.4.1

Pamja e ekranit pas ekzekutimit të programit class3

```
Të dhënat nga tastiera
Emri .....: Ualmira
Qyteti ...: Ohri
Viti .....: 1983

Të dhënat e lexuara
Emri .....: Ualmira
Qyteti ...: Ohri
Viti .....: 1983
Press any key to continue
```

FORMA E PËRGJITHSHME E KLASAVE

Zakonisht, kur flitet për klasat, nënkuptohet se komponentet e tyre përmbajnë *variabla* dhe *funksione* të tipeve të caktuara. Specifikuesit e qasjes së tyre, përveç **public**, që u shpjegua më sipër, mund të jenë edhe **private** ose **protected**. Këtu, fillimisht, do të flitet për dy tipet e para të specifikuesve, kurse për specifikuesin **protected** do të bëhet fjalë më vonë.

Forma e përgjithshme e definimit të klasës mund të duket:

```
class e
{
    private:
        t1 y1;
        t2 y2;
        .....
        tn yn;
    public:
        tp zp;
        tq zq;
        .....
        ts zs;
};
```

ku janë:

e - emri i klasës.

t1, t2, ..., ts - tipet e variablave ose të funksioneve në komponentet e klasës.

y1, y2, ..., yn - variablat ose funksionet në komponentet e klasës, të deklaruara si publike.

zp, zq, ..., zs - variablat ose funksionet në komponentet e klasës, të deklaruara si private.

Radha e shkruarjes së specifikuesve të qasjes brenda definicionit të klasës nuk ka rëndësi, si dhe specifikuesit e veçantë mund të shfrytëzohen edhe disa herë, gjë që nuk është e nevojshme.

Variablat e tipeve të caktuara që përfshihen në klasë, njihen si *komponente të dhënash* (ang. data components), ose *anëtarë të dhënash* (ang. data members). Kurse funksionet që përfshihen në klasë njihen si *komponente funksionesh* (ang. function components), ose *anëtarë funksionesh* (ang. member functions), ose edhe *metoda* (ang. methods). Të gjitha komponentet ose anëtarët brenda klasës me një fjalë mund të quhen *komponente të klasës* (ang. class components), ose *anëtarë të klasës* (ang. class members).

Zakonisht, komponentet me të dhëna deklarohen si private, kurse komponentet e funksioneve - si publike. Por, kjo nuk është e thënë, sepse brenda klasave njëkohësisht mund të deklarohen funksione dhe variabla private dhe publike.

Deklarimi i komponenteve të klasës si private nuk ka të bëjë me atë se të dhënat janë sekrete dhe si të tilla nuk duhet të shihen nga të tjerët. Por, kjo lidhet me pengimin e shfrytëzimit direkt të tyre me qëllim të eliminimit të gabimeve të mundshme gjatë shkruarjes së programeve të gjata.

Komponentet, të cilat brenda klasës deklarohen si private, nuk mund t'u qasemi direkt nga jashtë. Ato mund të shfrytëzohen direkt vetëm brenda funksioneve të klasës, pavarësisht se a kemi të bëjmë me funksione private ose publike. Kurse qasja te komponentet që deklarohen si publike është e lirë, si brenda klasës ashtu edhe në program.

Shembull 4.: Programi **class4a**, përmes së cilit definohet klasa **Jeta**, në komponentet e së cilës paraqiten variablat **m** dhe **a**, njëra e deklaruar si private, kurse tjetra - si publike, si dhe funksionet **vlera** dhe **shtypja** të deklaruara si publike.

```
// Programi class4a
#include <iostream>
using namespace std;
class Jeta
{
    private:
        int m;
    public:
        double a;
        void vlera(int k)
```



```

    {
        m=k;
    }
    void shtypja()
    {
        cout << "\nVlera e variablës private m=" << m << "\n";
    }
};
int main()
{
    Jeta Dita;
    Dita.vlera(77);
    cout << "\nLeximi i vlerës së variablës a: ";
    cin >> Dita.a;
    Dita.shtypja();
    cout << "\nVlera e variablës publike a=" << Dita.a << "\n\n";
    return 0;
}

```

Meqë variabla **m** është deklaruar si variabël private, asaj nuk mund t'i qasemi direkt nga jashtë. Prandaj, për ta inicializuar atë me një vlerë shfrytëzohet funksioni **vlera**, me parametrin formal **k**, të cilin mund ta shfrytëzojmë nga jashtë sepse është deklaruar si funksion publik. Brenda funksionit paraqitet shprehja **m=k**, përmes së cilës vlera e variablës **k** vendoset te variabla **m**.

Nga ana tjetër, variabla **a** dhe funksionet **vlera** dhe **shtypja** janë deklaruar si publike, gjë që e nënkupton se atyre mund t'u qasemi direkt nga jashtë, përkatësisht nga programi.

Në fillim të programit kryesor, përmes komandës:

```
Jeta Dita;
```

deklarohet objekti **Dita** i klasës **Jeta**. Pastaj, duke e thirrur funksionin **vlera**:

```
Dita.vlera(77);
```

ku në vend të parametrin formal **k** është shënuar parametri aktual **77**, variablës **m** i shoqërohet kjo vlerë. Kurse, përmes komandave:

```
cout << "\nLeximi i vlerës së variablës a: ";
cin >> Dita.a;
```

lexohet vlera e variablës **a**, e cila brenda klasës është deklaruar si publike, vlerë të cilën kompjuterit duhet t'ia japim përmes tastierës.

Për shtypje të vlerës së variablës **m**, meqë nuk mund t'i qasemi direkt, sepse është deklaruar si private, shfrytëzohet funksioni **shtypja**, i cili është deklaruar si funksion publik, duke e thirrur kështu:

```
Dita.shtypja();
```

Funksioni **shtypja** ka qasje te variabla **m**, pavarësisht se ajo është deklaruar si private, sepse ai gjendet në njërën nga komponentet e klasës.

Në fund të programit, për shtypje të vlerës së variablës **a**, lirisht është shfrytëzuar komanda:

```
cout << "\nVlera e variablës publike a=" << Dita.a << "\n\n";
```

sepse në të kemi qasje direkte për shkak të deklarimit të saj si publike.

Nëse, p.sh., përmes tastierës kompjuterit për variablën **a** ia japim vlerën hyrëse **44.56**, në ekran do ta kemi pamjen e dhënë në *Fig.4.2*.

Fig.4.2 Pamja e ekranit pas ekzekutimit të programit *class4a*

```

Leximi i vlerës së variablës a: 44.56
Vlera e variablës private m=77
Vlera e variablës publike a=44.56
Press any key to continue

```

Funksioni **shtypja** mund të shfrytëzohet edhe për shtypjen e vlerës së variablës **a**, ashtu siç është dhënë në vijim në versionin **class4b** të programit.

```
// Programi class4b
#include <iostream>
using namespace std;
class Jeta
{
private:
    int m;
public:
    double a;
    void vlera(int k)
    {
        m=k;
    }
    void shtypja()
    {
        cout << "\nVlera e variablës private m=" << m
        << "\n\nVlera e variablës publike a=" << a << "\n\n";
    }
};
int main()
{
    Jeta Dita;
    Dita.vlera(77);
    cout << "\nLeximi i vlerës së variablës a: ";
    cin >> Dita.a;
    Dita.shtypja();
    return 0;
}
```

Nëse ekzekutohet ky version i programit, për vlerë hyrëse të njëjtë, rezultati në ekran do të duket si ai që u dha në *Fig.4.2*.

Forma e përgjithshme e dhënë më sipër për definimin e klasave, plotësisht njëjloj mund të shfrytëzohet edhe te strukturat, përfshirë edhe specifikesit e qasjes **public**, **private** dhe **protected**.

Definimi i funksioneve jashtë klasës

Siç u shpjegua te strukturat, edhe te klasat funksionet mund të definohen jashtë trupit të klasave. Por, brenda definicionit të tyre duhet të *shënohen prototipet e funksioneve*.

Shembull 4.:Programi **class4c** si version i programit **class4b**, tek i cili brenda klasës **Jeta** janë deklaruar vetëm prototipet e funksioneve **vlera** dhe **shtypja**, kurse definimi i tyre është bërë jashtë definicionit të klasës.

```
// Programi class4c
#include <iostream>
using namespace std;
class Jeta
{
```

```

private:
int m;
public:
    double a;
    void vlera(int k);
    void shtypja();
};
int main()
{
Jeta Dita;
Dita.vlera(77);
cout << "\nLeximi i vlerës së variablës a: ";
cin >> Dita.a;
Dita.shtypja();
return 0;
}
void Jeta::vlera(int k)
{
    m=k;
}
void Jeta::shtypja()
{
    cout << "\nVlera e variablës private m=" << m << "\n\nVlera
e variablës publike a=" << a << "\n\n";
}

```

Nga definicionet e funksioneve shihet se, plotësisht njëloj si edhe te funksionet që përfshihen në struktura, para emrave të tyre shënohet edhe emri **Jeta** i klasës, i shoqëruar me *operatorin për zbërthim të dukjes* (ang. scope resolutin operator), që shënohet me katër pika (::).

Kjo formë e shkruarjes së funksioneve jashtë definicionit të klasës, sidomos kur funksionet përmbajnë më shumë komanda, është më praktike për shkak të dukshmërisë më të mirë të definicionit të klasës.

Funksionet që përfshihen brenda klasave mund të jenë edhe të tipit **inline**. Kështu, p.sh., nëse funksioni **vlera** te programi i fundit **class4c** merret i tipit **inline**, prototipi i tij brenda definicionit të klasës **Jeta** duhet të shkruhet në formën:

```
inline void vlerat(int k,double x);
```

Gjithashtu, titulli përkatës gjatë definimit të funksionit duhet të shkruhet:

```
inline void Jeta::vlerat(int k,double x)
```

Shfrytëzimi i funksioneve të tipit **inline** lidhet me rritjen e shpejtësisë së ekzekutimit të tyre. Por, gjatë kësaj rritet edhe programi, sepse, kur thirren funksionet e këtij tipi, ato bëhen pjesë e programit.

FORMA TË TJERA TË INICIALIZIMIT TË VARIABLAVE

Për inicializimin e variablave që paraqiten si anëtarë të klasës mund të shfrytëzohen edhe mënyra të tjera. Njëra mënyrë është ajo e leximit nga jashtë të vlerave të tyre, duke definuar brenda klasave funksione me komanda për lexim.

Por, nëse variablat vendosen vetëm në pjesën publike të klasave, inicializimi i tyre mund të bëhet edhe gjatë deklarimit të objekteve.

INICIALIZIMI PËRMES LEXIMIT

Një mënyrë tjetër e inicializimit të variablave, në veçanti atyre të deklaruara si private, përveç asaj që u shfrytëzua më sipër, është ajo e leximit të vlerave të tyre. Për këtë qëllim, komandat për lexim duhet të përfshihen në funksionin e veçantë brenda klasës, duke e deklaruar atë si funksion publik.

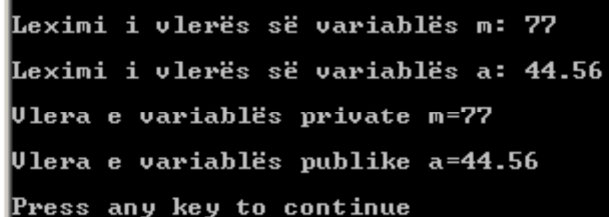
Shembull 5.: Programi **class5a** si version i programit **class4c**, tek i cili brenda klasës **Jeta** është definuar funksioni **leximi** si publik, përmes së cilit lexohet vlera e variablës private **m**.

```
// Programi class5a
#include <iostream>
using namespace std;
class Jeta
{
    private:
        int m;
    public:
        double a;
        void leximi();
        void shtypja();
};
int main()
{
    Jeta Dita;
    Dita.leximi();
    cout << "\nLeximi i vlerës së variablës a: ";
    cin >> Dita.a;
    Dita.shtypja();
    return 0;
}
void Jeta::leximi()
{
    cout << "\nLeximi i vlerës së variablës m: ";
    cin >> m;
}
void Jeta::shtypja()
{
    cout << "\nVlera e variablës private m=" << m
    << "\n\nVlera e variablës publike a=" << a << "\n\n";
}
```

Këtu, brenda klasës, si publik është definuar funksioni **leximi**, përmes së cilit lexohet vlera e variablës **m**. Meqë funksioni përfshihet në klasë, ai ka qasje direkte në këtë variabël. Pastaj, ky funksion thirret në fillim të programit kryesor përmes komandës:

Dita.leximi();

Nëse ekzekutohet programi i dhënë dhe përmes tastierës si vlera hyrëse për variablat **m** dhe **a** kompjuterit i jepen vlerat **77** dhe **44.56**, rezultati do të duket si në *Fig.4.3*.



```
Leximi i vlerës së variablës m: 77
Leximi i vlerës së variablës a: 44.56
Vlera e variablës private m=77
Vlera e variablës publike a=44.56
Press any key to continue
```

Fig.4.3

*Pamja e ekranit pas
ekzekutimit të programit
class5a*

INICIALIZIMI GJATË DEKLARIMIT TË OBJEKTEVE

Nëse variablat e përfshira në pjesën me të dhëna të klasës deklarohen vetëm si publike, inicializimi i tyre mund të bëhet edhe gjatë deklarimit të objektit përkatës.

Shembull 5b.: Versioni **class5b** i programit **class5a**, tek i cili variablat **a** dhe **m** janë përfshirë në pjesën publike të klasës **Jeta**, kurse inicializimi i tyre me vlera bëhet gjatë deklarimit të objektit përkatës **Dita**.

```
// Programi class5b
#include <iostream>
using namespace std;
class Jeta
{
    public:
        int m;
        double a;
        void shtypja();
};
int main()
{
    Jeta Dita={77,44.56};
    Dita.shtypja();
    return 0;
}
void Jeta::shtypja()
{
    cout << "\nVlera e variablës private m=" << m
    << "\n\nVlera e variablës publike a=" << a << "\n\n";
}
```

Siç shihet edhe në program, gjatë deklarimit të objektit **Dita**:

```
Jeta Dita={77,44.56};
```

pas barazimit, brenda kllapave janë shënuar vlerat **77** dhe **44.56**, të cilat u përgjigjen variablave **m** dhe **a**. Nëse ekzekutohet programi i dhënë, përmes funksionit **shtypja**, në ekran do të shtypen vlerat e variablave në fjalë, ashtu siç shihet edhe në pjesën e poshtme të *Fig.4.3*.

Variablat që definohen si anëtarë publikë të klasës mund të inicializohen me vlera edhe prej programit kryesor.

Shembull 5c.: Versioni **class5c** i programit **class5b**, tek i cili variablat **a** dhe **m** janë përfshirë në pjesën publike të klasës **Jeta**, kurse inicializimi i tyre bëhet duke i lexuar vlerat përmes komandave përkatëse në programin kryesor.

```
// Programi class5c
#include <iostream>
using namespace std;
class Jeta
{
    public:
        int m;
        double a;
        void shtypja();
};
int main()
{
```

```

Jeta Dita;
cout << "\nLeximi i vlerës së variablës m: ";
cin >> Dita.m;
cout << "\nLeximi i vlerës së variablës a: ";
cin >> Dita.a;
Dita.shtypja();
return 0;
}
void Jeta::shtypja()
{
    cout << "\nVlera e variablës private m=" << m
    << "\nVlera e variablës publike a=" << a << "\n\n";
}

```

Meqë variablat **m** dhe **a** përfshihen në komponentet publike të klasës, lejohet operimi me këto dy variabla në programin kryesor, duke e shfrytëzuar emrin e objektit përkatës dhe pikën si operator për qasje, kështu:

Dita.m

Dita.a

Nëse gjatë ekzekutimit të programit të dhënë më sipër, përmes tastierës kompjuterit i jepen dy vlerat hyrëse **77** dhe **44.56**, në ekran do ta kemi pamjen e dhënë në *Fig.4.3*.

SHFRYTËZIMI I VLERAVE TË VARIABLAVE PRIVATE

Qasja direkte në program të variablat të cilat brenda klasës janë deklaruar si private nuk lejohet. Por, për shfrytëzimin e tyre, brenda klasës duhet të definohen funksione të veçanta, të deklaruara si publike, përmes së cilave mundësohet marrja e vlerave të variablave në fjalë. Gjatë kësaj funksionet mund të jenë *pa parametra formale* ose me *parametra referentë*.

FUNKSIONET PA PARAMETRA FORMALË

Nëse përmes funksioneve merret vlera vetëm e një variable që përfshihet në komponentet private të klasave, atëherë variabla duhet të shënohet te komanda **return** përkatëse, pa shfrytëzuar parametra formale.

Shembull Programi **class5b** si version i programit **class5a**, tek i cili brenda klasës **Jeta** është definuar funksioni **marrja** në një komponente publike, përmes së cilit në program, nga komponentja private merret vlera e variablës **m**.

```

// Programi class5b
#include <iostream>
using namespace std;
class Jeta
{
    private:
        int m;
    public:
        double a;
        void leximi();
        void shtypja();
        int merre();
};
int main()
{
    int k;
    Jeta Dita;
    Dita.leximi();
    cout << "\nLeximi i vlerës së variablës a: ";
    cin >> Dita.a;
    k=Dita.merre();
    cout << "\nVlera e variablës private m=" << k << "\n";
    Dita.shtypja();
}

```

```

return 0;
}
void Jeta::leximi()
{
    cout << "\nLeximi i vlerës së variablës m: ";
    cin >> m;
}
void Jeta::shtypja()
{
    cout << "\nVlera e variablës publike a=" << a << "\n\n";
}
int Jeta::merre()
{
    return m;
}

```

Siç shihet në fund të pjesës së programit ku janë definuar funksionet, brenda funksionit **merre** është vendosur vetëm komanda:

return m;

me çka mundësohet marrja e vlerës së variablës **m**, e cila është deklaruar si private. Funksioni në fjalë është marrë i tipit **int**, sepse vlera e variablës **m** që nxirret si rezultat është e këtij tipi. Ky funksion, në program thirret përmes shprehjes:

k=Dita.merre();

dhe pas kësaj, te variabla **k** përcillet vlera e variablës **m**, e cila me ndërmjetësimin e funksionit **leximi**, përmes tastierës i është dhënë kompjuterit si vlerë hyrëse.

Nëse ekzekutohet programi i dhënë, rezultati në ekran do të duket plotësisht njëjloj me atë që u dha në Fig.4.3.

KONSTRUKTORËT

Me qëllim të inicializimit të variablave të cilat përfshihen në komponentet e klasës, brenda saj definoen funksione të veçanta, të cilat njihen si *konstruktorë*.

Këto funksione ekzekutohen automatikisht, kur deklarohen objektet përkatëse të klasave në fjalë. Konstruktorët dallohen nga funksionet e zakonshme brenda klasës, sepse *kanë emra të njëjtë me klasat* dhe para emrave të tyre si dhe te komandat e fundit **return** nuk shënohet asgjë.

Konstruktorët mund të kenë ose mund të mos kenë parametra formalë.

KONSTRUKTORËT PA PARAMETRA FORMALË

Nëse përcaktimi i vlerave të variablave në komponentet private bëhet brenda konstruktorëve, ato mund të mos kenë parametra formalë.

Shembull 9.: Programi **class9**, përmes së cilit llogaritet sipërfaqja **s** dhe perimetri **p** i rrethit me rreze **r**. Për inicializimin e konstantes **pi**, brenda klasës **rrethi**, e cila definohet në program shfrytëzohet konstruktori përkatës.

```

// Programi class9
#include <iostream>
using namespace std;
class rrethi
{
private:
    double pi,r,s,p;
public:

```

```

rrethi();
void rrezja(double x);
void llogaritja();
void shtypja();
};
int main()
{
rrethi Alfa;
double x;
cout << "\nRrezja e rrethit x: ";
cin >> x;
Alfa.rrezja(x);
Alfa.llogaritja();
Alfa.shtypja();
return 0;
}
rrethi::rrethi()
pi=3.1415926;
}
void rrethi::rrezja(double x)
{
    r=x;
}
void rrethi::llogaritja()
{
    s=pi*r*r;
    p=2*pi*r;
    return;
}
void rrethi::shtypja()
{
    cout << "\nVlerat e llogaritura" << "\n\nSipërfaqja s=" << s
    << "\n\nPerimetri p=" << p << "\n\n";
}

```

Në program, meqë klasa është quajtur me emrin **rrethi**, edhe konstruktori për inicializimin e konstantes **pi**, e cila brenda klasës është deklaruar si private, quhet njëlloj. Në këtë mënyrë, në momentin kur në program deklarohet objekti **Alfa** i klasës **rrethi**:

rrethi Alfa;

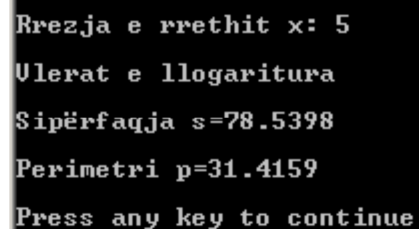
automatikisht ekzekutohet funksioni **rrethi** dhe konstantes **pi** i shoqërohet vlera përkatëse. Pastaj, pasi përmes tastierës kompjuterit t'i jepet vlera e variablës **x**, kur thirret funksioni **rrezja**:

Alfa.rrezja(x);

variablës **r**, e cila e paraqet rrezën e rrethit, i shoqërohet kjo vlerë. Në fund, në program thirren funksionet **llogaritja** dhe **shtypja**, përmes të cilave së pari llogariten sipërfaqja **s** dhe perimetri **p** i rrethit, dhe pastaj vlerat e llogaritura edhe shtypen. Rezultati që fitohet në ekran, për vlerën hyrëse **5** të variablës **x**, do të duket si në *Fig.4.10*.

Fig.4.10

Pamja e ekranit pas ekzekutimit të programit class9



```

Rrezja e rrethit x: 5
Ulerat e llogaritura
Sipërfaqja s=78.5398
Perimetri p=31.4159
Press any key to continue

```


Nuk është e thënë që vlerat e variablave private brenda klasës të jenë vetëm konstante. Ato mund të lexohen si vlera të çfarëdoshme edhe nga jashtë.

Shembull 10.: Programi **class10a**, përmes së cilit llogaritet sipërfaqja **s** dhe perimetri **p** i drejkëndëshit me brinjët **a** dhe **b**. Shoqërimi i vlerave për variablat e brinjëve **a** dhe **b**, brenda klasës **kater** që definohet në program, bëhet përmes konstruktorit përkatës.

```
// Programi class10a
#include <iostream>
using namespace std;
class kater
{
    private:
        double a,b,s,p;
    public:
        kater();
        void llogaritja();
        void shtypja();
};
int main()
{
    kater Omega;
    Omega.llogaritja();
    Omega.shtypja();
    return 0;
}
kater::kater()
{
    cout << "\nVlerat e lexuara" << "\n\nBrinja a: ";
    cin >> a;
    cout << "\nBrinja b: ";
    cin >> b;
}
void kater::llogaritja()
{
    s=a*b;
    p=2*(a+b);
    return;
}
void kater::shtypja()
{
    cout << "\nVlerat e llogaritura" << "\n\nSipërfaqja s=" << s
    << "\n\nPerimetri p=" << p << "\n\n";
}
```

Këtu, brenda konstruktorit **kater** janë përfshirë komandat për leximin e vlerave të variablave **a** dhe **b**, të cilat paraqiten në komponentet private të klasës. Pas deklarimit të objektit **Omega**, janë thirrë dy funksionet e përfshira brendaklasës përkatëse:

Omega.llogaritja();

Omega.shtypja();

ashtu që së pari të llogariten sipërfaqja dhe perimetri i drejkëndëshit dhe pastaj të shtypen ato vlera. Nëse ekzekutohet programi i dhënë dhe për brinjët e drejkëndëshit, përmes tastierës kompjuterit i jepen vlerat **5** dhe **4**, rezultati në ekran do të duket si në Fig.4.11.

Fig.4.11

Pamja e ekranit pas ekzekutimit të programit

class10a

```
Vlerat e lexuara
Brinja a: 5
Brinja b: 4
Vlerat e llogaritura
Sipërfaqja s=20
Perimetri p=18
Press any key to continue
```

KONSTRUKTORËT ME PARAMETRA FORMALË

Përmes konstruktorëve, si edhe përmes funksioneve të zakonshme, variabla brenda klasës mund t'u shoqërohen edhe vlera të çfarëdoshme.

Shembull 10.: Versioni **class10b** i programit **class10a**, tek i cili konstruktori **kater** ka edhe dy parametra formalë **x** dhe **y**.

```
// Programi class10b
#include <iostream>
using namespace std;
class kater
{
    private:
        double a,b,s,p;
    public:
        kater(double x,double y);
        void llogaritja();
        void shtypja();
};
int main()
{
    double x,y;
    cout << "\nVlerat e lexuara" << "\n\nBrinja a: ";
    cin >> x;
    cout << "\nBrinja b: ";
    cin >> y;
    kater Omega(x,y);
    Omega.llogaritja();
    Omega.shtypja();
    return 0;
}
kater::kater(double x,double y)
{
    a=x;
    b=y;
}
void kater::llogaritja()
{
    s=a*b;
    p=2*(a+b);
    return;
}
void kater::shtypja()
{
    cout << "\nVlerat e llogaritura" << "\n\nSipërfaqja s=" << s
    << "\n\nPerimetri p=" << p << "\n\n";
}
```

Këtu, parametrat formalë të konstruktorit **kater** shfrytëzohen për përcjelljen brenda tij të vlerave **x** dhe **y**, të cilat lexohen në programin kryesor.

Pastaj, këto vlera u shoqërohen variabla **a** dhe **b** të klasës, përmes shprehjeve:

a=x;

b=y;

të cilat përfshihen te konstruktori. Meqë në momentin e deklarimit të objektit të klasës **kater** automatikisht thirret edhe konstruktori, deklarimi i objektit duhet të shoqërohet edhe me vlerat aktuale të këtyre dy parametrave. Për këtë arsye, deklarimi i objektit **Omega** i klasës **kater** është bërë në formën:

kater Omega(x,y);

ku variablat **x** dhe **y** paraqesin parametra aktualë, vlerat e të cilëve janë lexuar paraprakisht. Rezultati që fitohet në ekran, për vlerat hyrëse **5** dhe **4** të brinjëve **a** dhe **b**, do të duket si në *Fig.4.11*.

Në literaturë takohet edhe një formë tjetër e definimit të konstruktorëve. Kjo formë, p.sh., për konstruktorin i cili u shfrytëzua te programi i mësipërm mund të duket:

```
kater::kater(double x,double y):a(x),b(y)
{
}
```

Edhe pas definimit të konstruktorit në këtë mënyrë, deklarimi i objektit **Omega** të klasës **kater** në program nuk do të ndryshojë.

Gjatë përcaktimit të vlerave duke e shfrytëzuar këtë formë të shkruarjes së konstruktorëve, brenda kllapave të variablave mund të shkruhen vlera numerike konkrete.

Shembull 11.: Programi **class11**, përmes së cilit llogaritet vlera e funksionit: $y = ax + (m + 1)!$

nëse variablat **m** dhe **x** përfshihen në komponentet private të klasës **Alfa**, kurse variabla **a** merret si variabël e jashtme.

```
// Programi class11
#include <iostream>
using namespace std;
class Alfa
{
    private:
        int m;
        double x;
    public:
        Alfa();
        double Funk(double a);
};
int main()
{
    double y;
    Alfa Dita;
    y=Dita.Funk(6.5);
    cout << "\nVlera e llogaritur y=" << y << "\n\n";
    return 0;
}
Alfa::Alfa():m(3),x(5)
{
    cout << "\nVlera e variablës m: " << m << "\n\nVlera e variablës x:" << x << "\n";
}
double Alfa::Funk(double a)
{
    double F=1,y;
    int i;
    for (i=1;i<=(m+1);i++)
        F=F*i;
    y=a*x+F;
    return y;
}
```

Në këtë rast, përmes konstruktorit janë inicializuar direkt variablat **m** dhe **x** me vlerat 3 dhe 5 - përkatësisht, duke i shënuar:

Alfa::Alfa(): m(3),x(5) kurse gjatë thirrjes së funksionit **Funk**:

y=Dita.Funk(6.5); ku : parametri formal **a** është zëvendësuar me vlerën aktuale 6.5. Rezultati në ekran pas ekzekutimit të programit të dhënë do të duket si në *Fig.4.12*.

Fig.4.12

Pamja e ekranit pas ekzekutimit të programit

class11

```
Vlera e variablës m: 3
Vlera e variablës x: 5
Vlera e llogaritur y=56.5
Press any key to continue
```

STRUKTURË ZGJEDHJE

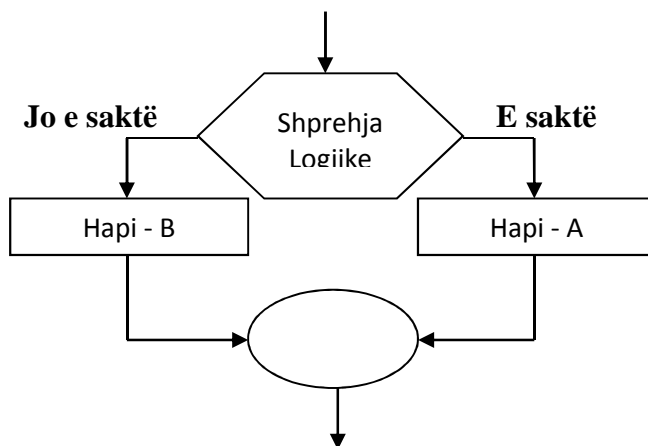
Struktura zgjedhje shfrytëzohet për zgjedhjen e njërës nga drejtimet e mundëshme për vazhdimin e veprimit të algoritmit mvarësisht nga ndonjë kusht. Kushti në të shumtën e rasteve është ndonjë shprehje që mund të fitojë vlera të ndryshme, pra nga vlera e shprehjes kemi drejtimin e vazhdimin të algoritmit.

Kemi dy tipe të strukturave zgjedhje dhe atë:

- Zgjedhje nga dy mundësi
- Zgjedhje nga më shumë mundësi

Zgjedhja nga dy mundësi

Në këtë strukturë kemi mundësi të zgjedhim një nga dy mundësitë mvarësisht nga vlerat e shprehjes. Shprehja mund të ketë dy vlera dhe atë e saktë dhe jo e saktë. Këto vlera quhen vlera logjike ndërsa shprehja shprehje logjike .

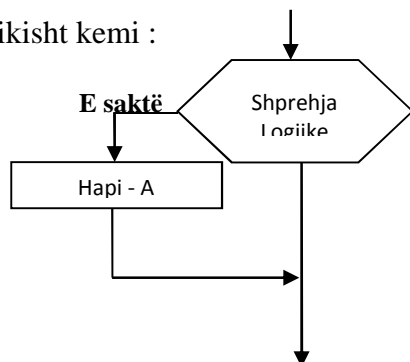


Nëse vlera e shprehjes logjike është e saktë atëherë do të kryhet hapi algoritmit A, ndërsa nëse vlera e shprehjes logjike është jo e saktë atëher do të kryhet hapi algoritmik B. Ndryshe kjo strukturë quhet edhe si nese - përndryshe (angl if - else).

Nëse Hapi-A dhe Hapi B janë struktura lineare atëher ato fillojnë me fjalën fillim dhe mbarojnë me fjalën fund.

Struktura zgjedhje mund të shfrytëzohet edhe në rastet kur njëra nga mundësit nuk ka hap për kryerje, atëher kemi mundësinë për kryerjen e ndonjë hapi ose mos kryerjen e atij hapi algoritmik.

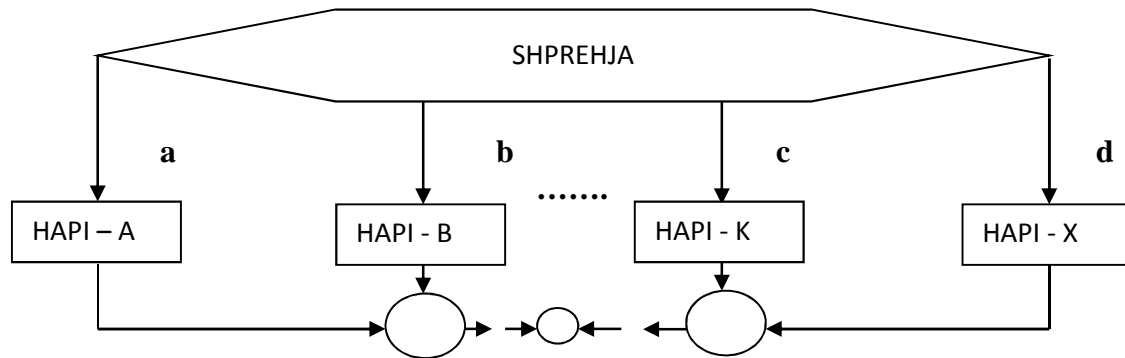
Pra grafikisht kemi :



ZGJEDHJE NGA MË SHUMË MUNDËSI

Nëse kemi rastin ku mundemi të zgjedhim nga më shumë mundësi për vazhdimin e algoritmit, atëherë shfrytëzohet strukturë zgjedhje nga më shumë mundësi. Zgjedhja kryhet mvarësisht nga vlera e ndonjë të dhëneje ose ndonjë shprehjeje, si p.sh nëse kemi përshkrimin tekstualisht të suksesit të ndonjë nxënësi atëherë e dhëna mund të ketë vlerat: 5, 4, 3, 2, 1 .

Grafikisht këto struktura paraqiten:



Vlera e shprehjes mund të jetë njëra nga vlerat A, B,K, ose mund të mos jetë asnjëra nga ato. Nëse vlera është a atëherë do të vazhdojë me hapin algoritmik A, nëse vlera është b atëherë do të vazhdojmë me hapin algoritmik B dhe ashtu me radhë, por nëse vlera e shprehjes nuk është asnjëra nga këto vlera pra prej a deri m k, atëherë do të kryhet hapi algoritmik X. Por kemi edhe rastet kur nuk është paraparë ndonjë hap algoritmik nëse vlera e shprehjes nuk është asnjëra nga ato vlera të parapara në algoritëm.

Nga e përmendura më lartë mund të themi se në këto struktura kemi zgjedhjen nga më shumë mundësi. Për këtë shkak edhe thirren si struktura ZGJEDHJE NË RAST, ose shkrutimisht RAST (angl. CASE)

Vlerat të cilat mund ti ketë shprehja duhet të jenë të tipit linear.

Shembull për përshkrimin e suksesit të nxënësve :

algoritëm Sukses;

fillim

lexo n;

rasti n

5 : shtyp"shkëlqyeshëm".

4 : shtyp"shumë mirë".

3 : shtyp"mirë".

2 : shtyp"mjafueshëm".

1 : shtyp"dobë".

përndyshe

shtyp"gabim".

fund_rast(n);

fund.

OPERATORET E KRAHASIMIT

Operatorët e krahasimit kryejnë krahasimin e vlerave të dy variablave dhe përdoren si në rastet e mëposhtme

Operatori	Domethënia	Shembull	Rezultati
<	Më i vogël se	2<3	e vërtetë
<=	Më i vogël se ose barazi me	3<=2	jo e vërtetë
==	Barazi me	3==	e vërtetë
>	Më i madhë se	3>2	e vërtetë
>=	Më i madhë se, ose barazi me	2>=3	jo e vërtetë
!=	Jobarazi me	2!=3	e vërtetë

Operatorët logjikë

Për krahasimin e më shumë shprehjeve njëkohësisht përdoren operatorët logjik të dhënë në tabelën e mëposhtme

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

REALIZIMI I STRUKTURAVE PËR ZGJEDHJE

Strukturat për zgjedhje realizohen me anë të urdhëresave për zgjedhje. Kemi tre lloje të urdhëresave për zgjedhje :

1. **if**
2. **goto**
3. **switch**

Urdhëresa IF në gjuhën programore C++ shfrytëzohet në dy forma dhe atë :

Forma e parë është :

if (kushti) urdhëresa;

Me këtë formë realizohet struktura zgjedhje:

Nëse në momentin e kryerjes kushti ka vlerë të saktë (është i plotësuar), atëherë do të kryhet urdhëresa përkatëse nëse në momentin e kryerjes kushti nuk ka vlerë të saktë (nuk është plotësuar) atëherë nuk do të kryhet urdhëresa përkatëse por program do të vazhdojë me rreshtat e ardhshëm programor pa e kryer atë urdhëresë programore.

Shembulli 1 : Me urdhëresën **if (A>5) C=2*A+1;**

do të njehsohet vlera e ndryshores C nëse vlera e ndryshores A është më e madhe se 5, përndryshe nuk do të njehsohet vlera e C.

Shembulli 2 : Me urdhëresat

```
if(X<0)
X=-(X)
Y=-sqrt(B);
```

në këtë shembull kemi njehsimin e rrënjës katrore të X-it nëse vlera e X-it është < 0 atëherë vlera e X-it do të bëhet pozitive dhe mandej do të njehsohet rrënja katrore e saj pra rreshti i ardhshëm programor, përndryshe nuk do të kryhet asnjë veprim por vetëm do të njehsohet vlera e rrënjës katrore e X-it.

Shembulli 3 : **if (a>b && c<0)**
y=sqrt(x);

pra për tu plotësuar kushti në këtë rast duhet të jetë edhe $a>b$ dhe $c<0$ pra të dy këto kushte duhet të jenë plotësuar për tu plotësuar kushti i urdhëresës if që të kryhet urdhëresa $y=sqrt(x)$.

Shembulli 4: **if (a>b || c<0)**

```
y=sqrt(x);
```

në këtë rast kemi plotësimin e njërit nga kushtet si kusht i domosdoshëm për plotësimin e kushtit global për kryerjen ose jo të urdhëresës $y=sqrt(x)$;

Në fillim u përmend se kjo urdhëresë (if) në gjuhën programore C++ shfrytëzohet në dy forma, formën e parë e pamë ndërsa forma e dytë është:

```
if(kushti)
urdhëresa-1;
else
urdhëresa-2;
```

Nëse në momentin e kryerjes kushti ka vlerë të saktë (është i plotësuar), atëherë do të kryhet urdhëresa-1 ndërsa nuk do të kryhet urdhëresa-2 dhe mandej program vazhdon me urdhëresat në rreshtat e ardhshëm programor, përndryshe nëse në momentin e kryerjes së urdhëresës kushti nuk ka vlerë të saktë (nuk është i plotësuar), atëherë nuk do të kryhet urdhëresa-1 por do të kryhet urdhëresa pas fjalës else (urdhëresa-2) dhe mandej programi do të vazhdojë me rreshtat e ardhshëm programor. Pra kemi rast kur kryhet njëra ose tjetra urdhëresë. Pra për dallim nga forma e parë ku kishim rastin e kryerjes ose jo të ndonjë urdhërese mvarësisht nga kushti, këtu kemi kryerjen e njëres urdhëresë ose tjetres mvarësisht nga kushti. Te kjo formë mund të kryhen më shumë urdhëresa . Në këtë rast urdhëresa do të duket kështu

```
if ( kushti )
{
urdhëresa-1;
urdhëresa-2;
}
else
{
urdhëresa-3;
urdhëresa-4;
}
```

Shembulli 5: Me urdhëresën **if(d>3)**

```
{
t=x*x;
}
else
{
```

```
t=x*x*x;
```

```
}
```

Shembulli 6 : Me urdhëresën

```
if (J<=65)
{
    Y=A*A+1;
    Z=E*E-18;
}
else
{
    X=A*A+1;
    Y=A*A*A+101;
    Z=E*E-4;
}
```

kemi kryerjen e bllokut të urdhëresave: $Y=A*A+1$;

$Z=E*E-18$;

nëse vlera e J është më e vogël ose e barabart me 65, përndryshe vlera e ndryshores J nëse nuk është më e vogël se 65 atëherë do të kryhet blloku i urdhëresave: $X=A*A+1$;

$Y=A*A*A+101$;

$Z=E*E-4$;

Shembuji dhe detyra

1. Të shkruhet program i cili për numrin pozitiv (numri>0) të futur nga tastiera paraqitet mesazhi “Numri qe lexuat eshte pozitiv”

Zgjidhja

```
#include <iostream>
using namespace std;
int main( )
{
    int numri;
    cout<<"lexo nje numer nga tastiera"<<endl;
    cin>>numri;
    if(numri>0)
    {
        cout<<"numri qe lexuat eshte pozitiv";
    }
    return 0;
}
```

2. Të shkruhet program i cili lexon një numër nga tastiera, mvarësisht prej numrit që lexohet të shtyp mesazhin Vlera e numrit x eshte pozitiv ose Vlera e numrit x eshte negativ ose e numrit x eshte e barabarte me zero dhe Vlera e numrit x eshte e barabarte me(me vleren e numrit)

Zgjidhje

```
#include <iostream>
using namespace std;
int main()
{
    int x;
    cout << "Vlera e dhene: x=";
    cin >> x;
```



```

cout << "\n";
if (x > 0)
    cout << "Vlera e numrit x eshte pozitiv! \n\n";
if (x < 0)
    cout << "Vlera e numrit x eshte negativ! \n\n";
if (x == 0)
    cout << "Vlera e numrit x eshte e barabarte me zero! \n\n";
if (x)
    cout << "Vlera e numrit x eshte e barabarte me " << x << "! \n\n";
return 0;
}

```

3. Të shkruhet program i cili mvarësisht nga numri i futur nga tastiera shtyp mesazhin numri që futet nga tastiera është numër çift ose numri që futet nga tastiera është numër tek
4. Të shkruhet program i cili gjen numrin më të madh nga dy numra të futur nga tastier.
5. Të shkruhet program i cili gjen numrin më të madh nga tre numra të futur nga tastier.
6. Të shkruhet program i cili pasi të lexohen tre numra e bën vërtetimin se a mundet ato tre numra të jenë brinjët e një trekëndëshi $(a+b>c)\&\&(b+c>a)\&\&(a+c>b)$

```

#include <iostream>
using namespace std;
int main ()
{
    int a,b,c;
    cout<<"Jepni numrin e pare\t";
    cin>>a;
    cout<<"jepni numrin e dyte\t";
    cin>>b;
    cout<<"jepni numrin e trete\t";
    cin>>c;
    if((a+b>c)&&(b+c>a)&&(a+c>b))
        cout<<"trekendeshi i atille ekziston\n";
    else
        cout<<"\ntrekendeshi i atille nuk egziston\n";
    return 0;
}

```

DEGËZIMET E SHUMËFISHTA - SWITCH()

Programet tek të cilat na paraqiten me shumë degëzime, thjeshtohen nëse degëzimi realizohet përmes komandës switch, forma e përgjithshme e së cilës është:

```
switch ( test )
```

```

{
    case 1 : test1; // ekzekutohet komanda test = 1 ...
break;
    case 5 : test5; // ekzekutohet komanda test = 5 ...
break;
    default : // te gjitha cases tjera....
}

```

//Shembull nga perdorimi i switch()

```

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

    cout<<"Jepni noten 1, 2, 3, 4 ose 5 : Nota=" ;
    cin >> Nota;
    switch(Nota)
    {
        case 1: cout<<"Dobet: ";
                break;
        case 2: cout<<"Mjaftueshem: ";
                break;
        case 3: cout<<"Mire: ";
                break;
        case 4: cout<<"Shume mire: ";
                break;
        case 5: cout<<"Shkelqyeshem: ";
                break;
        default: cout <<"Gabim nota e dhene!\n";
    }

    cout<<" Nota=" << Nota<< "\n";
    return 0;}

```

Shembuji dhe detyra

1. Të shkruhet program i cili na mundëson të zgjedhim një ditë nga java me numrat 1...7 ashtuqë kur të zgjedhim 1 për ditën e Hënë të shtypet orari i ditë së Hënë e kështu me radh. Kurse për çdo numër tjetër të lexuar nga tastiera të shtypet porosia se për ditë të tillë orari nuk egziston, apo dita e zgjedhur është ditë pushimi .

Zgjidhja:

```

#include <iostream>
using namespace std;
int main ()
{
    int dita;
    cout<<"Orari\n";
    cout<<"1 per ditën e hene\n";
    cout<<"2 per ditën e marte\n";
    cout<<"3 per ditën e merkure\n";
    cout<<"4 per ditën e enjte\n";
    cout<<"5 per ditën e premte\n";
}

```

```

cout<<"Fut numrin e dites qe don te shohish orarin\n";
cin>>dita;
switch(dita)
{
case 1: cout<<"matem, histo, gjeogra\n";
        break;
case 2: cout<<"fiz, kimi, kujd\n";
        break;
case 3: cout<<"angl, kimi, gjeogra\n";
        break;
case 4: cout<<"shqip, gjerm, bio\n";
        break;
case 5: cout<<"prog, hist, hist\n";
default:
        cout<<"dita e zgjedhur eshte dite pushimi\n";
}

return 0;
}

```

2. Të shkruhet program i cili na mundëson të zgjedhim operacionin matematikor që do të kryhet mbi numra të lexuara nga tastiera p.sh + për mbledhjen – për zbritjen e kështu me radhë.

```

#include <iostream>
using namespace std;
int main ()
{
char operatori;
float a;
float b;
cout<<"Lexoni dy numra a dhe b\n";
cin>>a>>b;
cout<<"shtyp + per te mbledh numrat e mesiperme\n";
cout<<"shtyp - per te zbrit dy numrat e mesiperme\n";
cout<<"shtyp * per te shumezuar dy numrat e mesiperme\n";
cout<<"shtyp / per te pjestuar dy numrat e mesiperme\n";
cin>>operatori;
switch(operatori)
{
case '+': cout<<a+b;
          break;
case '-': cout<<a-b;
          break;
case '*': cout<<a*b;
          break;
case '/': cout<<a/b;
          break;

default:
        cout<<"Operacioni qe shenuat nuk ekziston\n";
}

return 0;
}

```

KAPËRCIMI PA KUSHT GOTO

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

- Komanda goto

goto a;

- Ku (a:) është labela e zgjedhur lirisht si identifikatorë

- Shembuj labelash:

- fillimi:
- fundi:
- perserite:
- klasa:

// Programi i pare me goto

```
#include <iostream>
using namespace std;
int main()
{
    goto Fundi;
Fillimi:                //Labela - etiketa,adresa "Fillimi"
    cout<< "Fillimi\n";
    goto Dalja;
Mesi:                  //Labela - etiketa,adresa "Mesi"
    cout << "Mesi\n";
    goto Fillimi;
Fund:                  //Labela - etiketa,adresa "Fund"
    cout << "Fund\n";
    goto Mesi;
Dalja:                 //Labela - etiketa,adresa "Dalja"
    return 0;
}
```

STRUKTURË PËRSËRITJE

Këto struktura shfrytëzohen atëher kur është e nevojshme që një grup i hapave të kryhen më shumë herë. Përsëritje e tillë e një grupi të hapave algoritmik quhet përsëritje ciklike. Një kryerje e hapave quhet cikël.

Struktura e përsëritjes i lejon programuesit të realizojë përsëritjen e një aksioni përsa kohë që një kusht plotësohet.

```
shembull:    cikël
              hapi A;
              hapi B;
              ...
              hapi M;
fund_cikli
```

Nëse marrim rastin për njehsimin e shumës së 10 numrave të parë natyror pra $S=1+2+3+4+5+6+7+8+9+10$ do të kemi : Shuma në fillim është '0' dhe ja shtojmë numrin e parë '1' pra $0+1=1$. Numri i dytë është numri '2' të cilin poashtu e shtojmë në shumën, mandej numrin '3', '4' dhe ashtu me radhë deri te numri '10'. Nga kjo vërehet se e gjithë kjo punë përbëhet nga dy veprime dhe atë :

- shtohet numri në shumën dhe
- zmadhohet numri që shtohet për '1' (që të fitohet numri i ardhshëm natyror)

Tekstualisht mund ta paraqesim si :

```

shuma=0;
numri=1;
cikli
    -shto numri në shuma;
    -zmadho numri për 1;
fund_cikli

```

Hapat e ciklit do të përsëriten deri sat ë shtohet numri '10'. Pra kushti është numri të merr vlerë '10'. Për të ndërprerë përsëritjen e ciklit duhet kontrolluar kushtin. Kjo kontrollë mund të bëhet në fillim të ciklit ose në fund të ciklit dhe mënyra e tretë është që paraprakisht të përcaktohet numri i përsëritjeve të ciklit.

Pra kemi tre lloje të strukturave përsëritje dhe atë përsëritje me dalje në fillim të ciklit, përsëritje me dalje në fund të ciklit, përsëritje duke numëruar ciklat.

PËRSËRITJE ME DALJE NË FILLIM TË CIKLIT

Te këto struktura kemi kontrollimin prej në fillim pra para se të fillohet me ciklat kemi kontrollën e kushtit pra mund të ndodh që prej fillimi të mos jet kushti i plotësuar pra të mos kemi kryerjen e ciklit prej fillimi. Ndryshe këto struktura quhen edhe si DERISA kryej.(ëhile)

shembull:

```

derisa kushti
    urdhëresa1;
    urdhëresa2;

```

.....

fund_derisa

PËRSËRITJE ME DALJE NË FUND TË CIKLIT

Te këto struktura kemi kontrollimin e kushtit në fund të ciklit pra të përsëritet cikli ose jo.

Këtu pasi që kushti kontrollohet në fund të ciklit pra cikli patjetër kryhet një herë dhe pas kontrollës mund të mos përsëritet më. Nëse kushti është i plotësuar cikli do të përsëritet përndryshe nëse kushti nuk është i plotësuar më nuk do të përsëritet. Ndryshe këto struktura quhen si përsërit derisa (do while)

shembull

```

do
    hapi A;
    hapi B;

```

.....

derisa kushti

fund_përsëritje

PËRSËRITJE DUKE NUMËRUAR CIKLAT

Në dy strukturat e lart përmendura ciklat përsëriteshin ose jo mvarësisht nga ndonjë kusht dhe nuk dihet se sa herë do të përsëriten ciklat, te këto struktura kemi numrin e përsëritjeve paraprakisht të ditur. Numri i përsëritjeve numërohen me numërues të vecant për të cilin jepen vlera fillestare, vlera e fundit si dhe hapi (intervali) me të cilin ndryshohen vlerat duke filluar nga vlera fillestare deri tek vlera e fundit. Cikli përsëritet për çdo vlerë të numëruesit prej fillimi deri në fund duke pasur parasysh hapin. Këto struktura ndryshe quhen si për-deri-hapi)

për (numëruesi= fillimi; kushti; numeruesi=numeruesi+hapi)

hapi A;
hapi B;

hapi J;
fund_për

REALIZIMI I STRUKTURAVE CIKLIKE

Strukturat ciklike realizohen me anë të urdhëresave për përsëritje. Kemi tre lloje të urdhëresave për përsëritje :

1. while
2. do while
3. for

URDHËRESA WHILE

Forma e përgjithshme e urdhëresës while është:

while(kushti) urdhëri

sh. Te shkruhet program i cili pasi të lexohet një numër nga tastiera të shtyp numrat nga numri i futur gjer në 1 (p.sh nëse lexojmë nga tastiera numrin 5 të shtyp numrat 5, 4, 3, 2, 1)

```
#include <iostream>
using namespace std;
int main ()
{
    int n;
    n=5;

    while (n>0)
    {
        cout << n;
        n=n-1;
    }
    return 0;
}
```

sh. Të shkruhet program që llogarit shumën e 10 numrave të parë natyror

```
#include <iostream>
using namespace std;
int main ()
{
    int shuma,numri;
    shuma=0;
    numri=1;
    while(numri<=10)
    {
        shuma=shuma+numri;
        numri=numri+1;
    }
    cout<<shuma;
    return 0;
}
```

URDHËRESA DO WHILE

Forma e përgjithshme e urdhëresës do while është:

```
do { urdhëri }  
while (kushti);
```

sh. Program i cili njehson shumën e 10 numrave të parë natyror

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

int main ()
{
    int shuma,numri;
    shuma=0;
    numri=1;
    do
    {
        shuma=shuma+numri;
        numri=numri+1;
        cout<<shuma<<" , ";
    }

    While(numri<=10);
    return 0;
}
```

URDHËRESA FOR

Forma e përgjithshme e urdhëresës do for është:

```
for (numëruesi=fillimi; kushti; i=i+hapi)
```

sh. Program që do të bën shtypjen e numrave nga 0 gjer në 10 me hap 2 (0, 2, 4, 6, 8, 10)

```
#include <iostream>
using namespace std;
int main()
{
    int numero;
    for(numero=0; numero<=10; numero=numero+2)
        cout<<numero<<" , ";
    return 0;
}
```

sh. Program që bën llogaritjen e shume se 10 numrave te pare natyror

```
#include <iostream>
using namespace std;
int main()
{
    int numri;
    int shuma;
    shuma=0;
    for(numri=1; numri<=10; numri=numri+1)
        shuma=shuma+numri;
}
```

```
        cout<<"shuma e numrave natyror nga numri 0 gjer ne 10  
eshte"<<shuma<<"\n ";  
        return 0;  
}
```

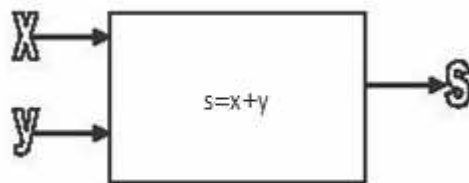
1. Program qe shtyp numrat nga numri 1 gjer ne 100

```
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int count;  
  
    for(count=1; count <= 100; count=count+1)  
        cout << count << " ";  
  
    return 0;  
}
```


FUNKSIONET

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

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



Forma e përgjithshme e funksionit:

```

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

ku janë:

tipi - tipi i rezultatit të funksionit.

emri - emri i funksionit.

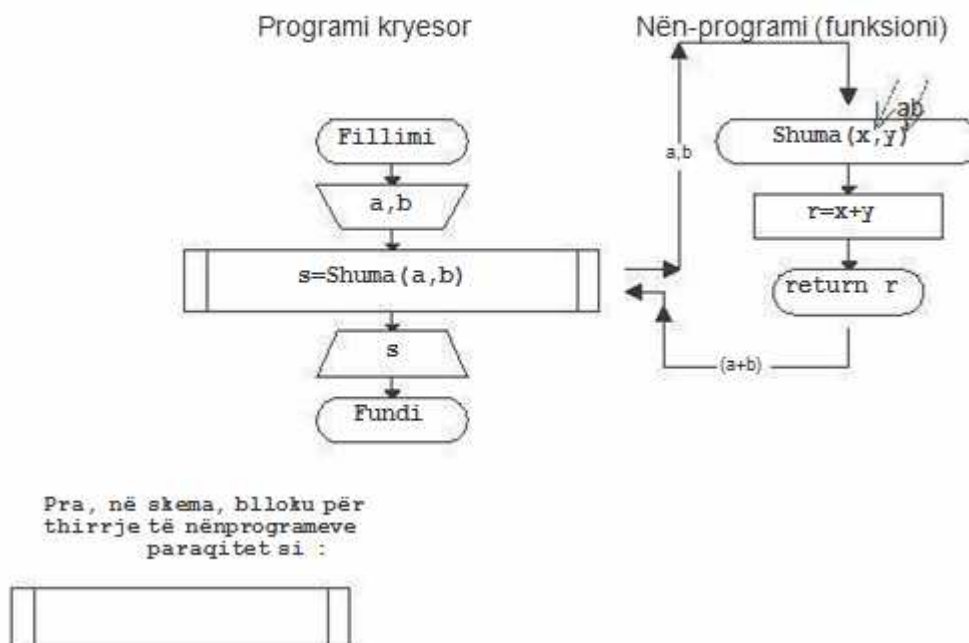
tipi1,... tipin - tipet e parametrave formal.

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

urdhëri/at - urdhëri/at që ekezekutohen brenda funksionit

r - rezultati të cilin e kthen funksioni.

Bloku skema e programit më nënprogram duket si në vijim:



Programi kryesor e thërret funksionin (nënprogramim) për të kryer një llogaritje. Funksioni e kryen llogaritjen duke i përdorur parametrat aktual të pranuar prej programit kryesor dhe në renditjen e definuar duke i zëvendësuar ata në vend të parametrevë të tij formal. Në fund, funksioni, vlerën e llogarituar ia kthen programit kryesor me urdhërin “return” dhe programi kryesor vazhdon aty ku e kishte ndërprerë “punën” për të pritur rezultatin e kërkuar prej nënprogramit.

```

// Programi Funksionil
#include <iostream>
using namespace std;
double Shuma(int x,int y);    //Prototipi i funksionit
int main()
{
double s;
int a=7,b=3;
s=Shuma(a,b)    /*Thirrja e funksionit - percjellja e vlerave aktuale*/
cout << "Shuma s=" << s << "\n";
return 0;
}

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

Pra, kur të thirret funksioni shuma, me urdhërin: **s=Shuma(a,b);**, atëherë programi kryesor e percjellë ekzekutimin në funksionin e thirrur, duke ia percjellë vlerat aktuale a=7 dhe b=3, për parametrat formal x dhe y. Në brendi të funksionit, llogaritet variabla e shumës **r = x+y**, e cila në rasing aktual llogaritet **r=a+b**, sepse në hyrjen **x** i ka ardhë vlera **a**, kurse në hyrjen **y** i ka ardhë vlera **b**. Këtë variabël (shumën “r”) funksioni ia kthen programit përmes urdhërit: **return r;**

Kështu, del se urdhëri i programit **s=Shuma(a,b);** në fakt është: **s=r;** , ku r - është rezultati i kthyer prej funksionit, me urdhërin: **return r;**.

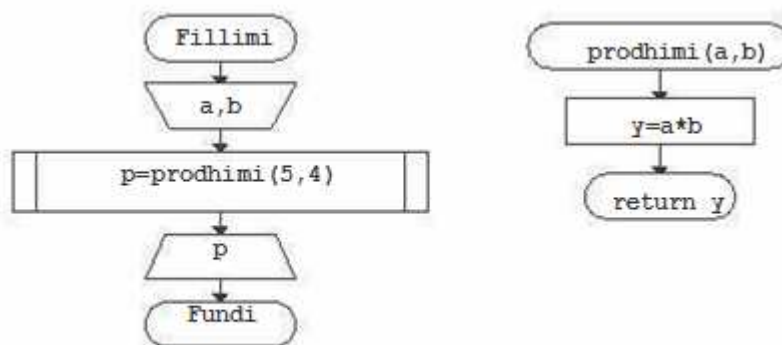
FUNKSIONET VOID

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

INLINE FUNKSIONET

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

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



Shembuj funksionesh

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

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

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

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

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

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

// Programi Funksion1

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

```
double ShumaVargut(int a, int b, int c, int d);
```

```
int main()
```

```
{
```

```
    int m,n; double
```

```
    Shuma; m=0;
```

```
    n=5;
```

```
    Shuma=ShumaVargut(m,n,1,0); //Funks. per shumen e vargut
```

```
    cout << "Shuma S=: " << Shuma;
```

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

```
return 0;
```

```
}
```

```
double ShumaVargut(int a,int b, int c, int d)
```

```
{
```

```
    int i,j;
```

```
    double S;
```

```
    S=0;
```

```
    for (i=a;i<=b;i++)
```

```
    {
```

```
        S=S+(c*i+d);
```

```
    }
```

```
    return S;
```

```
}
```

Te llogaritet:

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

// Programi Funksion2

```
#include <iostream>
```

```
using namespace std;
```

```
double ShumaVargut(int a, int b, int c, int d);
```

```
int main()
```

```
{
```

```
    int n;
```

```
    double Shuma;
```

```
    cout<<"Jepe vleren per n:";
```

```
    cin>>n;
```

```
    Shuma=ShumaVargut(2,n,3,2); //Funks. per shumen e vargut
```

```
    cout << "Shuma S=: " << Shuma;
```

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

```

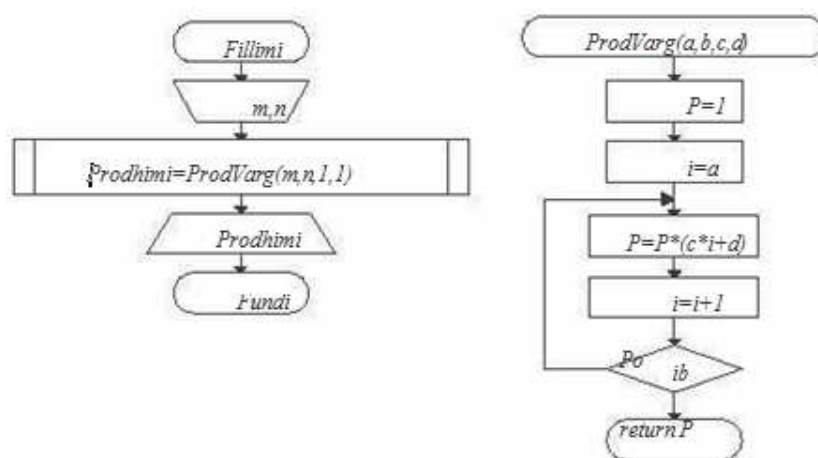
    return 0;
}
double ShumaVargut(int a,int b, int c, int d)
{
    int i,j;
    double S;
    S=0;
    for (i=a;i<=b;i++)
    {
        S=S+(c*i+d);
    }
    return S;
}

```

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

(Në këte rast do të kemi: $a=m$, $b=n$, $c=1$, $d=1$)

$$P = \prod_{i=m}^n (c*i + d)$$



```

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

double ProdVarg(int a, int b, int c, int d);
int main()
{
    int m,n;
    double Prodhimi;
    m=0;
    n=5;
    Prodhimi=ProdVarg(m,n,1,1); //Funks. per prod. e vargut
    cout << "Prodhimi P=: " << Prodhimi;
    cout << "\n";
    return 0;
}
double ProdVarg(int a,int b, int c, int d)
{
    int i,j;
    double P;
    P=1;
    for (i=a;i<=b;i++)

```

```

    {
        P=P*(c*i+d);
    }
    return P;
}

```

Shembull :

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

// Programi Funksion3

```

#include <iostream>
using namespace std;

```

```

double ProdVarg(int a, int b, int c, int d);

```

```

int main()
{

```

```

    int m,n;

```

```

    double Prodhimi;

```

```

    cout<<"Jepni vlerat per kufijte m dhe n:";

```

```

    cin>>m>>n;

```

```

    Prodhimi=ProdVarg(m,n,2,4);

```

//Funks. per prod. e vargut

```

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

```

```

    cout << "\n";

```

```

    return 0;
}

```

```

double ProdVarg(int a,int b, int c, int d)
{

```

```

    int i,j;

```

```

    double P;

```

```

    P=1;

```

```

    for (i=a;i<=b;i++)
    {

```

```

        P=P*(c*i+d);
    }

```

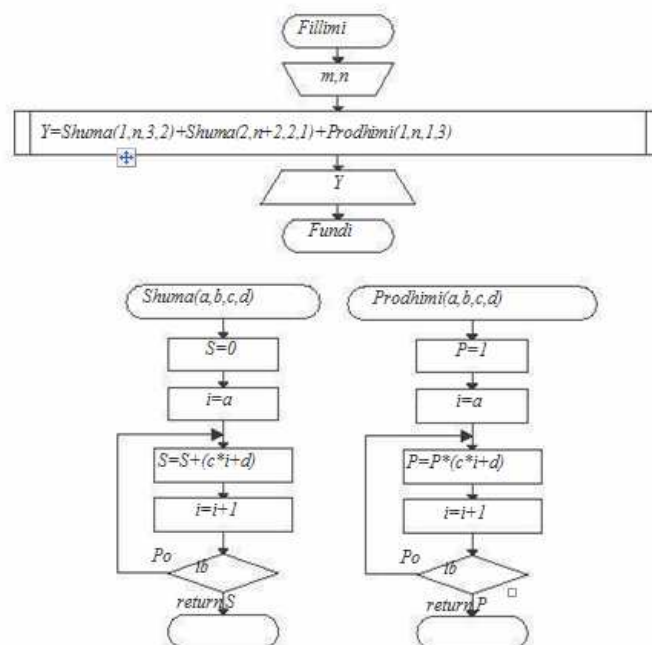
```

    return P;
}

```

të llogaritet:

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



```

// Programi Funksion5
#include <iostream>
using namespace std;

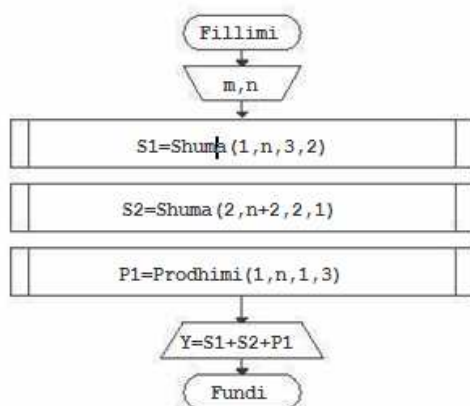
double Shuma(int a, int b, int c, int d);
double Prodhimi(int a, int b, int c, int d);
int main()
{
    int n;
    double Y;
    cout<<"Jepe vleren per n:";
    cin>>n;
    Y=3*Shuma(1,n,3,2)+Shuma(2,n+2,2,1)+Prodhimi(1,n,1,3);
    cout << "Shuma Y= " << Y;
    cout << "\n";
    return 0;
}

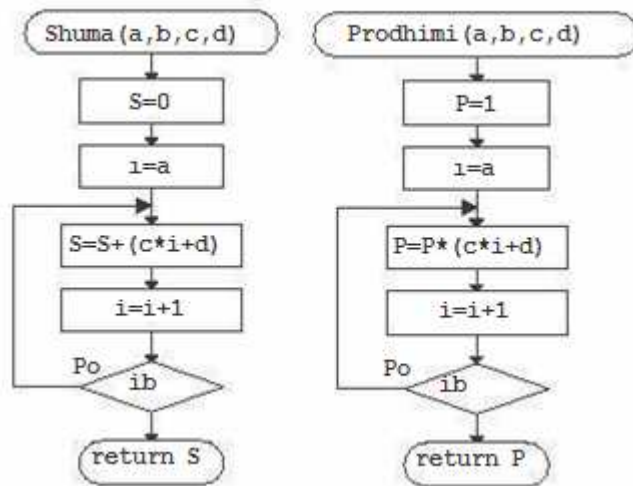
double Shuma(int a,int b, int c, int d)
{
    int i;
    double S;
    S=0;
    for (i=a;i<=b;i++)
    {
        S=S+(c*i+d);
    }
    return S;
}

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

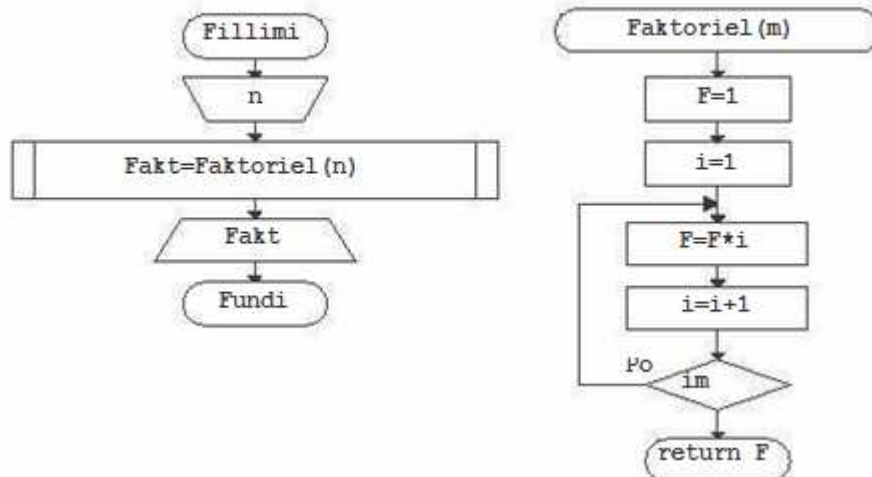
```

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





Llogaritja e faktorelit



```
// Programi Funksion4
```

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

```
double Faktoriel(int m);
```

```
int main()
{
```

```
    int n;
```

```
    double Fakt;
```

```
    n=5;
```

```
    Fakt=Faktoriel(n);
```

```
    //Funks. per Faktoriel
```

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

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

```
    return 0;
```

```
}
```

```
/* Funksioni per llogaritje te faktorielit */
```

```
double Faktoriel(int m){
```

```
    int i;
```

```
    double F;
```



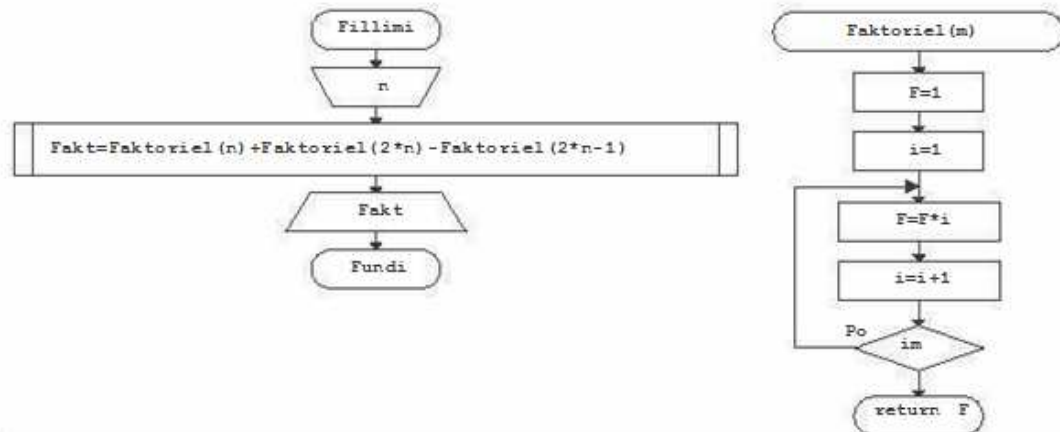
```

F=1;
for (i=1;i<=m;i++)
{
    F=F*i;
}
return F;
}

```

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

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



```

// Programi Funksion4
// Fakt = n! + (2n)! - (2n-1)!
#include <iostream>
using namespace std;

double Faktoriel(int m);
int main()
{
    int n;
    double Fakt;
    n=2;
    Fakt=Faktoriel(n)+Faktoriel(2*n)-Faktoriel(2*n-1);
    cout << "Faktorieli F = " << Fakt;
    cout << "\n";
    return 0;
}

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

```

REKURZIONI

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

```
// Programi Faktoriel-Rekurzioni
#include <iostream>
using namespace std;
double Faktoriel(int m);
int main()
{
    int n;
    double Fakt;
    n=5;
    Fakt=Faktoriel(n);           //Funks. per Faktoriel
    cout << "Faktorieli F = "<<n<<"! = " << Fakt;
    cout << "\n";
    return 0;
}

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

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

```
//Llogaritja e faktorielit, permes rekurzionit
#include <iostream>
using namespace std;
int main()
{
    int n;
    int fakt;
    cout << "Jepni vleren per n: ";
    cin >> n;
    fakt = Faktoriel (n);
    cout << n << "! = " << fakt << endl;
    return 0;
}

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

Funksionet dhe Vektorët

// Programi FunksioniVektor1

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

// Programi FunksionVektor2

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

// Programi FunksionVektor3 - Sortimi dhe shtypja me funksion

```
#include <iostream>
using namespace std;
void SortoVektorin(int X[],int n);
void ShtypeVektorin(int X[],int n);
int main()
{
    const int m=5;
    int Shuma;
    int A[m]={3,2,1,4,5};
    SortoVektorin(A,m) ShtypeVektorin(A,m);
}
```

```

    return 0;
}

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

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

```

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

```

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

int main()
{
    const int m=5;
    int Amin, Amax;
    int A[m]={3,2,1,4,5}; ShtypeVektorin(A,m);
    Amin=AntMinVektorit(A,m ); Amax=AntMaxVektorit(A,m);

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

    return 0;
}

void ShtypeVektorin(int X[],int n)
{

```

```

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

```

Funksionet dhe matricat

// Programi FunksionMatrical

```
#include <iostream>
```

```
using namespace std;
```

```
const int n=3;
```

```
void FormoMatricen(int A[][n],int m);
```

```
int main()
```

```
{
```

```
    const int m=3;
```

```
    int i,j,A[m][n];
```

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

```
    cout << "Matrica A: \n";
```

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

```
    {
```

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

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

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

```
    }
```

```
    return 0;
```

```
}
```

```
void FormoMatricen(int A[][n],int m)
```

```
{
```

```

    int i,j;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
            A[i][j]=i+j;
    }
    return;
}
// Programi FunkSIONMatrica2
#include <iostream>
using namespace std;
const int n=3;
double ShumaAntMatrices(int X[][n],int m);
int main()
{
    const int m=2, n=3;
    double Shuma;
    int A[m][n]={ {1,2,3},
                  {4,5,6},
                  };
    Shuma=ShumaAntMatrices(A,m);
    cout << "Shuma e anetareve te matrices S="
         << Shuma
         << "\n";
    return 0;
}
double ShumaAntMatrices(int X[][n],int m)
{
    int i,j;
    double s;
    s=0;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
            s=s+X[i][j];
    }
    return s;
}
// Programi FunkSIONMatrica2
#include <iostream>
using namespace std;
const int n=3;
double ShumaAntPozMat(int X[][n],int m);
int main()
{
    const int n=3;
    int i,j,m;
    int X[10][n];
    cout<< "Sa rreshta do t'i kete matrica: ";
    cin >> m;
    double Shuma;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
        {
            cout << "X[" << i+1 << "][" << j+1 << "]=";

```

```

        cin >> X[i][j];
    }
}
Shuma=ShumaAntPozMat(X,m);
cout << "Shuma e anetareve pozitiv te matrices S="
    << Shuma
    << "\n";
return 0;
}
double ShumaAntPozMat(int X[][n],int m)
{
    int i,j;
    double s;
    s=0;
    for (i=0;i<m;i++)
    {
        for (j=0;j<n;j++)
            if (X[i][j]>0)
                s=s+X[i][j];
    }
    return s;
}

```

Funksionet matematikore të bibliotekës “cmath” në gjuhën programore C++

Përveç funksioneve të ndryshme që mund të krijojmë, C++ gjithashtu përfshin disa funksione të dobishme që mund ti përdorim. Këto funksione janë në dispozicion në bibliotekën standarde në C dhe C++ dhe quhen funksione të para ndërtuara (ang. **built-in functions**). Këto janë funksione që mund të përfshihen në programet që i krijojmë dhe pastaj ti përdorim.

C++ ka një seri të pasur të operacioneve matematikore të cilat mund të kryhen me numra të ndryshëm. Ja disa funksione të dobishme të para krijuara në bibliotekën cmath të C++-it

Nr.	Funksioni & Qëllimi
1	double cos(double); Ky funksion merr një kënd (si double) dhe kthen kosinusin e tij.
2	double sin(double); Ky funksion merr një kënd (si double) dhe kthen sinusin e tij.
3	double tan(double); Ky funksion merr një kënd (si double) dhe kthen tangjentin e tij.
4	double log(double); Ky funksion merr një numër dhe kthen logaritmin natyral të atij numri.
5	double pow(double, double) Numri i parë është baza kurse numri i dytë fuqia e bazës
6	double hypot(double, double); Nëse këtij funksioni i japim gjatësitë e dy anëve të trekëndëshit kënddrejt, ai to dë kthejë gjatësinë e hipotenuzës.
7	double sqrt(double); Funksion i cili kthen rrënjën katrore të numrit.
8	int abs(int); Ky funksion kthen vlerën absolute të numrit të plotë.
9	double fabs(double); Ky funksion kthen vlerën absolute të numrit decimal.
10	double floor(double); Ky funksion kthen vlerën e plotë të numrit decimal edhe atë duke rrumbullaksuar kah më e vogla.

Tipi i të dhënave që ceket në kllapa double është tip i numrave real me saktësi të dyfishtë(15 shifra të rëndësishme), kurse float po ashtu është tip i numrave real po me saktësi të thjeshtë (7 shifra të rëndësishme).

Për të shfrytëzuar këto funksione, duhet të përfshihet në krye të programit biblioteka <cmath>. Ose duke e shtuar këtë rresht menjëherë para komandës para procesorike si në vijim:

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

Ja një shembull me disa funksione matematikore:

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

int main ()
{
    // deklarimi dhe inicializimi i variablave:
    short s = 10;
    int i = -1000;
    long l = 100000;
    float f = 230.47;
    double d = 200.374;

    // operacione matematikore;
    cout << "sin(d) :" << sin(d) << endl;
    cout << "abs(i) :" << abs(i) << endl;
    cout << "floor(d) :" << floor(d) << endl;
    cout << "sqrt(f) :" << sqrt(f) << endl;
    cout << "pow( d, 2) :" << pow(d, 2) << endl;
    return 0;
}
```

Pasi të përkthejmë (kompajlojm), ndërtojmë dhe ekzekutojmë programin në dalje do të fitojmë këtë rezultat.

```
sin(d) :-0.634939
abs(i) :1000
floor(d):200
sqrt(f) :15.1812
pow( d, 2 ) :40149.7
```

1. Të shkruhet funksioni për llogaritjen e vlerës absolute për një vlerë të caktuar.

```
#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
    cout<< "Vlera absolute e 3.1416 sht " << abs (3.1416) << endl;
    cout<< "Vlera absolute e -2.89 sht " << abs (-2.89) << endl;
    return 0;
}
```

2. Të shkruhet funksioni për llogaritjen e gjatësisë së hipotenuzës për trekëndëshin kënddrejtë, nëse dihen gjatësia e brinjës a dhe brinjës b.


```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double a, b;
    cout<<"Llogaritja e hipotenuzës për trekëndshin këndrejt\n";
    cout << "Shëno gjatësinë e brinjës a:";
    cin >> a ;
    cout << "Shëno gjatësiën e brinjës b:";
    cin >>b;
    cout << "Gjatësia e hipotenuzës:" << hypot(a, b) << '\n';
    system("pause");
    return 0;
}
```

3. Të shkruhet programi për llogaritjen e sipërfaqes së rrethit duke përdorur funksionin **pow**.

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

int main()
{
    float r, s;
    const float pi=3.14159;
    cout<< "Vlera e rrezes r = ";
    cin >> r;
    s = pi*pow(r,2);          //pow(r,2) e ngrit ne katror rrezen r
    cout<< "\nSiperfaqja e rrethit: s = " << s << endl;
    return 0;
}
```

4. Të shkruhet programi për llogaritjen e rrënjës katrore të një numri të dhënë, duke e përdorur funksionin **sqrt**.

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

int main()
{
    double x = 25;
    cout<<"Rrenja katrore e "<<x <<" eshte " <<sqrt(x);
    system("pause");
    return0;
}
```

5. Të shkruhet programi për llogaritjen e sinusit të një këndi të caktuar duke përdorur funksionin **sin**.

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

#define PI 3.14159265

int main ()
{
    double kendi;
    kendi = 30.0;
```

```

    cout<<"Sinusi i " << kendi << " shkalleve eshte "
        << sin (kendi*PI/180) <<endl;
    system("pause");
    return 0;
}

```

6. Të shkruhet programi për llogaritjen e kosinusit të një këndi të caktuar duke përdorur funksionin **cos**.

```

#include <iostream>
#include <cmath>
using namespace std;
#define PI= 3.14159265
int main ()
{
    double kendi;
    kendi = 45.0;
    cout<<"Kosinusi i " << kendi << " shkalleve është
        << cos (kendi*PI/180) <<endl;
    system("pause");
    return 0;
}

```

7. Të shkruhet programi për llogaritjen e tangjentit për një vlerë të caktuar, duke përdorur funksionin **tan**.

```

#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double x = 0.3333;
    cout<<"Tangjenti i " <<x <<" është " << tan(x);
    return 0;
}

```

8. Të shkruhet programi për llogaritjen e shprehjes $y=e^x$ duke përdorur funksionin **exp**.

```

#include <iostream>
#include <cmath>
using namespace std;
int main ()
double x = 2.7;
cout<<"'e' e ngritur ne fuqinë 2.7 është " <<exp(x) <<endl;
system ("pause");
return 0;
}

```

9. Të shkruhet programi për llogaritjen e $\ln(x)$ për një vlerë të caktuar duke e përdorur funksionin **log**.

```

#include <iostream>
#include <cmath>
using namespace std;
int main ()
{
    double x=5.8;
    cout<<"ln(" <<x<<") = " <<log (x);
    system("pause");
    return 0;
}

```

TIPET E STRUKTURUARA TË TË DHËNAVE

Deri më tani janë shfrytëzuar tipe të thjeshta të të dhënave (int, float, double etj.). Mirëpo egziston nevoja edhe për tipe të ndërlikuar të të dhënave, elementet e të cilëve janë tipe të thjeshtë të të dhënave. Këto tipe janë të ndërtuar sipas rregullave saktë të definuar duke ndërtuar ashtu struktura. Këto tipe quhen tipe të strukturuar të të dhënave. Këto tipe përcaktohen nga tipi i komponentëve të saj dhe nga mënyra e strukturimit të tyre. Në rast se edhe komponentat e këtyre strukturave janë të strukturuar atëherë bëhet fjalë për tip të strukturuar me më shumë nivele.

VARG

Ndonjëherë në programe kemi nevojë për më shumë të dhëna nga një tip i njëjtë. Ashtu që në vend që të definojmë për cdo të dhënë ndryshore të veçantë ne mundemi që të definojmë një ndryshore por me më shumë komponenta – për cdo të dhënë një komponentë. Ky tip quhet varg (fushë)

Numri i elementeve është i definuar saktë dhe jantë të tipit të njëjtë. Afrimi deri tek çdo element bëhet me anë të indeksit – që paraqet pozitën e elementit në atë varg. Të dhënat në varg mund të jenë te cilit do tip.

Numri i indekseve në varg nuk është i kufizuar, a mvarësisht nga numri i tyre kemi varg njëdimensional ose fushë njëdimensionale nëse numri i indekseve është 1, dydimensional ose fushë dydimensionale nëse numri i indekseve është 2, e ashtu me radhë.

Për ti kuptuar më mirë vargjet (fushat numerike), do të marrim si shembull grumbullimin e notave të 5 nxënësve të parë, në regjistrin e notave të një klase, i cili është dhënë në Fig. 1

Numri rendor	Emri	1	2	3	4	5	6	7	8
		Gjuhe shqipe	Gjuhe angleze	Matema tike	Fizike	Kimi	Biologji	Progra mimi	Sport
1	Arta	5	5	5	5	5	5	5	4
2	Artan	5	5	3	5	5	4	5	3
3	Arjeta	5	5	5	5	5	5	5	5
4	Gona	5	5	5	5	5	5	5	5
5	Dita	4	4	2	5	5	4	4	2

Fig 1

Këtu grumbulli i notave të një nxënësi është varg (fushë) njëdimensionale dhe e paraqet vargun

(vektorin) e notave të tij. Kështu, p.sh vektori i notave të Arjetes është :

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

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

Vargjet (fushat) deklarohen si variabla të indeksuara, duke i shkruar indeksat brenda kllapave të mesme. Në gjuhën C++ indeksat fillojnë me zero.

Si edhe tipet e thjeshtë të të dhënave edhe këto tipe duhet deklaruar, poashtu në pjesën për deklarimin e ndryshoreve dhe atë në këtë formë :

tipi Emriifushe[dimensioni] = { elementi1, elementi2, ..., elementin};

Shembulli1

Për deklarimin e vargut D me 5 anëtarë duhet kështu;

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

Meqë sic u tha edhe më sipër, indekset fillojnë me vlerën zero, pas deklarimit të mësipërm, indekset e 5 anëtarëve të vektorit D janë: D[0], D[1], D[2], D[3], D[4].

Shembulli2

```
int notat[40];
```

Pra kemi deklaruar një ndryshore të tipit varg, të quajtur notat e cila ndryshore mund të ketë maksimum 40 elemente të tipi int (të plotë), pra do të paraqet notat e klasës në ndonjë lëndë. Çdo element i vargut është i definuar me indeksin e saj pra p.sh notat[i] paraqet elementin e i-të në atë varg. Në matematikë do të shkruhej kështu : notat_i.

Sa i përket operacioneve elementare si për shembull shoqërimi dalja dhe hyrja bëhen njëlloj si edhe me tipe të ndryshoreve të thjeshtë por duke u bazuar në indeksin e elementit p.sh.

notat[3]={6}; - do të thotë se element i 3-të në varg do të merr vlerën 6

notat[i]={8}; - do të thotë se element i-të do të merr vlerën 8

notat[i+1]={A}; - do të thotë se element i+1-të në varg do të merr vlerën e ndryshores A;

B=notat[2]; - do të thotë se ndryshore B do ti shoqërohet vlera e elementit të 2-të vargut etj

char Z[6]={ 'd', '4', '*', 'a', 'G', '\$' }; - do të thotë vektorit Z i shoqërohen 6 karaktere

const int m=5; int A[m]={7,2,4,1,3}; - do të thotë se vektorit A me indeks m=5 i shoqërohen 5 anëtar

Leximi (hyrja) e vlerave bëhet poashtu me urdhëresën për hyrje (lexim) cin pra

cin<<notat[2]; - do të thotë se i lexohet vlerë elementit të dytë të vargut

cin<<notat[i]; do të thotë se i lexohet vlerë elementit të i-të të vargut

Në mënyr identike bëhet edhe dalja (paraqitja) e vlerave të ndryshoreve bëhet me urdhëresën cout njëlloj si edhe me tipet e thjeshta të ndryshoreve pra :

cout<<notat[2] – do të thotë se paraqitet vlera e elementit të dytë në ekran

cout<<notat[i] – do të thotë se paraqitet vlera e elementit të i-të të vargut në ekran

Për futje (lexim) ose dalje (paraqitje) të vlerave të këtyre vargjeve shfrytëzohet struktura e urdheresës për përsëritje me numër të caktuar të përsëritjeve pasi që përputhet me natyrën e vargjeve pra kemi numër të caktuar të elementeve, pra

```
for(i=0; i<n; i++)
    cin>>notat[i] <<endl;
```

Pra duke filluar me vlerë i=0 do të lexohet (futet) vlera e elementit të parë pra notat₁, mandej ndryshorja i fiton vlerë 1 pra do të lexohet vlera e elementit të dytë e ashtu me radhë deri tek elementi i n-të pra element i fundit notat_n. Eventualisht nëse dëshirojmë që të plotësojmë këtë lexim me tekst për të patur më të qartë se çfarë jemi duke lexuar (fatur) atëherë do të shkruarjmë kështu

```
#include <iostream>
using namespace std;
int main()
{
    int notat[40];
    int i;
    int n;
    cout<<"Shkruani numrin e notave qe doni te utni";
    cin>>n;
    cout<<"futni notat";
    for(i=0; i<n; i++) cin >>notat[i];
    return 0;
}
```

Njëlloj si në shembullin e parë por me përcjellje të tekstit futni notën dhe numri rendor i notës që duhet futur, ashtu me radhë prej te nota e parë e deri tek nota e fundit.

Poashtu edhe për dalje shfrytëzohen këto struktura në mënyrë identike si edhe për hyrjen e vlerave p.sh

```
for(i=0; i<n; i++)
    cout << notat[i] <<endl;
```

Pra duke filluar me vlerë i=0 do të del (paraqitet) vlera e elementit të parë pra notat₁ në ekran, mandej ndryshorja fiton vlerë 2 pra do të paraqitet vlera e elementit të dytë e ashtu me radhë deri tek elementi i n-të pra elementi i fundit notat_n. Eventualisht nëse dëshirojmë që të plotësojmë këtë paraqitje të vlerave me tekst për të patur më të qartë se çfarë jemi duke paraqitur në ekran (dalje) atëherë do të shkruarjmë kështu :

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

```

    int notat[40];
    int i; int n; n=10;
    cout<<"futni notat"<<endl;
    for(i=0; i<n; i++)
        cin >>notat[i] ;
    return 0;
}

```

Njëlloj si në shembullin e parë por me përcjellje të tekstit vlera e elementit të numri rendor i notës që duhet paraqitur, e ashtu me radhë prej te nota e parë e deri tek nota e fundit.

Shembull: Program per futjen e notave ashtuqe numri i notave qe futen lexohet nga tastiera

```

#include <iostream>
using namespace std;
int main()
{
    int notat[40];
    int i;
    int n;
    cout<<"Shkruani numrin e notave qe doni te futni"<<endl;
    cin>>n; cout<<"futni notat"<<endl;
    for(i=0; i<n; i++)
        cin >>notat[i] ;
    return 0;
}

```

Shembull : Programi permes se cilit gjendet anetari me i madh ne vektorin e dhene

A[m].Programi "for"

```

#include <iostream> using
namespace std; int main()
{
    const int n=7;
    int i, b, A[n] = {2,9, -1, 4, 3,6, 8};
    b = A[0];
    for (i=0;i<n;i++)
    {
        if (A[i] > b)
            b = A[i];
    }
    cout<<"Anetari me i madh b = "<<b<< "\n";
    return 0;
}

```

Shembull :Te shtypen anetaret pozitiv dhe negativ te vektorit M[n]

```

#include <iostream>
#include <math.h> using
namespace std; int main()
{
    const int n=6;
    const double M[n]={-1.3 ,3.14, -6.5, -0.5, 3, 7.1};
    int negativ = 0,pozitiv = 0;
}

```

```

int i;
for(i=0;i<n;i++)
{
    if(M[i] >= 0)
    {
        pozitiv = pozitiv +1;
    }
    else
    {
        negativ = negativ +1;
    }
}
cout<<"Vektori:\n\nM[" <<n <<"]={";
for(i=0;i<n;i++)
{
    cout<<M[i];
    if(i!=n-1)
        cout<< ", ";
}
cout<<"} Permban " << negativ <<" anetar negativ"
<<" dhe " <<pozitiv <<" anetar pozitiv.\n\n";
return 0;
}

```

VARG SHENJASH - STRING

Tipi shenjë është sqaruar tek tipi i thjeshtë it ë dhënave. Në këtë ndryshore të tipit të shenjë mund të shoqërohet një shenjë. Paraqitet nevoja në programet të shfrytëzohen ndryshore të cilave do të mund t'u shoqërohen më shumë shenja p.sh. emri, mbiemri, profesioni etj. Për këtë qëllim shfrytëzohen ndryshore të deklaruar si vargje shenjash, respektivisht të deklaruar në tip të strukturuar të vacant të ashtuquajtur (string).

Mund të thuhet se string paraqet varg njëdimenzional të paketuar nga elemente të tipit shenjë.

Vlerat e tipit string janë vargje të shenjave ASCII, ndërsa gjatësia e tyre mund të jetë deri më 255. Mund të deklarohet edhe i ashtuquajturi nullstring pra stringu me gjatësi 0. Nëse gjatësia e stringut nuk është përcaktuar gjatë deklarimit të ndryshores, atëherë gjatësia e saj merret ajo maksimale pra 255.

Shembull për deklarimin e ndryshores të tipit string :

```
char emri[15];
```

```
char mbiemri[15];
```

Deri tek cilido element i stringut mund të afrohet nëpërmjet indeksit – numrit rendor të shenjës në stringun. Për shembull, nëse ndryshorja me emër emri deklaruar si string ka vlerë “SHKOLLA” , atëherë emrië0ç është ‘S’, emrië2ç është ‘H’, etj. Te tipi string mund të shfrytëzohen operatorët për shoqërim dhe relacion. Procedurat për lexim dhe paraqitje shfrytëzohen operatorët për shoqërim dhe relacion. Procedurat për lexim dhe paraqitje (cin, cout) si parametra mund të shfrytëzojnë ndryshore të deklaruar si string. Dallimi ndërmjet tipit shenjë dhe string është se një ndryshore e tipit shenjë mund të ketë gjatësi maksimale 1 pra një shenjë, ndërsa ndryshorja e tipit string si që e thamë pak më parë prej 1 deri në 255 ku në rast gjatësie prej 1 atëherë është i ngjajshëm me tipin shenjë pasi që komponentat e stringut janë të tipit shenjë.

Shembull:

```
const char='#';
```

```
char s1[1];
```

```
char string;
```

atëhere janë të lejuar këto shprehje :

```
s1=simbol;
```

```
s=simbol;
```

```
s=s1;
```

```
s1='+'
```

```
etj.
```

Ne C++ për punë me stringje janë të definuar shumë procedura dhe funksione standarde. Disa prej tyre me spjegim elementar janë këto :

Përcaktimi i gjatësisë së tekstit që lexohet

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

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

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

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

```
//program shtypja
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char a[10];
```

```
    cout<<"Teksti hyres: ";
```

```
    cin.width(5);
```

```
    cin>>a;
```

```
    cout<<"\nTeksti i lexuar: "
```

```
        <<a
```

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

```
    return 0;
```

```
}
```

Nëse ekzekutohet programi i dhënë dhe nëse pas mesazhit :

Teksti hyrës :

Si vlerë hyrëse përmes tastierës, shkruhet fjala Struga, si rezultat në ekran do të shtypet kështu:

Teksti i lexuar :Stru

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

Në program, te komanda për shtypje:


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

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

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

Leximi i fjalive

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

Për leximin e teksteve të cilat përmbajnë edhe zbraztira, përdoret komanda e vecante për lexim :

cin.get(a,k);

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

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

```
//program lexo
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char A[20];
```

```
    cout<<"Fjalja qe lexohet: ";
```

```
    cin>>A;
```

```
    cout<<"Fjalja qe u lexua: "
```

```
        <<A
```

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

```
    return 0;
```

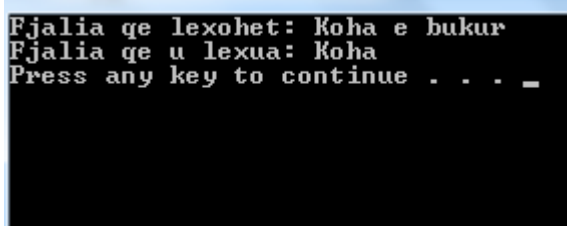
```
}
```

Nëse ekzekutohet programi i dhënë dhe si vlerë hyrëse përmes tastierës kompjuterit i jepet fjalja:

Koha e bukur, në ekran do ta kemi pamjen:

Fjalja që lexohet : Koha e bukur

Fjalja që u lexua : Koha



```
Fjalja qe lexohet: Koha e bukur
Fjalja qe u lexua: Koha
Press any key to continue . . . _
```

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

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

```
//program lexo
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```

const int m=20;
char A[m];
cout<<"Fjalja qe lexohet: ";

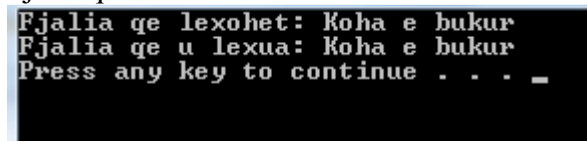
cin.get(A,m);
cout<<"Fjalja qe u lexua: "
    <<A
    <<"\n";
return 0;
}

```

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

Fjalja qe lexohet : Koha e bukur

Fjalja qe u lexua : Koha e bukur



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

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

Leximi komplet i rreshtit

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

`cin.getline(a,k);`

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

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

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

```

//program lexo
#include <iostream>
using namespace std;
int main()
{const int m=20;
    char A[m];
    cout<<"Fjalja qe lexohet: ";

    cin.getline(A,m);
    cout<<"Fjalja qe u lexua: "
        <<A
        <<"\n";
    return 0;
}

```

FUNKSIONET PËR PUNË ME STRINGJE

Nga grumbulli i funksioneve për punë me stringje, të cilat përfshihen në modulën string, përmes shembujve këtu do të tregohet përdorimi i vetëm disa prej tyre. Në fillim të çdo programi vijues është vendosur komanda paraprocesorike `#include <string>`, për ta njoftuar kompjuterin se do të shfrytëzohen funksione të modulit në fjalë.

Gjatësia e stringut

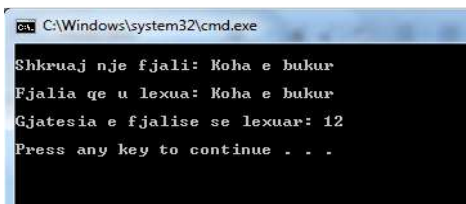
Për përcaktimin e gjatësisë të stringut shfrytëzohet funksioni `strlen(x)`, ku `x` është variabla përmes së cilës ruhet stringu në memorien e kompjuterit.

Shembull Programi përmes së cilit përcaktohet gjatësia e tekstit të lexuar në variablën `A`, për të cilën në memorien e kompjuterit janë rezervuar 20 simbole.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    const int m=20;
    char A[m];
    int b;
    cout << "\nShkruaj nje fjali: ";
    cin.getline(A,m);
    cout << "\nFjalja qe u lexua: "
         << A
         << "\n";
    b=strlen(A);
    cout << "\nGjatesia e fjalise se lexuar: "
         << b
         << "\n\n";
    return 0;
}
```

Në program, për leximin e fjalisë është shfrytëzuar komanda: `cin.getline(A,m);`

Nëse ekzekutohet programi i dhënë më sipër, pasi përmes tastierës të shkruhet fjalia Koha e bukur, rezultati do të duket si në Fig



Kopjimi i stringut

Komplet stringu mund të kopjohet në një string tjetër duke e shfrytëzuar funksionin `strcpy`. Por, ekziston mundësia e kopjimit të vetëm një pjese të stringut përmes funksionit `strncpy`.

Kopjimi i komplet stringut

Stringu `x` mund të kopjohet në stringun `y`, duke e shfrytëzuar funksionin `strcpy(y,x)`.

Shembull Programi përmes së cilit tregohet kopjimi te variabla `B`, i fjalisë së lexuar, e cila ruhet te variabla `A`.

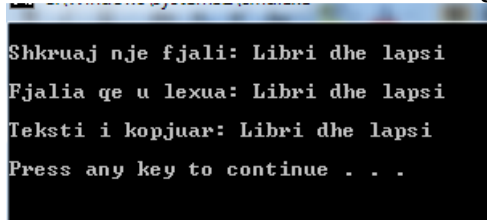
```
// Programi
#include <iostream>
#include <iostream>
#include <string>
using namespace std;
```

```

int main()
{
    const int m=20;
    char A[m],B[m];
    cout<<"\nShkruaj nje fjali: ";
    cin.getline(A,m);
    cout<<"\nFjalja qe u lexua: " <<A << "\n";
    strcpy(B,A);
    cout<< "\nTeksti i kopjuar: " <<B << "\n\n";
    return 0;
}

```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepet teksti Libri dhe lapsi, rezultati në ekran do të duket si në Fig.



```

Shkruaj nje fjali: Libri dhe lapsi
Fjalja qe u lexua: Libri dhe lapsi
Teksti i kopjuar: Libri dhe lapsi
Press any key to continue . . .

```

Kopjimi i një pjese të stringut

Përmes funksionit **strncpy(y,x,n)**, te stringu y mund të kopjohen vetëm n-simbolet e para të stringut x.

Shembull Programi përmes së cilit tregohet kopjimi te variabla B, i nsimboleve të para të fjalisë së lexuar, e cila ruhet te variabla A.

```

// Programi string
#include <iostream>
#include <string>
using namespace std;
int main()
{
    const int m=20;
    int n;
    char A[m],B[m];
    cout << "\nShkruaj një fjali: ";
    cin.getline(A,m);
    cout<< "\nFjalja që u lexua: "
        <<A
        << "\n";
    cout<<"\nNumri i simboleve që kopjohen: ";
    cin>>n;
    strncpy(B,A,n);
    B[n]='\0';
    cout<<"\nTeksti i kopjuar: "
        <<B
        <<"\n\n";
    return 0;
}

```

Këtu, pas kopjimit të n-simboleve të para të fjalisë, duke e shfrytëzuar funksionin:
`strncpy(B,A,n);`
 përmes komandës:

`B[n]='\0';` në fund të pjesës së kopjuar të fjalisë vendoset karakteri zero.

Nëse ekzekutohet programi i dhënë dhe përmes tastierës, si string hyrës, kompjuterit i jepet fjalia Libri dhe lapsi, kurse për numrin n të simboleve që kopjohen në tastierë shkruhet numri 9, rezultati do të duket si në Fig

```

C:\Windows\system32\cmd.exe
Shkruaj nje fjali: Libri dhe Lapsi
Fjalja qe u lexua: Libri dhe Lapsi
Numri i sinboleve qe kopjohen: 9
Teksti i kopjuar: Libri dhe
Press any key to continue . . . _

```

Bashkimi i dy stringjeve

Stringut x mund t'i shtohet stringu y, duke e shfrytëzuar funksionin `strcat(x,y)`.
 Shembull Programi përmes së cilit tregohet shtimi i stringut B në vazhdim të stringut A.

```

// Programi string
#include <iostream>
#include <string>
using namespace std;
int main()
{
    const int m=10,n=25;
    char A[m+n],B[n];
    cout<< "\nTeksti i pare A: ";
    cin.getline(A,m);
    cout << "\nTeksti i dyte B: ";

    cin.getline(B,n);
    strcat(A,B);
    cout << "\nTeksti i bashkuar: " << A << "\n\n";
    return 0;
}

```

Nëse ekzekutohet programi i dhënë dhe përmes tastierës kompjuterit i jepen dy tekste, rezultati do të duket si në Fig.

```

C:\Windows\system32\cmd.exe
Teksti i pare A: Struga
Teksti i dyte B: , dhe Tetova
Teksti i bashkuar: Struga, dhe Tetova
Press any key to continue . . .

```

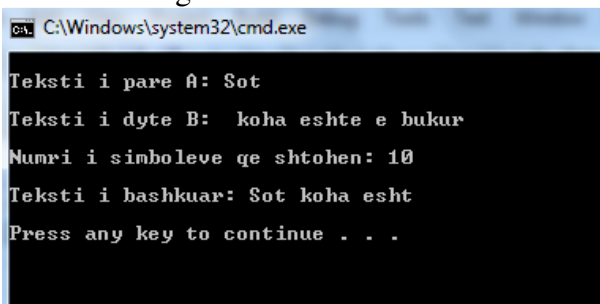
Shtimi i pjesës së stringut

Për shtimin e n-simboleve të stringut y në vazhdim të stringut x, shfrytëzohet funksioni `strncat(x,y,n)`.

Shembull Programi përmes së cilit tregohet shtimi i k-simboleve të stringut B në vazhdim të stringut A.

```
// Programi string8
#include <iostream>
#include <string>
using namespace std;
int main()
{
    const int m=10,n=25;
    char A[m+n],B[n];
    int k;
    cout << "\nTeksti i pare A: ";
    cin.getline(A,m);
    cout << "\nTeksti i dyte B: ";
    cin.getline(B,n);
    cout << "\nNumri i simboleve qe shtohen: ";
    cin >> k;
    strncat(A,B,k);
    cout << "\nTeksti i bashkuar: " << A << "\n\n";
    return 0;
}
```

Nëse, pas ekzekutimit të programit të dhënë, kompjuterit përmes tastierës i jepen dy tekstet e shfrytëzuara në shembujt paraprakë, dhe numri i simboleve që shtohen merret 10, rezultati do të duket si në Fig.



```
C:\Windows\system32\cmd.exe
Teksti i pare A: Sot
Teksti i dyte B: koha eshte e bukur
Numri i simboleve qe shtohen: 10
Teksti i bashkuar: Sot koha esht
Press any key to continue . . .
```

Nga figura e dhënë shihet se prej tekstit të dytë, në vazhdim të tekstit të parë janë shtuar 10 simbolet e para, përkatësisht janë shtuar simbolet: ***koha esht***.