ПРОГРАМИРАЊЕ
ЗА ДРУГИ РАЗРЕД

Садржај

Увод у програмске језике

Језик којим се споразумевају људи је природни језик. Поред природних језика постоје и вештачки језици. То су језици чија се правила дефинишу и пре него што језик почне да се користи. Баш зато што постоји велики број природних језика они су непогодни за комуникацију са рачунаром који користи искључиво машински језик, а који познаје веома мали број корисника. За ту комуникацију створени су ПРОГРАМСКИ језици. То је језик који је прилагођен тако да може да га прихвати рачунар.

ТО ЈЕ СКУП ПРАВИЛА И ЕЛЕМЕНАТА (СИМБОЛА) КОЈИ ЈЕ ТАЧНО ПРОПИСАН И ОД КОГА НЕМА ОДСТУПАЊА. Он је намењен описивању рачунарских поступака које треба да спроведе рачунар како би се неки проблем решио.

Програмски језик треба да има следеће особине:

1. да буде што сличнији људском начину изражавања
2. да буде формално дефинисан
3. да омогући јасан запис поступака решавања
4. да буде лак за усвајање
5. да омогући превођење на машински језик

Машински језик се своди на бинарно кодирање операција и бинарно навођење адреса променљивих у меморији. Машински језик различитих рачунара је различит. Због свега тога створени су програмски језици. Развој програмских језика траје кроз четири фазе:

1. машински језици
2. симболички машински језици
3. процедурални програмски језици
4. непроцедурални програмски језици

Развојно окружење се састоји из различитих програма (програмских алата). То су следећи програми: едитор, преводилац, повезивач и програм за отклањање грешака.

Од тренутка када је програм написан на папиру до тренутка када се изврши на рачунару, потребно је извршити његову припрему која пролази кроз 3 фазе.

1. Уношење програма

Уноси се преко тастатуре у рачунар као било који текст. То се зове изворни код. Изворни код за програмски језик C++ има наставак .cpp

2. Превођење програма

Да би рачунар разумео изворни код мора да се преведе. Постоје три врсте преводиоца:

а) асемблери и макроасемблери
б) компајлери
ц) интерпретатори

Асемблери служе за превођење симболичких машинских језика.

Компајлери преводе цео изворни програм и не учествује у његовом извршењу

Интерпретатори преводе сваку наредбу изворног програма и одмах је извршава.

Преведени програм се зове објектни модул и има наставак .OBJ

3. Повезивање програма

Објектни модул није спреман за директно извршење. Он представља улаз у ЛИНКЕР који има задатак да повеже модул са неким другим библиотекама и да формира извршни програм који има наставак .EXE

Приликом израде програма јављају се две врсте грешака:

а) синтаксне
б) семантичке

Синтаксне грешке се лако уочавају јер су то грешке приликом писања команди и врло често нас на њих упозорава и преводилац, а јављају се услед непоштовања правила за писање програма.

Семантичке грешке је много теже открити јер је то грешка у самој логици решавања проблема и обично доводе до погрешног решења задатка.

Питања:

1. Шта су програмски језици?
2. Подела програмских језика. (објаснити бар једну врсту програмских језика).
3. Фазе у програмирању.
4. Прва фаза програмирања.
5. Друга фаза програмирања.
6. Која је разлика између компајлера и интерпретатора.
7. Повезивање програма.
8. Грешке приликом писања програма.
9. Шта су то семантичке грешке?
10. Шта су то синтаксне грешке?

Алгоритми

Појам са којим се често срећемо је појам РАДЊЕ. То је нешто што има коначно трајање и доводи до жељеног разултата.

Свака радња има свој ОБЈЕКАТ над којим се извршава. Ако радња може да се разложи на саставне делове тада се поступак њеног извршавања назива израчунавање.

Неки сложени задатак се решава тако што се разлаже на низ простих „елементарних“ радњи. Овако дефинисан поступак зове се АЛГОРИТАМ.

Алгоритам представља коначан скуп добро дефинисаних правила за решавање неке класе задатака у коначном броју корака.

То је упутство за израчунавање, састоји се од извршења простих правила, по одређеном редоследу а чији је циљ долажење до коначног решења неког проблема.

Примена сваког правила зове се АЛГОРИТАМСКИ КОРАК. Добро дефинисано правило значи да је резултат примене правила увек исти, без обзира ко је извршилац.

Време извршења сваког правила мора бити коначно, а мора бити коначан и број примењених правила.

ОСОБИНЕ АЛГОРИТАМА:

Сваки алгоритам мора да поседује следећа правила

1. **Дискретност** – Алгоритам је састављен од коначног броја корака. Но сваки алгоритамски корак треба да покаже на који начин се стиже до следећег корака.
2. **Коначност** – све операције које треба да се изврше у неком алгоритамском кораку треба да се изврше у неком временском интервалу и не сме да постоји ни једна препрека за извршење.
3. **Резултативност** – особина која захтева да се алгоритам за било који задатак зауставља после одређеног броја корака и доводи до одређеног резултата.
4. **Одређеност** – тумачење и извршење алгоритма не сме да зависи од воље људи или машине, већ треба да буде разумљив за све могуће извршиоце.
5. **Масовност** – Ако се прави алгоритам за само тачно одређене податке он онда нема сврхе. Циљ алгоритма је да може бити примењен за велики број различитих података чиме он обезбеђује масовност.

Да би програмер написао програм треба да зна алгоритам решења постављеног задатка, односно треба да буде у стању да одради правила уз помоћ којих ће се доћи до решења.

При састављању алгоритма води се рачуна о следећем: полази се од познатог скупа података,

такође зна се које се решење тражи. Дакле, потребно је повезати те две врсте података, тј. пронаћи решење за алгоритам.

Програм на било ком програмском језику написан је само једна варијанта алгоритма.

Сам опис алгоритма има два циља:

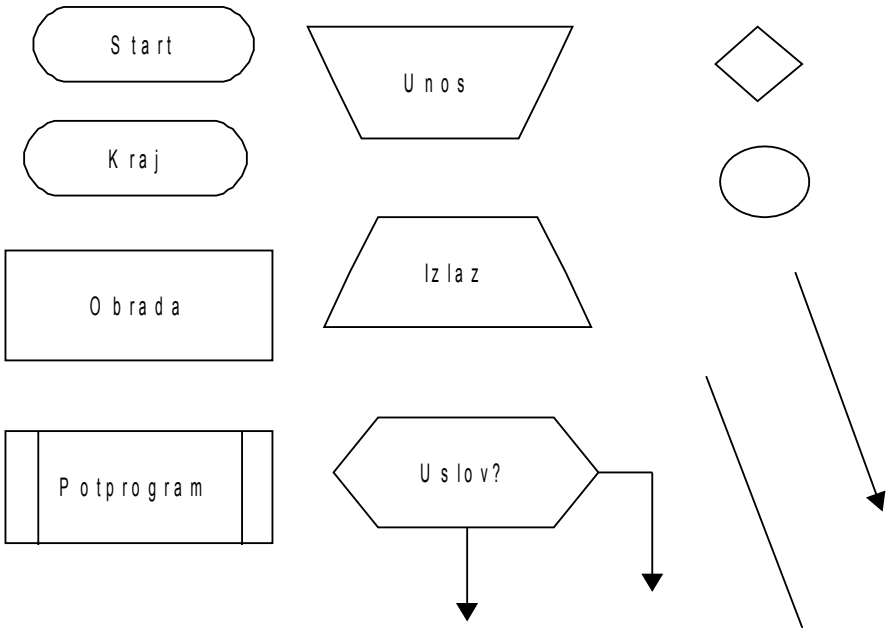
1) разумљивост
2) на основу њега треба да се напишу програми

Зато се каже да алгоритам представља „универзални“ језик који није прилагођен ни једном конкретном рачунару, а људи их могу лако пратити и преносити. Запис алгоритма путем прогамског језика представља програм.

Алгоритам се описује текстуално (као писани састав) и графичком шемом.

Графички приказ алгоритма

Алгоритме је најлакше представити графичким путем, јер на тај начин може најлакше да се изврши тестирање проблема. Графичке ознаке алгоритма су следеће:



Но пре него што се приступи графичком приказу проблема сваки задатак пролази кроз неколико фаза. То су фазе у програмирању које служе да би се од идеје дошло до њене потпуне реализације.

Фазе су:

1. Дефинисање проблема и израда пројектног задатка
2. Пројектовање програма
3. Писање и тестирање програма
4. Анализа рада програма и израда пројектне документације
5. Испорука и одржавање

Правилно дефинисање проблема (задатка) који се решава је прва и најважнија фаза у изради неког програма.

У неким ситуацијама дешава се да је проблем већ дефинисан и у потпуности јасан (пр. збир првих три природних бројева). Међутим, у пракси се често јављају ситуације у којима је најважније успешно открити и дефинисати проблем. Од начина на који је дефинисан проблем зависиће и израда програма. Уколико се правилно дефинише проблем се лакше и брже решава.

Дакле, треба сагледати:

1. Шта програм треба да ради
2. Који су улазни подаци (познате ствари од којих крећемо)

3. Шта су излазни подаци (тј. шта нам треба)
4. Која су ограничења о којима треба да водимо рачуна

Када се све то сазна прелази се на другу фазу: Пројектовање програма

У овој фази потребно је анализирати постављен задатак и осмислити програм за његово решавање.

Поступак решавања се показује на начин погодан за писање програма, најчешће у графичком облику.

Потребно је осмислити и сам изглед програма и начин комуникације са корисником.

Као што смо рекли, та трећа фаза је најлакша уколико су претходне две добро и темељно одрађене. Писање програма је део који се односи на представљање програма, најпре у виду алгоритма, а касније у у виду неког програмског кода.

Алгоритам представља основу за писање програма и само ако је алгоритам тачан биће тачан и код. Због тога је потребно да се најпре изврши тестирање алгоритма. То се чини на основу познатих величина и познатих резултата. Пример: ако знамо да је површина коцке чија је страна 2, 24, а запремина 8, ми лако можемо да прођемо кроз алгоритам и да утврдимо да ли нам алгоритам даје баш те резултате. Уколико је то тако приступамо изради програма у неком од програмских језика.

Потребно је да алгоритам тестирамо више пута, јер се деси да програм ради за неке полазне податке, а за друге не ради.

Такође, пожељно је да за све то што радимо водимо уредне белешке које ће бити основ за следећу фазу.

Последња фаза је испорука и одржавање програма. Одржавање програма је лако уколико имамо уредно вођену документацију коју смо начинили у свим претходним фазама док смо решавали проблем. Онда је лако унети евентуалне измене које корисник од нас захтева.

У зависности од тога какав проблем се решава постоје неколико структура писања програма, а самим тим и алгоритама.

Питања:

1. Шта је алгоритам?
2. Шта је алгоритамски корак?
3. Које су особине алгоритма?
4. Објаснити особину: дискретност.
5. Објаснити особину: коначност.
6. Објаснити особину: резултативност.
7. Објаснити особину: одређеност.
8. Објаснити особину: масовност.
9. Како можемо записати алгоритам?
10. Нацртати графички симбол за сваки алгоритамски корак.
11. Набројати и објаснити све фазе у програмирању.
12. Подела алгоритамских структура.
13. Карактеристике линијске алгоритамске структуре.
14. Карактеристике разгранате алгоритамске структуре.
15. Карактеристике цикличне алгоритамске структуре.

Линијска структура

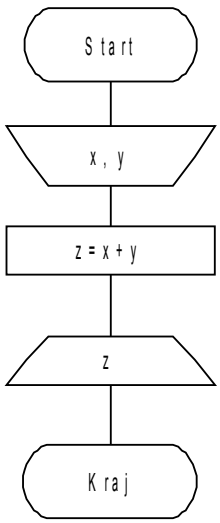
Најпростија структура је линијска структура. Карактеристична је по томе што се сваки корак извршава само једном и то у оном редоследу који је и записан. Састоји се од улазних корака, корака обраде и излазних корака.

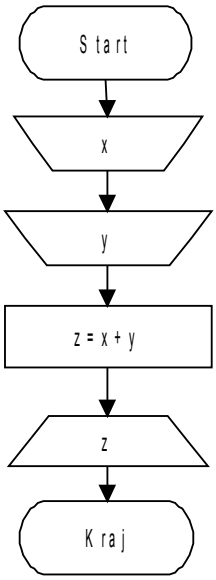
Разграната структура се користи онда када имамо дилему. То је условна структура и даљи ток програма зависи од тога да ли је услов испуњен или не. (Пример: Ако пада киша понећемо кишобран, а ако не пада идемо на пут без кишобрана).

Циклична структура се користи онда када се један или више корака понављају у истом редоследу. (Пример: Једна календарска година се састоји од 12 месеци и увек су у истом редоследу: јануар, фебруар... децембар. Немогуће је да имамо јануар па мај, онда октобар...).

Као што смо рекли, линијска структура је најпростија и састоји се од корака улаза, излаза и обраде.

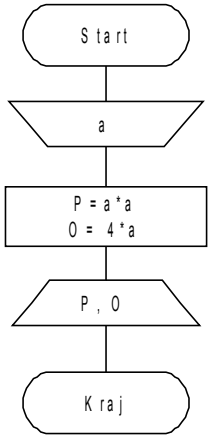
Пример: Нацртати алгоритам којим ће се сабрати два броја.



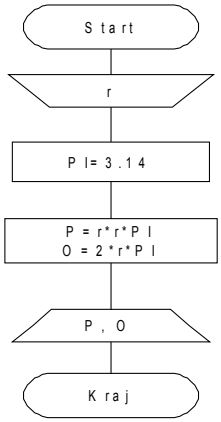


Примери:

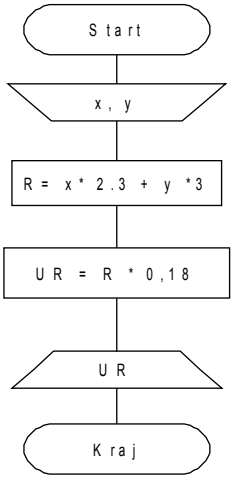
1. Саставити алгоритам за израчунавање површине и обима квадрата



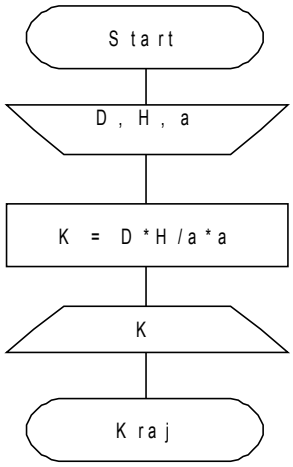
2. Саставити алгоритам за израчунавање површине и обима круга



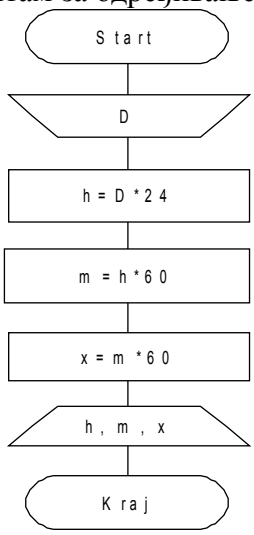
3. Потрошено је X киловата струје по цени од 2,3 динара и Y киловата струје по цени од 3 динара. На цео износ додаје се 18% пореза. Колико износи укупан рачун.



4. Зид дужине D и висине H треба облепити плочицама квадратног облика ивице a. Направити алгоритамску блок шему којом се израчунава потребна количина плочица.



5. Дат је број дана. Направити алгоритам за одређивање колико је то сата, минута и секунди.



6. Кроасан кошта x динара. Прво поскупи 10%, а онда појефтини 10%. Колико ће на крају да кошта тај кроасан.
7. Унети цене за три производа. На први додати порез од 8%, а на други и трећи додати порез од 18%. Израчунати укупну цену за сва три производа.
8. Унети вредности за четири отпорника. Израчунати њихову укупну отпорност ако су они редно везани.
9. Дата је вредност за два отпорника. Израчунати њихову укупну отпорност ако су они везани у паралелну везу.
10. Рачун за воду се израчунава тако што се помножи количина потрошене воде са ценом по литру. На то се додаје фиксни износ који представља додатак за одржавање акумулационог језера и на крају се додаје порез који износи 8% од суме на рачуну. Ако једно домаћинство утроши X литара воде и ако литар воде износи A динара. Колики рачун треба да уплате на рачун водовода.
11. Написати алгоритам којим се израчунава површина и запремина купе.

(P=r*Pi*(s+r), V=1/3*r*r*Pi*h)

12. Написати алгоритам којим ће се израчунати растојање између тачака А (x1, y1) и Б (x2, y2) које су задате координатама.

$$d = \sqrt{(x1-x2)^2+(y1-y2)^2}$$

13. Путник А прелази X km/h, а путник Б прелази Y km/h. Оба путника у исто време и из исте позиције почињу кружни обилазак града дужине C км. Направити алгоритам којим се одређује после колико времена ће бржи путник сустићи споријег.

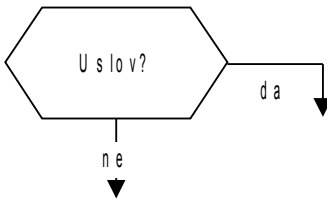
$$t = \frac{s}{|x - y|}$$

14. Воз се кретао T1 секунди брзином од V1 m/s. Затим је прешао X км брзином од V2 m/s. Направити алгоритам који израчунава колика је средња брзина воза на пређеном путу.
15. Дат је временски интервал изражен у минутима. Направити алгоритам који ће тај временски интервал да изражава у сатима и секундама.
16. Дата је правилна тространа пирамида висине h и дужине основице a. Написати алгоритам за израчунавање површине и запремине пирамиде.

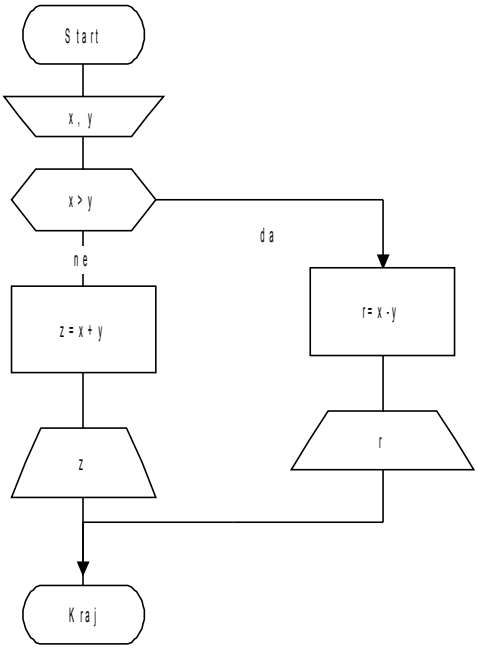
Разгранате структуре

Разгранате структуре су карактеристичне по томе што постоји могућност да се део програма никада не изврши. Ове структуре у себи садрже услов. Даљи ток програма зависи од тога да ли је тај услов испуњен или не.

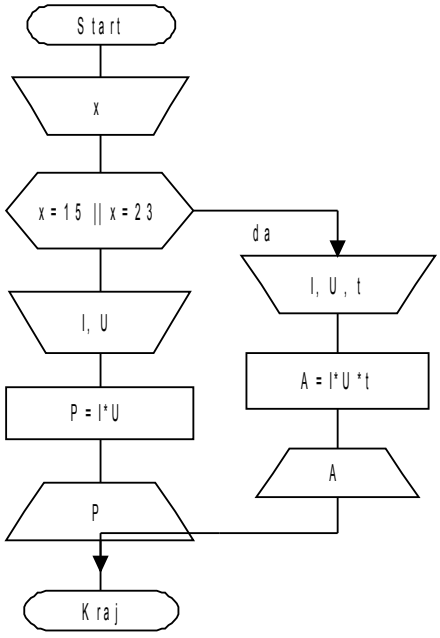
Графичка ознака услова је:



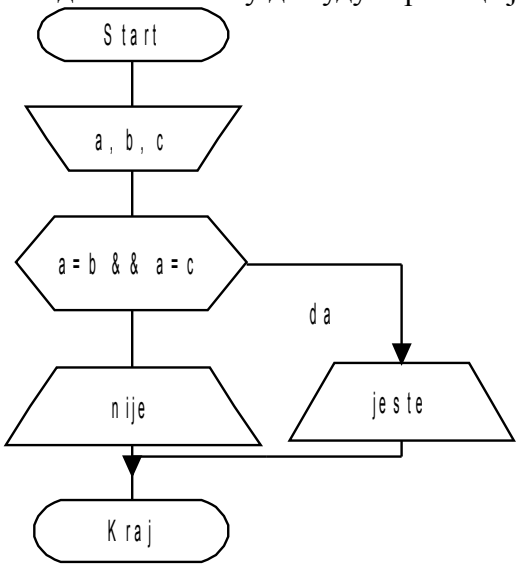
Пример: Унети два броја. Ако је први већи од другог израчунати њихову разлику, у супротном израчунати њихов збир.



1. Унети број. Уколико је он 15 или 23 израчунати рад електричне струје. У било ком другом случају израчунати снагу електричне струје.



2. Унети три броја па испитати да ли они могу да буду странице једнакостраничног троугла.



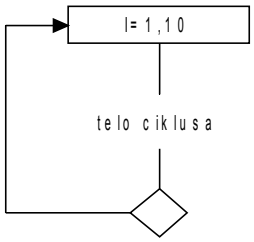
3. За вредност унету са тастатуре која није 108 израчунати снагу струје.
4. Унети број па испитати да је он двоцифрен.
5. Један купац је купио X тастатура по цени од K динара, а други је купио T мишева по цени од C динара. Ко је потрошио више пара?
6. Унети три броја, па одредити који је највећи.
7. Унети координате за три тачке, па одредити која је најдаља од координатног почетка. $d=\sqrt{x^2+y^2}$
8. Унети број, па утврдити да ли је он дељив са бројем 8.
9. Воз се креће брзином од x км на сат и прелази растојање од y км. Аутомобил јеу исто време кренуо и вози T км на сат, а треба да пређе Л км. Ко ће брже да стигне на одредиште?
10. Драган је купио X литара млека по цени од 80 динара. Лазар је купио У литара бензина по цени од 140 динара. Порез на млеко је 8%, а на бензин 18%. Ко је издвојио више новца за порез?
11. Унети три броја ако су они различити израчунати обим неједнакостраничног троугла. Ако су исти израчунати запремину коцке.
12. Израчунати обим квадрата. Ако је он 20 израчунати и површину, ако је мањи од 20 израчунати дијагоналу, а ако је већи од 20 израчунати полуобим тог квадрата.

Цикличне структуре

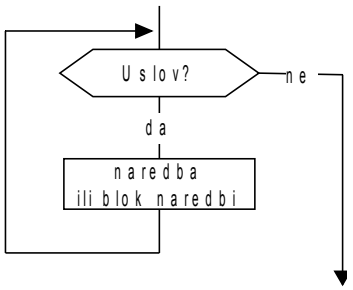
Цикличне структуре се разликују од других структура по томе што се код ових структура део програма понавља више пута. Уколико се део програма који се понавља састоји од више од једног корака редослед истих не сме да се мења, већ увек остаје исти.

Свака циклична структура има почетак и крај циклуса. Све између се назива телом циклуса.

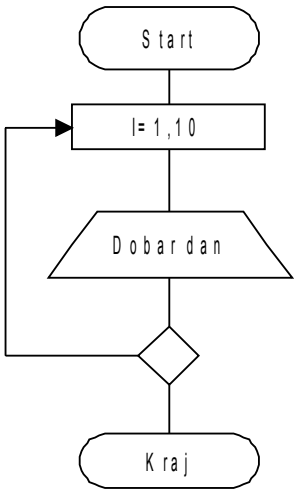
Постоје две врсте циклуса: бројачки и условни циклуси. Бројачки се користе онда када знамо колико пута се нешто понавља.



Условни циклуси се користе онда када не знамо колико пута ће нешто да се понови, већ постоји одређени услов који кад буде испуњен значи и крај циклуса.



Пример: Десет пута одштампати реченицу: Добар дан.



Примери:

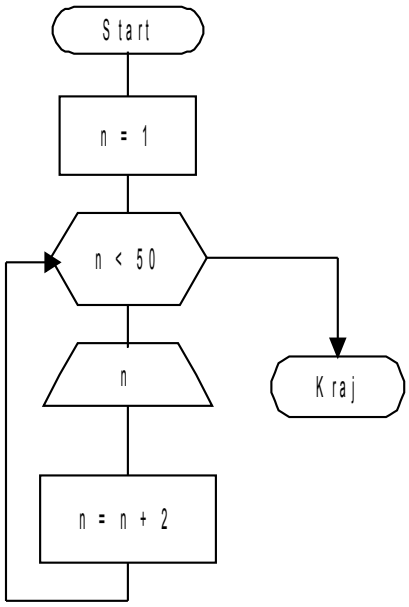
1. Написати алгоритам за израчунавање суме првих 10 бројева.
2. Саставити алгоритам којим ће десет пута да се штампа реченица „Добар дан“
3. Саставити алгоритам којим ће се израчунати производ 5 бројева унетих са тастатуре.
4. Саставити алгоритам којим се израчунава сума N бројева унетих са тастатуре.
5. Саставити алгоритам којим се израчунава просечна оцена за ученика који има 15 предмета.
6. Саставити алгоритам којим се израчунава аритметичка средина за N бројева унетих са тастатуре.
7. Унети вредности за N отпорника па израчунати колика је просечна вредност сваког отпорника.
8. Купац је купио X производа и сваки кошта B динара. Колико ће да износи рачун који треба

да плати.

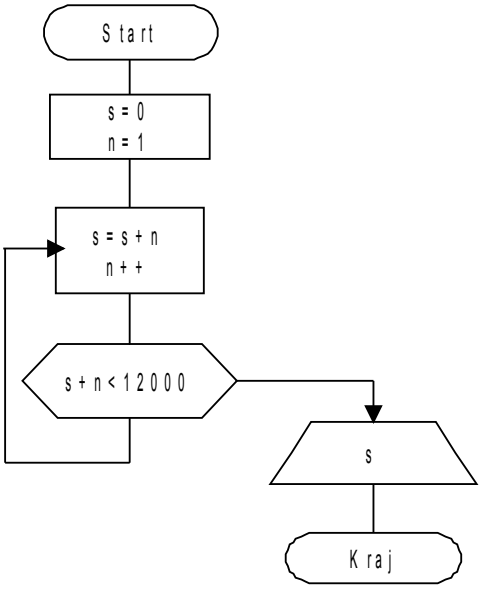
9. Сваког месеца неки потрошач утроши одређени број киловата струје. Цена једног киловата је 4 динара. Сваког месеца се плаћа и фиксни део од 50 динара. Колико износи рачун за целу годину, а колики је просечни рачун за тог потрошача.
10. Саставити алгоритам за израчунавање суме првих N бројева.
11. Саставити алгоритам за израчунавање производа N бројева унетих са тастатуре.

Комбиноване структуре

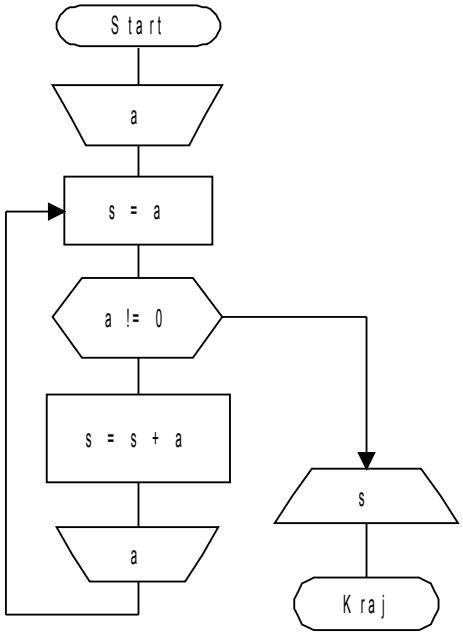
1. Написати програм који ће штампати све непарне бројеве мање од 50.



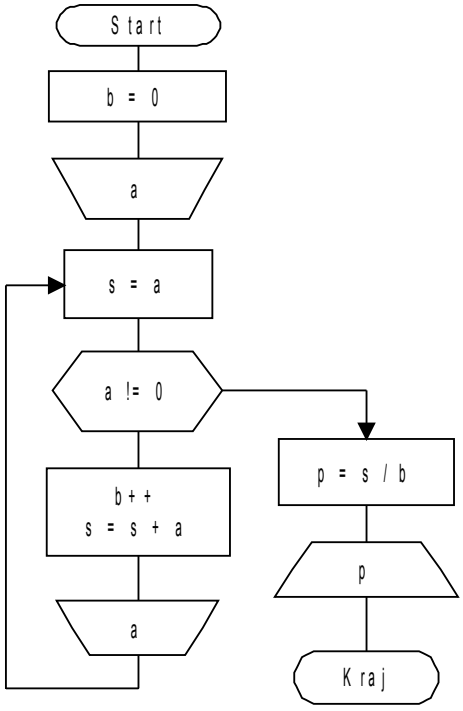
2. Написати програм који ће сабрати све природне бројеве док збир не буде већи од 12000.



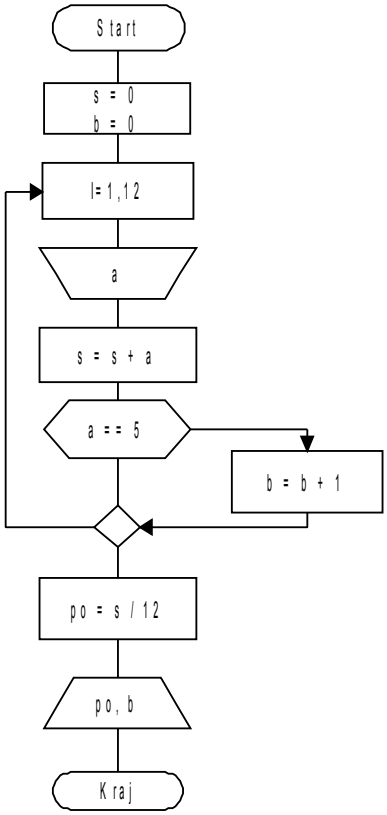
3. Уносити бројеве све док се не унесе нула. Израчунати суму унетих бројеве.



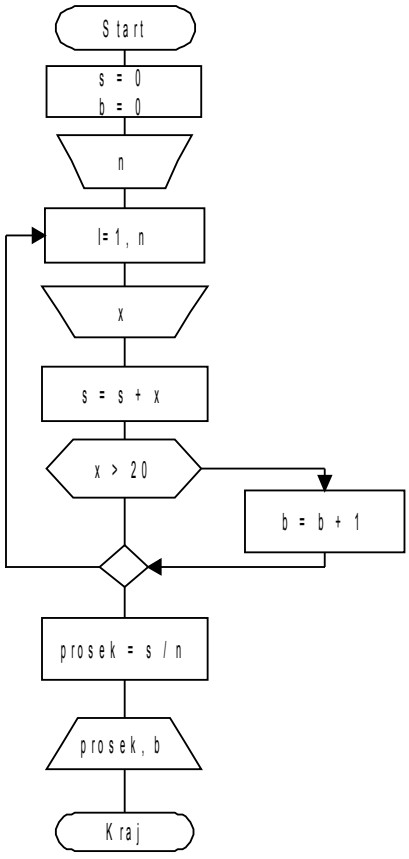
4. Уносити бројеве све док се не унесе нула. Израчунати просечну вредност унетих бројева.



5. Написати програм којим ће се за једног ученика унети оцене из 12 предмета и на основу њих израчунати просечна оцена и утврдити број петица.



6. Унети N бројева са тастатуре па израчунати њихову просечну вредност, као и колико је бројева веће од 20.



Програмски језик C++

C++ је виши програмски језик који је првобитно развијен у Bell Labs (лабораторији телекомуникационе компаније Bell) за објектно оријентисано програмирање у пројекту под руководством Бјарнеа Строуструпа током 1980их.

Карактеристике овог програмског језика:

1. Преноси се с рачунара на рачунар и може бити реализован на различитим архитектурама као и у различитим оперативним системима.
2. Има богат скуп операција
3. Семантика програмског језика је проста, а синтакса формално дефинисана

Азбука програмског језика C++ је проста (садржи велика и мала слова, бројеве, специјалне знаке и беле-нештампајуће знаке).

Скуп лексичких симбола (основни градивни елементи језика) дели се на:

1. идентификаторе
2. константе
3. операторе
4. сепараторе

а помоћу њих граде се и остале конструкције језика:

1. наредбе,
2. програми
3. директиве (наредбе претпроцесора)

Једна од основних карактеристика програмског језика C++ је претпроцесор.

Претпроцесор није део програмског језика C++, али је део преводилаца за програмски језик C++. Служи за обраду текста програма пре превођења. Наредбе претпроцесора зову се **директиве**. У претпроцесору се врши разне трансформације и припреме пред превођење превођење програмског кода.

Те трансформације укључују: уметање садржаја неке датотеке на одређено место у програму или условно укључивање (или изостављање) делова програма.

У принципу директиве претпроцесора садрже одређене библиотеке које објашњавају како треба извршити неку од наредби која се налази у програму. У зависности од тога шта је све потребно у програму неке од тих библиотека су укључене или изостављене из претпроцесора.

Све директиве у претпроцесору почињу знаком #.

Свака директива се пише у посебном реду.

Најчешће се користи директива:

`#include`

Директива **`#include`** врши уметање датотеке која представља део стандардног преводиоца за програмски језик C++. Има следећи облик:

```
#include < >
```

унутар заграда налазе се хедер фајлови. Хедер фајлови садрже информације о константама, функцијама, класама и другим елементима програмског језика C++. Хедер фајл *iostream* садржи информације о стандардним улазним и излазним токовима.

Пример: **`#include <iostream>`**

овај ред се не завршава са ; јер то није наредба програмског језика већ претпроцесорка директива.

У програмском језику C++ се врло често пишу и **коментари** у самом коду. Коментаре преводилац не преводи и они служе само да објасне неком кориснику шта се ради у неким наредним корацима. Коментари се пишу на следећи начин:

```
// ovde ide neki tekst
```

такође коментар може да се пише у једном или у више редова на следећи начин:

```
/*ovde ide neki tekst */

/*ovde
ide
neki
tekst */
```

Оно о чему мора да се води рачуна је да сваки коментар мора да има почетак и свој крај. Уколико нема ***/** овог знака на крају преводилац ће и регуларни део кода да смести под коментар и неће да га преводити.

Оно о чему мора посебно да се води рачуна у програмском језику C++ су идентификатори и кључне речи.

Како је могуће да више програмера раде на истом програму и користе исте променљиве за различите ствари и тада долази до грешке у програму. Да би се то избегло користи се механизам *namespace*. Сва стандардом предвиђена заглавља стављају се у простор *std*. Тако да у C++-у имамо следећи ред:

```
using namespace std;
```

који нам омогућава коришћење стандардних ствари у језику.

Идентификатори служе за означавање основних елемената програмског језика. То су

1. променљиве
2. симболи константе
3. типови података
4. функције

Идентификатори су ознаке (рецимо x, y, a, b, m...) или читав назив неке променљиве (prva_stranica, broj, prosek....). За идентификаторе користимо слова, бројеве, специјалне знаке или комбинацију свега.

Оно што је битно и о чему треба водити рачуна је да идентификатор не сме да почне бројем.

Такође идентификатор не сме да има размак у имену. Уколико је потребно да идентификатор садржи две речи у свом називу препоручљиво је користити доњу црту. (promenljiva_1, Ime_prezime, str_kocke...).

Посебно значајна чињеница је да програмски језик C++ разликује мала од великих слова. Тако су променљиве BGD и bgd различите.

Препорука је да име идентификатора не заузима више од 8 знакова.

Константе су вредности које не можемо мењати у току програма, већ оне од почетка до краја програма имају исту вредност. Константе у C++-у се дефинишу на следећи начин:

```
const tip podatka ime_konstante = vrednost
```

пример: Уколико желимо да дефинишемо константу Пи урадићемо на следећи начин:

```
const float Pi=3.14;
```

Уколико у току програма покушамо да променимо ову вредност јавиће нам се грешка приликом превођења програма.

Резервисане речи су одређени број енглеских речи које имају фиксно значење и које се користе у неке друге сврхе и не могу бити имена иденгификатора. Зову се још и **кључне речи** или **службене речи**. Све резервисане речи пишу се малим словима. Углавном представљају команде програма или типове података.

Ово је табела службених речи које се користе и које не могу бити искоришћене као имена промењљивих:

auto	**void**	**reinterpret_cast**	**static_cast**
const	**case**	**try**	**typeid**
double	**default**	**asm**	**catch false**
float	**enum**	**dynamic_cast**	**operator**
int	**goto**	**explicit**	**template**
short	**register**	**bool**	**wchar_t**
struct	**volatile**	**typename**	**public**
unsigned	**typedef**	**class friend**	**throw**
break	**char**	**private**	**virtual**
continue	**do**	**this**	**delete**
else	**extern**	**using**	**protected**
for	**if**	**const_cast**	**true**
long	**return**	**inline**	**new**
signed	**static**	**mutable**	**namespace**
switch	**union**		
sizeof	**while**		

Поред ових постоје и службене речи које се користе за рада са операторима, а то су следеће речи:

and	**compl**	**not**	**and_eq**
bitand	**not_eq**	**or**	**bitor**
or_eq	**xor_eq**	**xor**	

Основни типови података

У делу који се односи на идентификаторе неколико пута су поменути типови података. Типови података су уствари особине које описују неки података. У зависности од типа података мењају се:

1. вредности коју може да има неки податак
2. операције које се могу извршити над тим податком
3. начин представљања података у меморији.

Типови података могу бити прости и сложени. Сложени типови састоје се од више простих типова.

У програмском језику C++ у основне (просте) типове података спадају:

1. знаковни подаци (**char**)
2. цели бројеви (**int, long int**)
3. реални бројеви (**float, double, long double**)
4. набројиви тип (**enum**)
5. празан тип (**void**).

char – представља слово алфабета или неки специјални знак
int – представља цео број (4, 343, 59823)
long int – представља цео број али велике дужине
float – претстављају реалне бројеве (децимални зарез је обележен тачком 4.15, 65.879)
double – представљају реалне бројеве двоструке прецизности
long double – предстаљају реалне бројеве проширене прецизности

Као посебна група целобројних података јавља је тип **bool** где постоји аутоматска конверзија између типа **bool** и нумеричких типова. Када се користи у изразима логичка вредност **false** се претвара у целобројну вредност нула, а **true** у један. Ако се променљивој типа **bool** додељује нумеричка вредност, вредност нула се претвара у **false**, а било која ненулта вредност у **true**.

Оно што је изузетно битно је да или на почетку сваког програма или кад први пут користи неку променљиву програмер мора да декларише тип те променљиве. За сваку променљиву корисник мора да одреди тип и да је на тај начин обележи. Дакле, програмер мора да води рачуна о томе да уколико врши дељење неког броја или његово кореновање вероватно ће добити реални број. Па према томе ће да одреди који тип променљиве ће дефинисати за ту променљиву.

НАПОМЕНА: *програмски језик C++ прави грешку у томе што приликом дељења два цела броја даје као резултат цео број без остатка, па се препоручује да се бројеви који се деле увек декларишу као реални. То није случај са рецимо кореновањем или множењем једног реалног и једног целог броја.*

НАПОМЕНА: *мора да се дефинише свака променљива у програму. То се односи не само на улазне податке, него и на излазне и на помоћне променљиве у програму.*

Декларација променљивих врши се на следећи начин:

```
tip promenljive  imePromenljive;
```

наравно ако постоје више променљивих истог типа све могу да се наведу у једном реду или за сваку променљиву да се пише тип:

```
tip promenljiva1, prom_2, prom_n;

tip promenljiva1;
tip prom_2;
tip prom_n;
```

оно што је важно је да се наведу све променљиве и да се сваки ред заврши са тачка-зарез (;).

Треба нагласити и то да постоји могућност **иницијализације променљиве**. Иницијализација подразумева доделу поченте вредности тој променљивој. То се ради на следећи начин:

```
tip promenljiva1 = neka vrednost;
```

рецимо:

```
int s=0, a, b;
```

у овом реду декларисали смо три променљиве и све три су истог типа тј. цели бројеви. Само променљива s има почетну вредност која је нула.

```
int a=0, b=0, c, d;
```

овог пута декларисане су четири променљиве а две, а и б имају почетну вредност нула.

Идентификатор (симболичко име) служи за означавање основних елемената програмског језика:

- променљивих,
- константи,
- типови података,
- функције...

Формира се као низ малих и великих слова, цифара и доње црте. Доња црта се не препоручује као први знак, а почетак идентификатора никада не може бити број.

Исправно написани идентификатори су:

```
alfa
b01
povr
P1
strana_1
```

Неисправно написани иденитфикатори су:

`2strana` - број није дозвољен као почетни знак за идентификатор

`a-1` - знак минус није дозвољен јер се користи као математичка операција

`#pov` - знак # није дозвољен јер се користи за претпроцесорске директиве

`^strana` - знак није дозвољен

`$iznos` - знак није дозвољен

Да би се извршило уношење или штампање података у програмском језику C++ потребно је укључити библиотеку у оквиру претпроцесорских директива. Пише се на следећи начин и углавном представља почетак 99,9 % свих програма.

```
#include <iostream>
```

Питања:

1. Набројати карактеристике програмског језика C++.
2. Из којих симбола се састоји азбука програмског језика C++?
3. Шта је претпроцесор?
4. Којим знаком почињу директиве у C++-у?
5. Како се пишу коментари?
6. Шта је namespace?
7. Шта су идентификатори?
8. Како се дефинишу константе?
9. Шта су резревисане речи?
10. Који типови података постоје?
11. Објаснити како се декларишу подаци и о чему треба водити рачуна при декларацији.

Наредбе излаза и улаза

Приказ података

Наредба која се користи за излазне вредности је наредба *cout* и њена синтакса гласи:

```
cout<<''komentar ''<<promeljiva_1<< … << promenljiva_n<<endl;
```

коментар се пише у зависности од тога шта желимо да нам пише на екрану.

<< оператор уметања који се чита као „послато је“ или „узима се“. Оператор << има улогу оператора излаза само ако се са његове леве стране налази објекат cout.

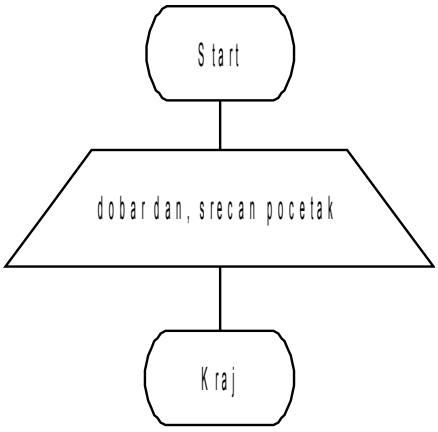
Пре назива промењљиве која се штампа поново се наводи оператор <<. Овај оператор се користи и између више промењљивих. (нпр. уколико имамо три промењљиве између сваке од њих наводимо овај оператор).

endl представља завршетак линије и скок у нови ред.

Уколико се у наредби за штампање користе више коментара између промењљивих то можемо написати на следећи начин:

```
cout<<''komentar ''<<promeljiva_1<< ''komentar ''<< promenljiva_2<<endl;
```

Пример: Одштампати поздравну поруку „добар дан, срећан почетак“.




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

int main() //pocetak programa
{
    cout << " Dobar dan!" << endl;
    return 0; //kraj programa
}
```

Унос података

У програмском језику C++ ова наредба за унос података је: cin.

Синтакса ове наредбе гласи:

```
cin >> промењљива >> промењљива;
```

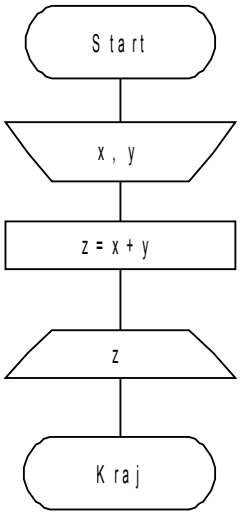
Оба оператора >> и << могу да се лакше схвате као стрелице које показују правац улаза и излаза. И да би имали ту улогу обавезно са леве стране мора да им стоји cin или cout.

Пример за наредбе уноса и приказа података

```
#include <iostream> /* standardno zaglavlje za naredbe ulaza i izlaza i operatore
>> i << */
 using namespace std; /* u ovaj prostor se stavljaju sva standardna zaglavlja koja
sadrze standardne bibliotecke f-je, globalne identifikatore kako bi mogla da se
koriste imena iz tog prostora. */
 int main()                  //glavni program
 {
  int a, b;
```

```
 char c;
 // unos se vrsi naredbom cin i navodjenjem operatora >>
 // prikaz se vrsi naredbom cout i navodjenjem operatora <<
 // za prelaz u novi red \n, a sada mozete koristiti i endl
 // za tabulaciju \t
 // za navode \"
 // za kosu crtu \\
 // itd.
 // primeri:
 cin >> a;
 cin >> b;
 cin >> c;
 cout << a << "\n";
 cout << a << " ovo je neki text " << b << endl;
 cout << "znak koji si uneo za c je " << c << endl;
 return 0;
}                              //kraj glavnog programa
```

Пример: Саставити алгоритам и програм за израчунавање збира два броја.




```
#include <iostream>
using namespace std;
int main ()
{
  int x, y;
  cout << "unesi dva broja" << endl;
  cin >> x >> y;
  int z=x+y;
  cout <<  "zbir je " << z << endl;
  return 0;
}
```

Оператори у C++-у

Оператор или знак операције је посебан симбол којим се означава нека радња тј. примена одређене операције над подацима. **Операнди** су подаци над којима се врши операција. У програмским језицима (не само у C++-у) под појмом оператор подразумева се примена операција која траје и даје коначно одређени резултат. Зато се уместо термина оператор користи и термин наредба. У језику C++ постоје три врсте наредби:

- **просте,**
- **сложене (наредбе управљања – гранање и циклуса) и**
- **декларативне које служе за дефинисање података**.

Израз је формула или правило израчунавања које при извршењу даје резултат.

Важно питање код израчунавања вредности израза је како објединити операнде и операторе. Зато постоји систем приоритета којим се одређује редослед примене оператора. Ова правила су слична као и у математици тј. множење и дељење имају предност над сабирањем, као што степеновање има предност над множењем.

Такође је врло битно водити рачуна о томе који тип податка се додељује операндима, због правила за конверзију података.

Додела вредности:

```
a= израз
```

где = нема исту фунцкију као у математици и не означава једнакост леве и десне стране, већ се операнду а додељује вредност (и тиме се мења његова претходна вредност) вредношћу коју има израз.

пример: **a=5+4**

тиме се операнду а додељује вредност 9.

У језику C++ могуће је употребити и додатне операторе доделе вредности. Пример **a+=б** је исто што и **a=a+б** или **c*=a+б** је исто што и **c=c*(a+б)**.

Можемо да пишемо и следећи израз: **a=b=c=x-3**, а извршава се следећим редоследом:

```
c=x-3,
b=c
a=b
```

ако x има вредност 8, резултат ће бити следећи:

```
c=5
b=5
a=5
```

Аритметички оператори:

* множење
/ дељење
% остатак целобројног дељења
+ сабирање
- одузимање

Кад смо рекли да мора да се води рачуна о томе који тип података се додељује операндима, имамо у виду следеће. Ако је с типа int, а имамо следеће ситуације:

```
c=7/5    // c=1
c=56/100 // c=0
```

тј. због типа податка одбацује се реални део броја и остаје само целобројно решење. Када приемњуемо дељење по модулу добијамо следеће резултате:

```
c=7%3         // c=1  tj. 7:3 je 2 и остатак je 1
```

Релациони оператори

су оператори поређења и врше нумеричко поређење два операнда а резултати тог поређења је истина (true) или лаж-неистина (false). Релациони оператори су следећи:

< мање
<= мање или једнако
> веће
>= веће или једнако
= = једнако
!= различито

Логички оператори:

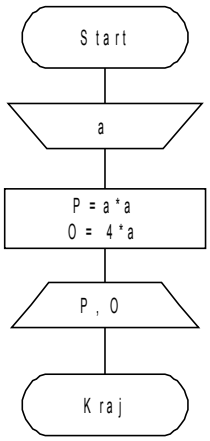
Постоје три логичка оператора и то су НЕ, И и ИЛИ оператори. Они имају следеће ознаке

! не
&& и
|| или

пример `a == б || a > б` значи ако је а једнако или веће од б.

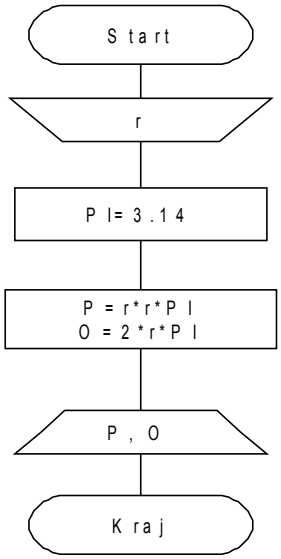
Програми линијске структуре

1. Саставити алгоритам за израчунавање површине и обима квадрата



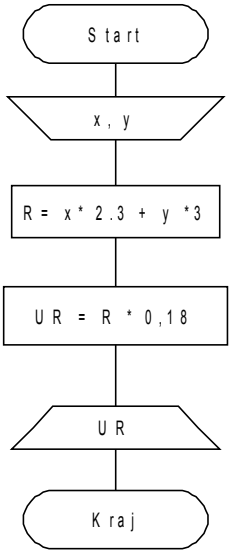

```
#include <iostream>
#include <cmath>// Sadrzi matematicke operacije
using namespace std;
int main()
{
int a, p, o;
cout<<"unesi stranicu"<<endl;
cin>>a;
p=pow (a, 2);
o=4*a;
cout<<"povrsina je "<<p<<"a obim je "<<o<<endl;
return 0;
}
```

2. Саставити алгоритам за израчунавање површине и обима круга



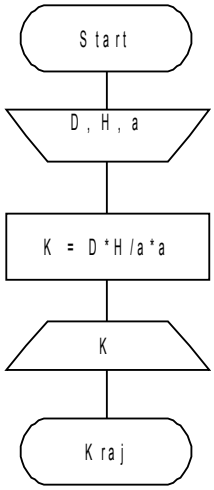

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int r;
float p, o;
const float pi=3.14
cout<<"unesi poluprecnik"<<endl;
cin>>r;
p=pow (r, 2)* pi;
o=2*r*pi;
cout<<"povrsina je "<<p<<endl;
cout<<"obim je "<<o<<endl;
return 0;
}
```

3. Потрошено је X киловата струје по цени од 2,3 динара и Y киловата струје по цени од 3 динара. На цео износ додаје се 18% пореза. Колико износи укупан рачун.



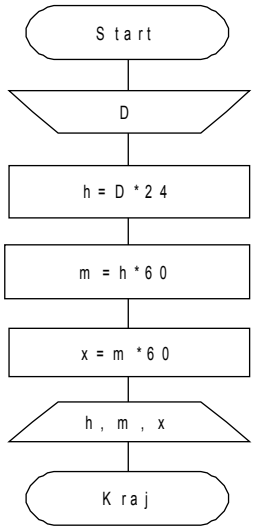

```
#include <iostream>
using namespace std;
int main()
{
int x, y;
float R, UR;
cout<<"unesi br kilovata"<<endl;
cin>>x>>y;

R=x*2.3+y*3;
UR=R+R*0.18;
cout<<"ukupan racun iznosi "<<UR<<endl;
return 0;
}
```

4. Зид дужине D и висине H треба облепити плочицама квадратног облика ивице a. Направити алгоритамску блок шему којом се узрачунава потребна количина плочица.



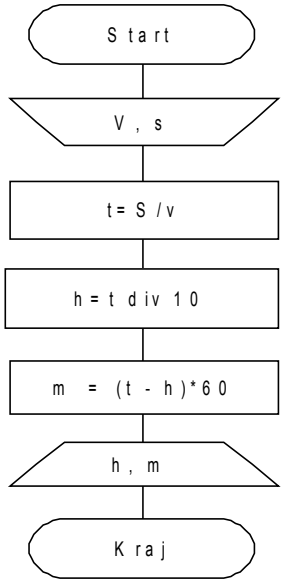

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
float D, H, a;
cout<<"unesi duzinu i visinu zida, kao i duzinu strane"<<endl;
cin>>D>>H>>a;
float K=(D*H)/pow(a,2);//promenljivu mozemo
//da definisemo tek kad je prvi put koristimo

cout<<" broj plocica je "<<K<<endl;
return 0;
}
```

5. Дат је број дана. Направити алгоритам за одређивање колико је то сата, минута и секунди.




```
#include <iostream>
using namespace std;
int main()
{
int D, h, m, x;
cout<<"unesi broj dana "<<endl;
cin>>D;
h=D*24;
m=h*60;
x=m*60;
cout<<"za dati broj dana to je "<<h<<" sati,"<< m<<
  " minuta i "<<x<<" sekundi"<<endl;
return 0;
}
```

6. Написати алгоритамску блок шему за израчунавање времена потребног да аутомобил крећући се брзином V km/h пређе пут S у км (време изразити у сатима и минутима)




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

{
float v, s, t;
int h, m;
cout<<"uneti brzinu i put "<<endl;
cin>>v>>s;
t=s/v;
h= t;
m=(t-h)*60;
cout<<"automobil se krece "<<h<<" sati, i "<<m<<
  " minuta"<<endl;
return 0;
}
```

7. Написати алгоритам којим ће се за два унета броја израчунати њихов количник.
8. Написати алгоритам којим ће се израчунати површина и запремина коцке.
9. Написати алгоритам којим ће се израчунати квадрат и куб броја унетог са тастатуре.

Програми разгранате структуре

Наредба гранања је наредба IF. Гранање може да буде једноструко, двоструко или вишеструко условно гранање.

Једноструко гранање има синтаксу:

```
if (услов)
{
наредбе на грани ДА;
}
```

пример: Унети број различит од нуле па приказати његову апсолутну вредност.

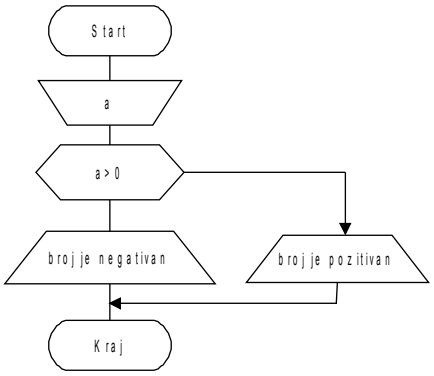
```
#include<iostream>
using namespace std;
int main()
{
     int a;
     cout<<"Uneti broj razlicit od nule"<<endl;
     cin>>a;
     if(a<0)
     {
         cout<<"Broj "<<a<<" je negativan. Apsolutna vrednost mu je "<<-a<<endl;
     }
     if(a>0)
     {
         cout<<"Broj "<<a<<" je pozitivan. Apsolutna vrednost mu je "<<a<<endl;
     }
return 0;
}
```

Двоструко гранање има синтаксу:

```
if (услов)
{
наредбе на грани ДА;
}
else
{
наредбе на грани НЕ;
}
```

То у пракси изгледа овако:

Написати алгоритам и програм којим се одређује да ли је унети број позитиван или не.




```
#include <iostream>
using namespace std;
int main()
{
int a;
cout<<"unesi broj razlicit od nule "<<endl;
cin>>a;
if (a>0)
cout<<"broj je pozitivan"<<endl;
else
cout<<"broj je negativan"<<endl;
return 0;
}
```

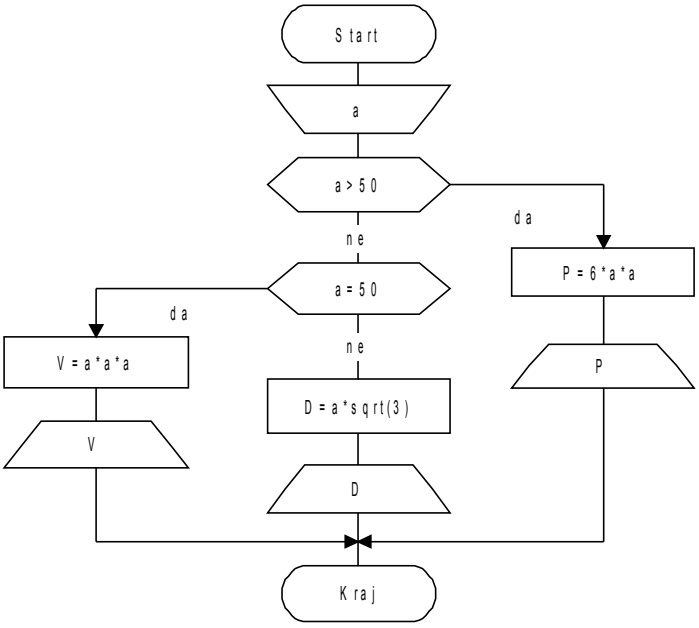
Уколико се на грани ДА или грани НЕ налази само по једна наредба онда витичасте заграде нису потребне, оне се користе само ако има више од једне наредбе било на грани да или грани не.

Вишеструко гранање има синтаксу:

```
if (услов1)
{
наредбе на грани ДА;
}
 else
{
наредбе на грани НЕ које представљају нови услов
if (услов2)
     {
     наредбе на грани ДА новог услова;
     }
     else
     {
     наредбе на грани НЕ новог услова;
     }
}
```

То у пракси изгледа овако:

Унети број. Уколико је он већи од 50 израчунати површину коцке, ако је 50 израчунати запремину коцке, а ако је мањи од 50 израчунати дијагоналу коцке.




```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int a, P, V;
float D;
cout<<"unesi stranicu kocke "<<endl;
cin>>a;
if (a>50)
   {
   P=6*pow(a,2);
   cout<<"povrsina je "<< P<<endl;
   }
   else
   {
    if (a == 50)
      {
      V=pow(a, 3);
      cout<<"zapremina je "<< V<<endl;
      }
      else
      {
      D=a*sqrt(3);
      cout<<"dijagonala je "<< D<<endl;
      }
   }
return 0;
}
```

Наредба SWITCH - CASE

Уколико имамо више услова у једном задатку, често је све много једноставније решити наредбом switch.

општи обилик ове наредбе је:

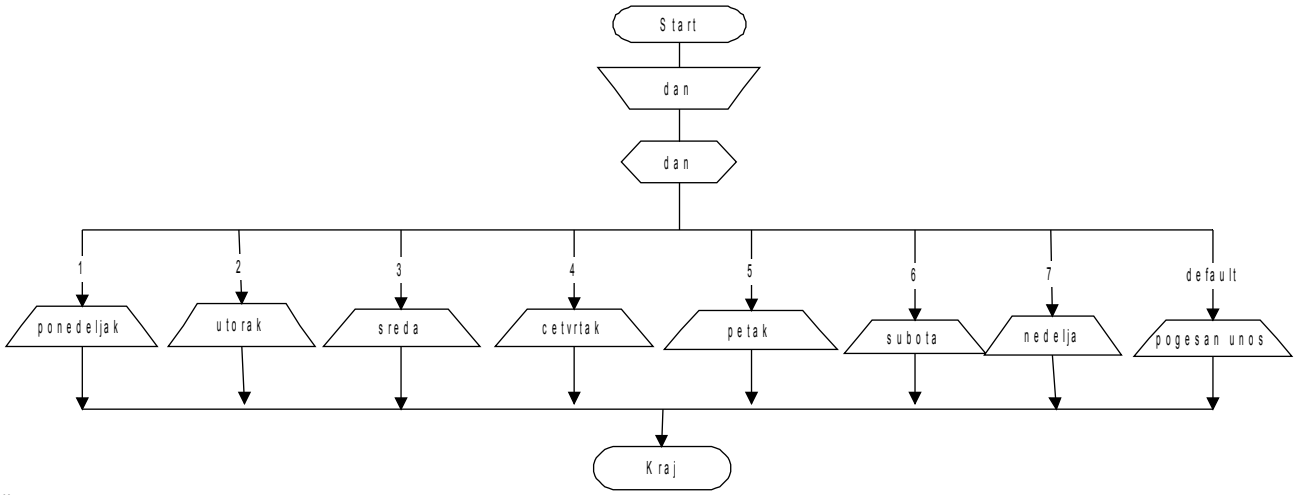
```
swich (izraz 1)
{
    case1 oznaka1: naredba1;
    case2 oznaka2: naredba2;
    ….
```

```
    default: naredba(n+1);
}
```

испитује се израз1, ако је он једнак ознаци1 онда се извршавају једна или више наредби из те гране, ако није испитује се да ли је израз1 једнак ознаци2, и поступак се понавља све док постоје кључне речи case.

Уколико ни једна ознака1 не одговара изразу1 прелази се на ред који у себи саджи кључну реч default. Ова кључна реч није обавезна и уколико не постоји наставља се са извршавањем прве наредбе после switch.

1. Написати програм у коме ће се унети неки број и уколико је тај број у распону између 1 и 7 на екрану треба да се појави назив дана у недељи који одговара том броју, у свим другим случајевима треба да се појави порука: Погрешан унос.




```
#include <iostream>
using namespace std;
int main()
{
int dan;
cout<<"unesi jedan broj"<<endl;
cin>>dan;
switch (dan)
       {
        case 1: cout<<"ponedeljak"<<endl;break;
        case 2: cout<<"utorak"<<endl;break;
        case 3: cout<<"sreda"<<endl;break;
        case 4: cout<<"cetvrtak"<<endl;break;
        case 5: cout<<"petak"<<endl;break;
        case 6: cout<<"subota"<<endl;break;
        case 7: cout<<"nedelja"<<endl;break;
        default: cout<<"pogresan unos"<<endl;
        }
return 0;
}
```

Наредбе безусловног гранања су, у ствари, наредбе скока на одређену линију програма. Наредбе безусловног гранања су break и goto.

Наредба **break** се најчешће користи приликом употребе switch наредбе. У другим случајевима није пожељна јер нарушава секвенцијални ток програма.

Наредба **goto** узрокује безусловни скок на тачно одређено место у програму. Место тј. ред у

програму од којег се наставља са извршењем програма дефинише се ознаком, лабелом. Лабела се ставља непосредно пре наредбе која је одредишно место скока. У општем случају синтакса ове наредбе гласи:

```
goto labela_1;

…

labela_1: naredba_1;
```

Овај начин решавања проблема није популаран, јер може да се изгуби контрола над програмом.

Кад год се користи било која од наредби безусловног скока потребно је све лепо и прецизно документовати.

Пример: Потребно је унети број различит од нуле. Уколико корисник унесе нулу вратити га на поновни унос податка:

```
#include<iostream>
using namespace std;
int main()
{
     int a;
upis:cout<<"Upisi broj razlicit od 0:";cin>>a;
     if(a==0)
     {
           cout<<"Pogresan unos. Ponovi!"<<endl;
           goto upis;
     }
     if(a<0)
     {
       .
       .
```

Наставак задатка је небитан може да се понови нека од претходних наредби, рецимо исписивање апсолутне вредности броја.

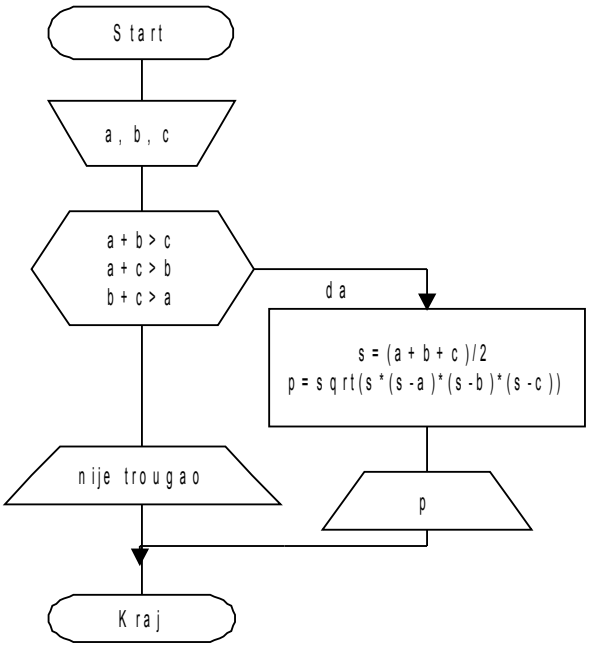
Примери:

1. Унети број. Ако је тај број јединица израчунати површину коцке, ако је унети број два израчунати површину квадра, а за било који други унети број исписати поруку о погрешно унетом броју. Користити наредбу switch – case.

```
#include <iostream>
using namespace std;
int main()
{
int a,b, c, p, n;
cout<<"unesi broj"<<endl;
cin>>n;
switch (n)
 {
case 1:
{
cout<<"unesi stranicu kocke"<<endl;
con>>a;
p=6*a*a;
cout<<"povrsina kocke je "<< p<<endl;
break;
}
case 2:
{
```

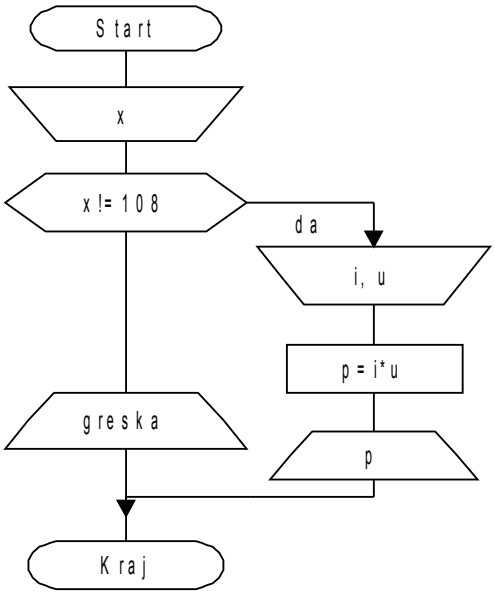
```
cout<<"unesi stranice kvadra"<<endl;
cin>>a>>b>>c;
p=2*(a*b+a*c+b*c);
cout<<"povrsina kvadra je "<< p<<endl;
break;
}
default: cout<<"unet je pogresan broj, pokusaj ponovo"<<endl;
}
return 0;
}
```

2. Напиши програм који врши проверу да ли унете дужине страница a, b и c могу формирати троугао. Уколико могу израчунати површину тог троугла.



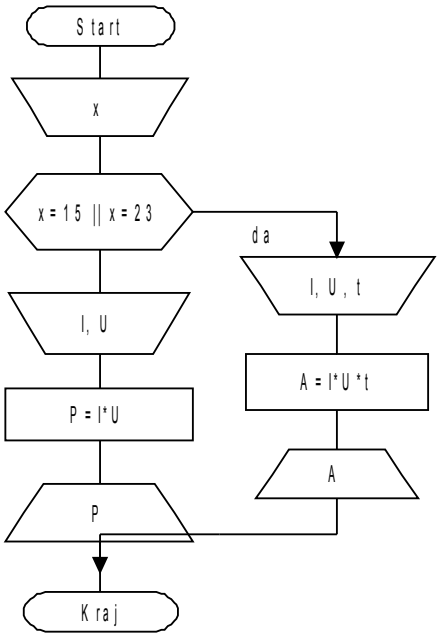

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
float a,b,c,p,s;
cout<<"Unesi stranice trougla "<<endl;
cin>>a>>b>>c;
if (a+b>c && a+c>b && b+c>a) //koristimo logicke operator "i"
{
s=(a+b+c)/2;
p=sqrt(s*(s-a)*(s-b)*(s-c));
cout<<"p= "<<p<<endl;
}
else
cout<<"ne moze se formirati trougao"<<endl;
return 0;
}
```

3. За вредност унету са тастатуре која није 108 израчунати снагу струје.



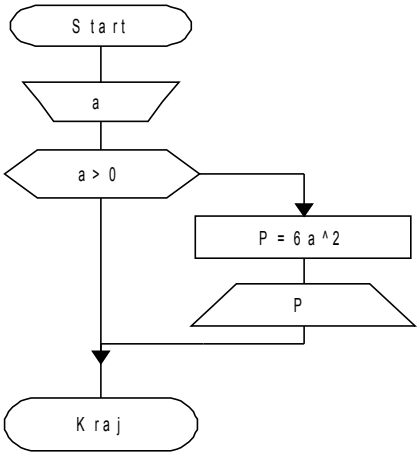

```
#include <iostream>
using namespace std;
int main()
{
int x, u, i, p;
cout<<"uneti broj"<<endl;
cin>>x;
if (x!=108)
{
cout<<"unesi struju i napon"<<endl;
cin>>i>>u;
p=i*u;
cout<<"snaga je "<< p<<endl;
}
else
cout<<"uneta vrednosti je pogresna"<<endl;
return 0;
}
```

4. Унети број. Уколико је он 15 или 23 израчунати рад електричне струје. У било ком другом случају израчунати снату електричне струје.



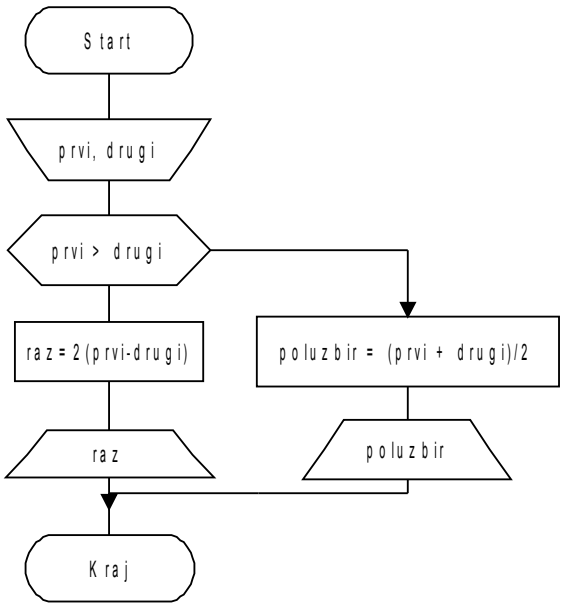

```
#include <iostream>
using namespace std;
int main()
{
int x, I, U, t, A, P;
cout<<"unesi neki broj"<<endl;
scin>>x;
if (x==15 || x==23)
   {
   cout<<"unesi jacinu, napon i vreme"<<endl;
   cin>>I>>U>>t;
   A=I*U*t;
   cout<<"Rad elektricne struje je "<< A<<endl;
   }
   else
   {
   cout<<"unesi jacinu i napon struje"<<endl;
   cin>>I>>U;
   P=I*U;
   cout<<"Snaga elektricne struje je "<< P<<endl;
   }
return 0;
}
```

5. Саставити алгоритам и програм којим ће се за унети позитиван број израчунати површина коцке.



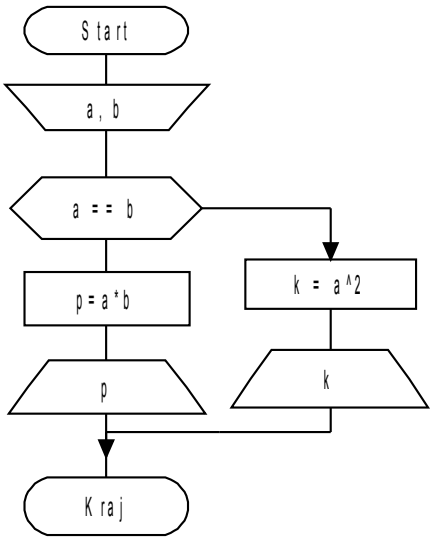

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int a, P;
cout<<"unesi duzinu stranice"<<endl;
cin>>a;
if (a>0)
{
P=6*pow (a, 2);
cout<<"povrsina je "<<P<<endl;
}
return 0;
}
```

6. Унети два броја, ако је први већи од другог израчунати њихов полузбир, а ако није израчунати њихову удвостручену разлику.




```
#include <iostream>
usnig namespace std;
int main()
{
float prvi, drugi, poluzbir, raz;
cout<<"unesi dva broja"<<;
cin>>prvi>>drugi;
if (prvi>drugi)
{
poluzbir= (prvi+drugi)/2;
cout<<"poluzbir je "<< poluzbir<<endl;
}
else
{
raz=2*(prvi-drugi);
cout<<"udovostrucena razlika je "<<
raz<<endl;
}
return 0;
}
```

7. Унети два броја. Ако су иста израчунати површину квадрата, а ако нису израчунати површину правоугаоника.




```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int a, b, p, k;
cout<<"unesi dva broja"<<endl;
cin>>a>>b);
if (a == b)
{
k=pow(a, 2);
cout<<"povrsina kvadrata je "<<k<<endl;
}
else
{
p=a*b;
cout<<"povrsina pravougaonika je "<<p<<endl;
}
return 0;
}
```

8. Израчунати запремину коцке ако је унети број позитиван.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int a,v;
cout<<"uneti broj"<<endl;
cin>>a;
if (a>0)
{
v=pow (a, 3);
cout<<"zapremina kocke je "<<v<<endl;
}
return 0;
}
```

9. Унети три броја па одредити који је највећи

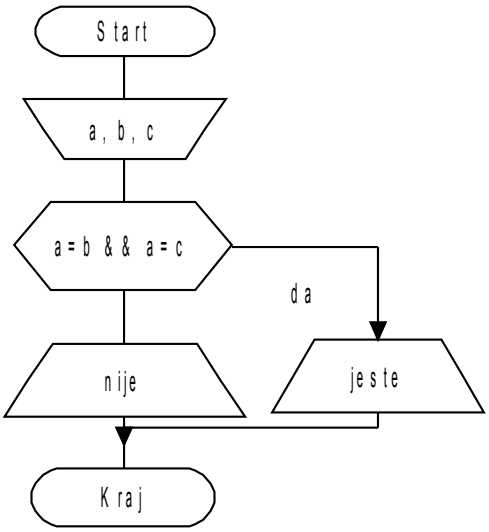
```
#include <iostream>
using namespace std;
int main()
{
int a, b, c, max;
cout<<"unesi tri broja"<<endl;
cin>>a>>b>>c;
if (a>b) max=a;
else max=b;
if (c>max) max=c;
cout<<"najveci broj je "<< max<<endl;
return 0;
}
```

10. Унети три стране троугла па испитати да ли је он правоугли.

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

```
int main()
{
int a, b, c, D;
cout<<"unesi tri broja"<<endl;
cin>>a>>b>>c;
D=sqrt (pow(a,2)+ pow(b,2));
if (c!=D)
cout<<"trougao nije pravougli"<<endl;
else
cout<<"trougao je pravougli"<<endl;
return 0;
}
```

11. Унети три броја па испитати да ли они могу да буду странице једнакостраничног троугла.




```
#include <iostream>
using namespace std;
int main()
{
int a, b, c;
cout<<"unesi tri broja"<<endl;
cin>>a>>b>>c;
if (a==b && a==c)
cout<<"trougao je jednakostranican "<<endl;
else
cout<<"trougao nije jednakostranican "<<endl;
return 0;
}
```

12. Купац у продавници је пазарио X производа по цени од C динара. Уколико је његов рачун већи од 5.000 динара, продавац му даје попуст од 2%. Колико износи рачун купца.

```
#include <iostream>
using namespace std;
int main()
{
int x, c, R;
cout<<"uneti broj proizvoda i cenu po komadu"<<endl;
cin>>x>>c;
R=x*c;
if (R>5000)
{
float popust=R*0.02;
float UR=R-popust;
cout<<"ukupan racun iznosi "<< UR<<endl;
}
else
cout<<"racun iznosi "<< R<<endl;
return 0;
}
```

13. Унети број. Уколико је он различит од броја 25 израчунати обим круга.

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

```
int n, r;
float o;
const float pi= 3.14;
cout<<"unesi broj"<<endl;
cin>>n;
if (n!=25)
{
cout<<"unesi poluprecnik kruga"<<endl;
cin>>r;
o=2*r*pi;
cout<<"obim je "<< o<<endl;
}
else
cout<<"error!!! uneti broj je 25"<<endl;
return 0;
}
```

14. Треба израчунати укупан отпор за отпоре R1 и R2 зависно од тога да ли веза између њих серијска или паралелна. Користити switch-case.

```
#include<iostream>
using namespace std;
int main()
{
float R,R1,R2;
int i;
cout<<"Otpor R1 (u omima):";
cin>>R1;
cout<<"Otpor R2 (u omima):";
cin>>R2;
cout<<"Za serijsku vezu upisi 1, a za paralelnu 2:";
cin>>i;
switch (i)
{
case 1:
R=R1+R2;
cout<<" Ukupni je otpor serijske veze je: " <<R<<" oma."<<endl;
break;
case 2:
R=(R1*R2)/(R1+R2);
cout<<" Ukupni otpor paralelne veze je " <<R<<" oma."<<endl;
break;
default:
cout<<"Pogresan unos. Unesi 1 ili 2"<<endl;
}
return 0;
}
```

15. Унети износ рачуна за струју. Ако је он већи од 10.000 додати на тај рачун камату од 15%. Штампати износ који треба да се плати.
16. Унети број. Испитати да ли је реч о троцифреном броју.
17. Унети два броја. Ако је први већи од другог израчунати површину и обим правоугаоника. Ако су исти израчунати површину и обим круга, а ако је други већи од првог израчунати њихов збир, разлику, количник и производ та два броја.
18. Унети број. Ако је тај број 1 израчунати квадрат тог броја, ако је 2 израчунати куб тог броја, ако је 3 израчунати корен тог броја, а ако је 4 спеновати га са самим собом. У случају да се унесе било који други број исписати поруку о грешци. Користити switch-case наредбу.
19. Потрошено је L литара воде по цени од K динара по литру. Израчунати вредност рачуна за воду. Уколико је он већи од 1.500 динара додати камату од 20%. Штампати укупан износ рачуна за воду.
20. Унети број па испитати да ли се он налази у интервалу од 50 до 75.

21. Унети број. Ако је он позитиван израчунати његов корен, ако је нула штампати је, а ако је негативан израчунати његов квадрат.
22. Унети број. Ако је тај број 1 израчунати укупну отпорности у колу где су четири отпорника везана редно. Ако је тај број 2 израчунати укупну отпорност у коју где су два отпорника везана паралелно. Ако је унет било који други број исписати поруку о грешци. Користити switch-case наредбу.

23. Чоколада у продавници кошта x динара. Донета је одлука да ако је цена већа од 150 динара та чоколада појефтини 10%. Штампати цену чоколаде.
24. Унети број. Ако је он негативан или већи од 100 израчунати његов квадрат.
25. Унети број. Ако је он 1 израчунати површину круга, ако је 2 израчунати обим круга, ако је 3 израчунати дужину пречника круга. За било који други број исписати поруку о грешци. Користити switch-case наредбу.
26. Дневно у некој фабрици је x производа. Цена једног производа је s динара. Уколико број произведених производа буде већи од 14.000 цена по производу се смањује за 5%. Колика је укупна цена производа за дан 18.04.2010?
27. Унети број. Уколико је он двоцифрен штампати његову дуплирану вредност.
28. Унети два броја. Ако су исти израчунати износ месечне рате за кредит (годишњи кредит је 38.000 динара). Ако је први број већи од другог израчунати полугодишњи износ рате за кредит. Ако је други број већи израчунати тромесечни износ рате за кредит.
29. Унети број. Ако је он 1 изрчунати напон на примару трансформатора, ако је 2 израчунати напон на секундару трансформатора. Ако је унети број 3 израчунати број намотаја на примару, а ако је 4 израчунати број намотаја на секундару. За било који други број исписати поруку о грешци. Користити switch-case наредбу.

Програми цикличне стуктуре

У општем случају, петља је управљачка структура која омогућава понављање одређене наредбе или групе наредби произвољан број пута. Свака петља садржи бар две целине:

- услов који се испитује и
- тело петље.

У зависности од услова тело петље извршава се одређени број пута. У језику С петље су подржане резервисаним речима: for, while и do-while.

For се користи за циклус са коначним бројем понављања и тај број се унапред зна.

While се користи за циклус са неодређеним бројем понављања и извршава се све док је услов испуњен. Услов је увек на врху, тј. пре тела петље.

Do-while се користи за циклус који се извршава неодређени број пута, тј. све док је услов испуњен, али се овај услов налази на дну, тј. на крају петље.

Општи облик наредбе For је следећи:

```
for (izraz1; izraz2; izraz3)
{
naredba_1;
…
naredba_n;
}
```

Део програма између витичастих заграда је тело програма и може да се састоји од једне или више наредби. Уколико се тело петље састоји од само једне наредбе витичасте заграде се не употребљавају. Оне се користе само ако има више од једне наредбе у телу петље.

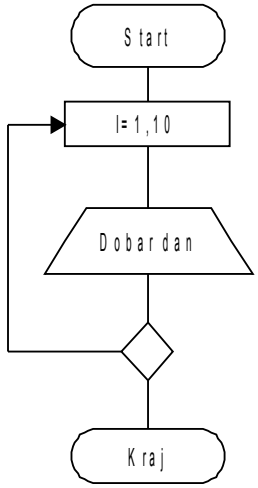
Израз1 представља почетну вредност петље.

Израз2 представља крајњу вредност петље која се испитује кроз одређени услов.

Израз3 представља промену вредности бројача, која је начешће 1 али може да буде и било која друга вредност.

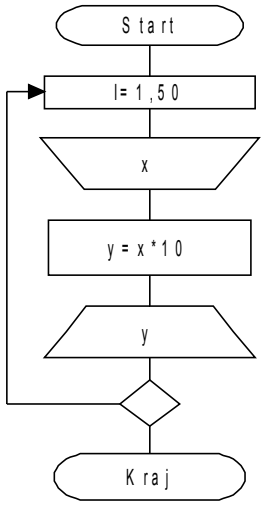
Пример:

1. Саставити алогритам и програм којим ће се десет пута штампати реченица: Добар дан.




```
#include <iostream>
using namespace std;
int main()
{
for (int i=1; i<=10; i++)
printf ("Dobar dan!\n");
return 0;
}
```

2. Саставити алгоритам и програм којим ће се унети 50 различитих бројева, а онда штампати њихова десетострука вредност.




```
#include <iostream>
using namespace std;
int main()
{
int x, y;
for (int i=0; i<50; i++)
{
cout<<"unesi broj"<<endl;
cin>>x;
y=x*10;
cout<<"deset puta uvecani broj je "<< y<<endl;
}
return 0;
}
```

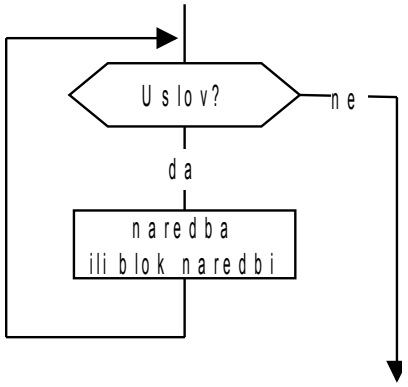
While наредба

Наредба FOR се користи код циклуса који имају унапред одређен број понављања. Међутим, некада је потребно користити циклусе чији се број понављања не зна унапред. Тада се користе нарбе while и do-while.

У првом случају услов је на почетку циклуса, а у другом услов је на крају циклус.

Примена наредбе while се врши тако што се најпре испита услов и онда уколико је он испуњем спроводе се наредбе у дело. Онда када услов више није испуњен искаче се из петље и наставља са извршењем задатка.

графички приказ наредбе while је следећи:



а синтакса наредбе гласи:

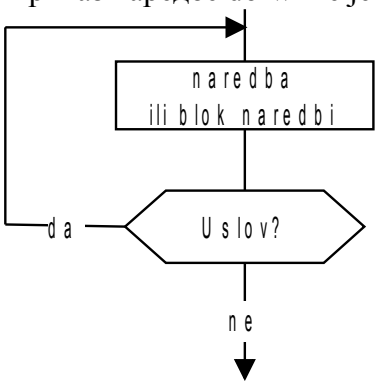
```
while (izraz)
{
     naredba_1;
     naredba_2;
}
```

Пример: Написати програм који ће штампати све непарне бројеве мање од 50.

```
#include <iostream>
using namespace std;
int main()
{
int n=1;
while (n<50)
{
cout<<n<<endl;
n=n+2;
}
return 0;
}
```

Уколико се наредба најпре изврши, па се тек онда испитује услов који означава прекид петље говоримо о do-while циклусу.

графички приказ наредбе do-while је следећи:



а синтакса наредбе гласи:

```
do
{
  naredba_1;
  naredba_2;
}
while (izraz);
```

1. Написати програм који ће сабрати све природне бројеве док збир не буде већи од 200.

```
#include <iostream>
using namespace std;
int main()
{
int s=0, n=1;
do
{
s=s+n;
n++;
}
while (s+n<200);
cout<<"zbir je "<<s<<endl;
return 0;
}
```

Примери:

1. Написати програм којим ће се израчунати сума првих десет бројева.

```
#include <iostream>
using namespace std;
int main()
{
int s=0;
for (int i=0; i<10; i++)
s+=i;
cout<<"suma je "<< s<<endl;
return 0;
}
```

2. Написати програм којим ће се израчунати сума десет бројева унетих са тастатуре.

```
#include <iostream>
using namespace std;
int main()
{
int s=0, n, x;
for (int i=0; i<10; i++)
{
cout<<"unesi broj"<<endl;
cin>>x;
s+=x;
}
```

```
cout<<"suma je "<< s<<endl;
return 0;
}
```

3. Написати програм којим ће се израчунати сума првих Н бројева.

```
#include <iostream>
using namespace std;
int main()
{
int s=0, n;
cout<<"unesi broj do kog se vrsi sabiranje"<<endl;
cin>>n;
for (int i=0; i<n; i++)
s+=i;
cout<<"suma je "<< s<<endl;
return 0;
}
```

4. Написати програм којим ће се израчунати сума Н бројева унетих са тастатуре.

```
#include <iostream>
using namespace std;
int main()
{
int s=0, n, x;
cout<<"unesi broj N"<<endl;
cin>>n;
for (int i=0; i<n; i++)
{
cout<<"unesi broj "<<endl;
cin>>x;
s+=x;
}
cout<<"suma je "<< s<<endl;
return 0;
}
```

5. Написати програм којим ће се израчунати просечна оцена за ученика који има 14 предмета.

```
#include <iostream>
using namespace std;
int main()
{
int i, x;
float s=0, po;
for (i=0; i<14; i++)
{
cout<<"unesi ocenu"<<endl;
cin>>x;
s+=x;
}
po=s/14;
cout<<"suma je "<< po<<endl;
return 0;
}
```

6. Написати програм којим ће се израчунати аритметичка средина за Н бројева унетих са тастатуре.

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

```
int main()
{
int i, n, x;
float as, s=0;
cout<<"unesi broj N"<<endl;
cin>>n;
for (i=0; i<n; i++)
{
cout<<"unesi broj "<<endl;
cin>>x;
s+=x;
}
as=s/n;
cout<<"aritmeticka sredina je "<< as<<endl;
return 0;
}
```

7. Израчунати производ првих 8 бројева.

```
#include <iostream>
using namespace std;
int main()
{
int p=1;
for (int i=1; i<=8; i++)
p=p*i;
cout<<"proizvod prvih 8 brojeva je "<< p<<endl;
return 0;
}
```

8. Израчунати годишњи износ рачуна за струју ако се зна да се сваког месеца потроши x киловата јефтиније струје по цени од 6 динара и p киловата скупље струје по цени од 9 динара. На целокупни рачун се обрачунава ПДВ од 18%.

```
#include <iostream>
using namespace std;
int main()
{
int x, r, s=0;
float gr;
for (int i=0; i<12; i++)
{
cout<<"unesi mesecnu potrosnju za skupu i jeftinu struju za "<<i+1<< "
mesec"<<endl;
cin>>x>>r;
s=s+(x*6+r*9);
}
gr=s+s*0.18;
cout<<"godisnji racun iznosi "<< gr<<endl;
return 0;
}
```

9. Унети дужину стране за 18 различитих коцки. Израчунати и приказати површину и запремину за све коцке и израчунати просечне површину и запремину.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
int a, p, v;
float prp, prv, s=0, b=0;
```

```
for (int i=0; i<18; i++)
{
cout<<"unesi duzinu stranice za "<<i+1<<". kocku"<<endl;
cin>>a;
p=6*pow(a,2);
v=pow(a,3);
s=s+p;
b=b+v;
cout<<"povrsina je "<< p<<endl;
cout<<"zapremina je "<< v<<endl;
}
prp=s/18;
prv=b/18;
cout<<"prosecna povrsina je "<<prp<<", a prosecna zapremina je "<< prv<<endl;
return 0;
}
```

10. У одељењу је 24 ученика. За сваког унети висину па израчунати просечну висину у одељењу и одредити колико ученика је више од 168 центиметра.
11. Сваког дана једна трафика прода К различите дневне новине. Унети количину продатих новина за сваког издавача. Израчунати просечну продају новина, као и просечну продају за новине чији је тираж испод 500 броја дневно и просечну продају за новине чији је дневни тира 500 и више броја дневно.

12. Израчунати просечну вредности унети бројева. Бројеве уности све док се не унесе нула. Користити while наредбу.

```
#include <iostream>
usnig namespace std;
int main()
{
float b=0, s, a, p;
cout<<"unesi broj"<<endl;
cin>>a;
s=a;
while (a!=0)
{
b++;
s=s+a;
cout<<"unesi broj"<<endl;
cin>>a;
}
p=s/b;
cout<<"prosecna vrednost je "<< p<<endl;
return 0;
}
```

13. Израчунати суму бројева који се уносе са тастатуре. Бројеве уносити све док се не унесе број 5. Користити while наредбу.

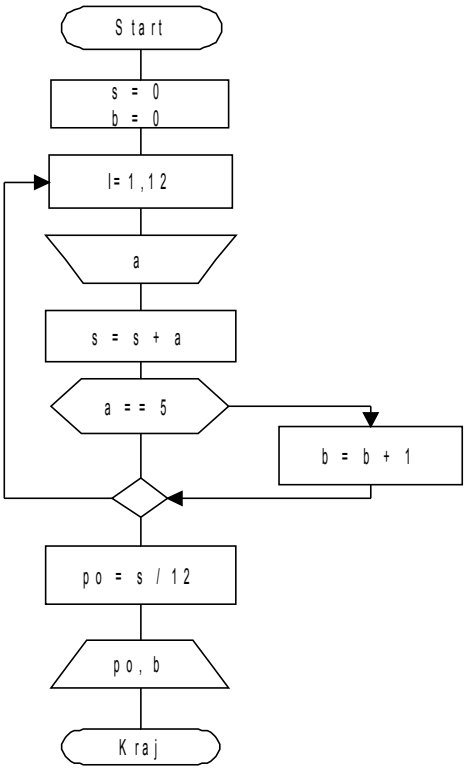
```
#include <iostream>
using namespace std;
int main()
{
int s=0, a, p;
cout<<"unesi broj"<<endl;
cin>>a;
while (a!=5)
{
s=s+a;
cout<<"unesi broj"<<endl;
cin>>a;
```

```
}
cout<<"suma unetih brojeva je "<< s<<endl;
return 0;
}
```

14. Саставити алгоритам и програм којим ће се унети N бројева, а затим избројати колико њих је негативно.
15. Унети температуру за 15 различитих градова па израчунати просечну температуру за све градове.

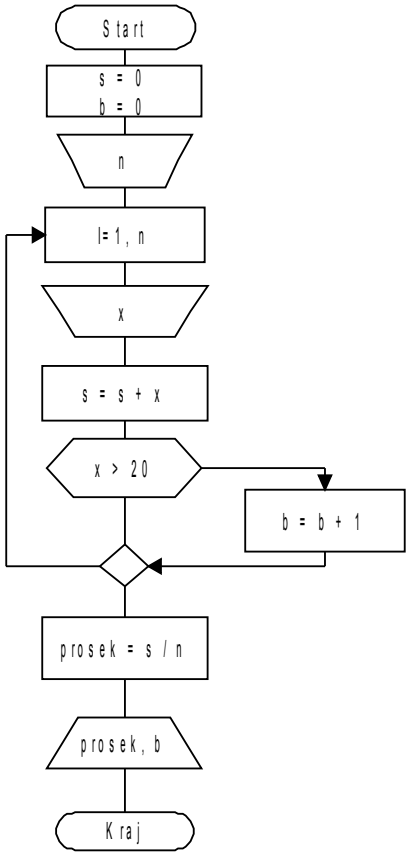
Програми са комбинованом структуром

1. Написати програм којим ће се за једног ученика унети оцене из 12 предмета и на основу њих израчунати просечна оцена и утврдити број петица.




```
#include <iostream>
using namespace std;
int main()
{
int b=0, a;
float s=0, po;
for (int i=0; i<12; i++)
{
cout<<"unesi ocene"<<endl;
cin>>a;
s=s+a;  /* ili s+=a */
if (a==5) b++;  /* ili b=b+1*/
}
po=s/12;
cout<<"prosecna ocena ucenika je " << po <<
  ", a broj petica je "<< b<<endl;
return 0;
}
```

2. Унети N бројева са тастатуре па израчунати њихову просечну вредност, као и колико је бројева веће од 20.




```
#include <iostream>
using namespace std;
int main()
{
int b=0, i, x, n;
float s=0, prosek;
cout<<"unesi broj"<<endl;
cin>>n;
for (i=0; i<n; i++)
{
cout>>"unesi "<<i+1<<" broj"<<endl;
cin>>x);
s=s+x;
if (x>20) b++;
}
prosek=s/n;
cout<<"prosecna vrednost je"<<prosek<<" a "<<
  b<<" brojeva je vece od dvadeset"<<endl;
return 0;
}
```

3. У фабрици је 200 радника. За сваког унети плату за месец мај па израчунати просечну плату у фабрици и одредити број радника чија је плата већа од 30.000 динара.

```
#include <iostream>
using namespace std;
int main()
{
int b=0, i, a;
float s=0, po;
for (i=0; i<200; i++)
{
cout<<"unesi platu za radnike"<<endl;
cin>>a;
s=s+a;
if (a>30000) b++;
}
po=s/200;
cout<<"prosecna plata je "<<po <<" a broj radnika sa vecom platom je "<< b<<endl;
return 0;
}
```

4. 20 радника свакодневно хвата одређену количину рибе. Избројати колико радника ухвати мање од 200 кг. рибе дневно, а за раднике који ухвате више од 200 кг. израчунати просечан улов.

```
#include <iostream>
using namespace std;
int main()
{
int b=0, k=0, i, a;
float s=0, po;
for (i=0; i<20; i++)
{
```

```
  cout<<"unesi ulov"<<endl;
  cin>>a;
  if (a>200)
   {
   k++;
   s=s+a;
   }
   else
   b++;
  }
  po=s/k;
  cout<<"sa manje od 200 kg je: "<<b<<" radnika, a prosecan ulov ostalih je "<<
po<<endl;
  return 0;
  }
```

5. Унети H бројева са тастатуре па одредити колико њих је негативно.

```
  #include <iostream>
  using namespace std;
  int main()
  {
  int b=0, i, x, n;
  float s=0;
  cout<<"unesi broj"<<endl;
  cin>>n;
  for (i=0; i<n; i++)
  {
  cout<<"unesi broj"<<endl;
  cin>>x;
  s=s+x;
  if (x<0)
  b++;
  }
  cout<<"uneto je "<<b<<" negativnih brojeva"<<endl;
  return 0;
  }
```

6. Унети температуру за H дана у једном граду. Одредити просечну температуру и број дана у којима је температура била испод нуле.

```
  #include <iostream>
  using namespace std;
  int main()
  {
  int b=0, i, x, n;
  float s=0, prosek;
  cout<<"unesi broj dana"<<endl;
  cin>>n;
  for (i=0; i<n; i++)

  {
  cout<<"unesi temperaturu za"<<i+1<<". dan"<<endl;
  cin>>x;
  s=s+x;
  if (x<0)
  b++;
  }
  prosek=s/n;
  cout<<"prosecna temperatura je "<<prosek<<" a "<<b<<" dana je bila ispod
nule"<<endl;
  return 0;
  }
```

Питања:

1. Коју наредбу користимо за приказ података?
2. Написати како гласи синтакса наредбе за приказ података.
3. Коју наребу користимо за унос података?
4. Написати синтаксу наредбе за унос података.
5. Шта су оператори, а шта су операнди?
6. Набројати аритметичке операторе.
7. Набројати релационе операторе.
8. Које наредбе користимо за разгранте програме?
9. Која је разлика између једноструког, двоструког и вишеструког гранања?
10. Написати синтаксу наредбе if.
11. Чему служи наредба switch-case?
12. Које наредбе користимо за прекид програма?
13. Објаснити наредбу break.
14. Објаснити наредбу goto.
15. Које наредбе користимо за цикличне структуре?
16. Објаснити разлику између while и do-while наредбе.
17. Написати синтаксу наредбе for.