

STRUKTURA GRANANJA

PROGRAMSKA STRUKTURA GRANANJA

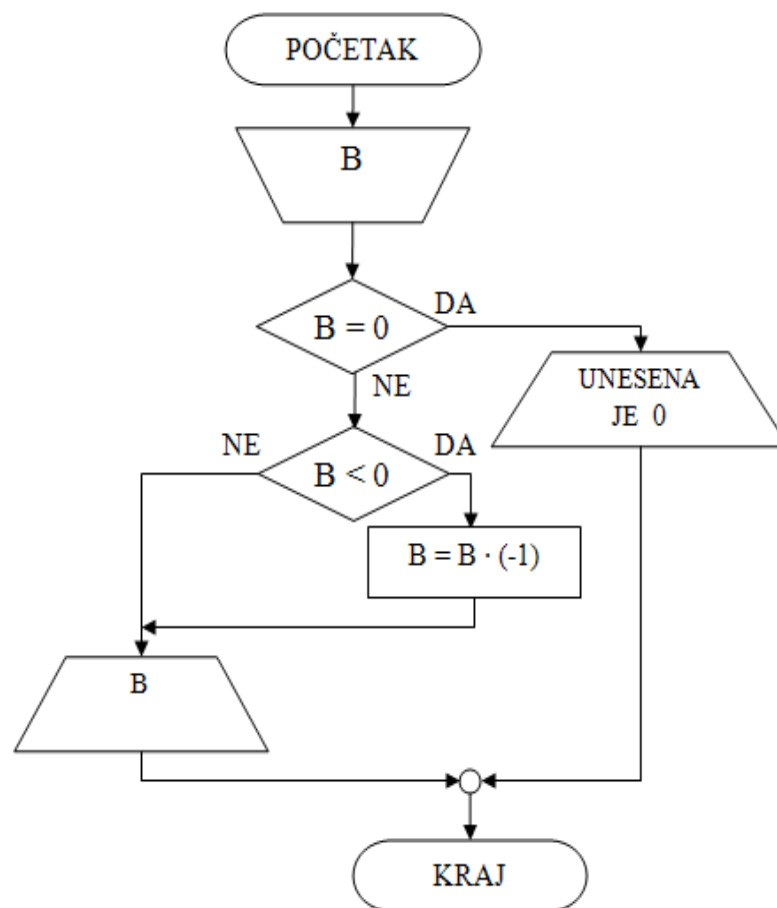
- za rješavanje većine zadataka potrebne su programske strukture kod kojih redoslijed izvršavanja naredbi ovisi o vrijednostima podataka koji se obrađuju
- grananje je programska struktura koja omogućuje različit tijek programa, ovisno o rezultatu postavljenog uvjeta



PROGRAMSKA STRUKTURA GRANANJA

Primjer programske strukture grananja:

- korisnik unosi cijeli broj, a zatim se računa apsolutna vrijednost tog broja i ispisuje rezultat



BLOK NAREDBI

- dijelovi programa koji se uvjetno izvode grupiraju se u blokove naredbi
- blok naredbi se omeđuje parom vitičastih zagrada, zbog preglednosti piše se uvučeno

```
{  
    int x;  
    cout<<endl<<"Upisi cijeli broj u bloku:";  
    cin>>x;  
}
```

Zagrade se mogu izostaviti ako se blok sastoji od jedne naredbe.



LOKALNE VARIJABLE

- varijable deklarirane unutar bloka naredbi nazivaju se lokalne varijable
- ako se varijable deklariraju unutar bloka, postoje samo unutar bloka u kome su deklarirane, u glavnoj funkciji one ne postoje



PRIMJER 1

Primjer lokalne varijable:

Treba deklarirati cjelobrojnu varijablu x unutar zasebnog bloka, pa joj pridružiti vrijednost.

Pokušati ispisati vrijednost varijable x u glavnoj funkciji.

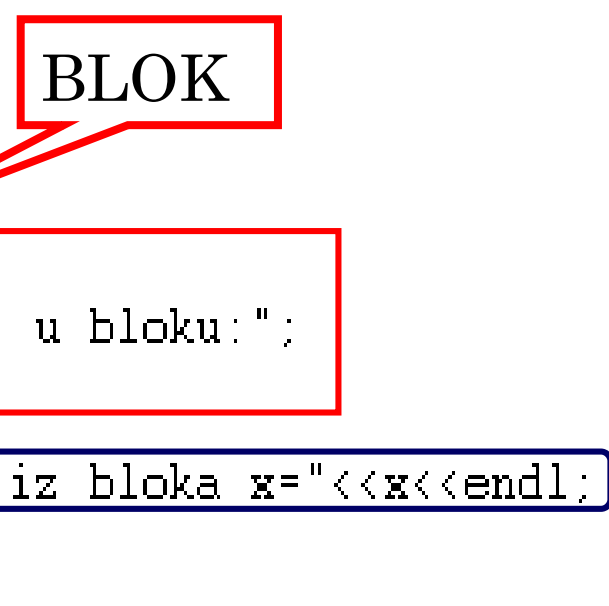
Upisi cijeli broj u bloku:

Vrijednost varijable iz bloka $x=...$



PRIMJER 1

```
#include<iostream>
using namespace std;
int main()
{
    {
        int x;
        cout<<endl<<"Upisi cijeli broj u bloku:";
        cin>>x;
    }
    cout<<endl<<"Vrijednost varijable iz bloka x="<<x<<endl;
    return 0;
}
```



BLOK

PRIMJER 1

- pri pokušaju prevođenja javit će se pogreška

```
Compiling...  
x.cpp  
  \x.cpp(10) : error C2065: 'x' : undeclared identifier  
Error executing cl.exe.  
x.exe - 1 error(s), 0 warning(s)
```

- varijabla deklarirana unutar bloka vidljiva je samo unutar tog bloka
- u glavnoj funkciji varijabla x ne postoji, zato se pri prevođenju javlja pogreška



NAREDBE GRANANJA

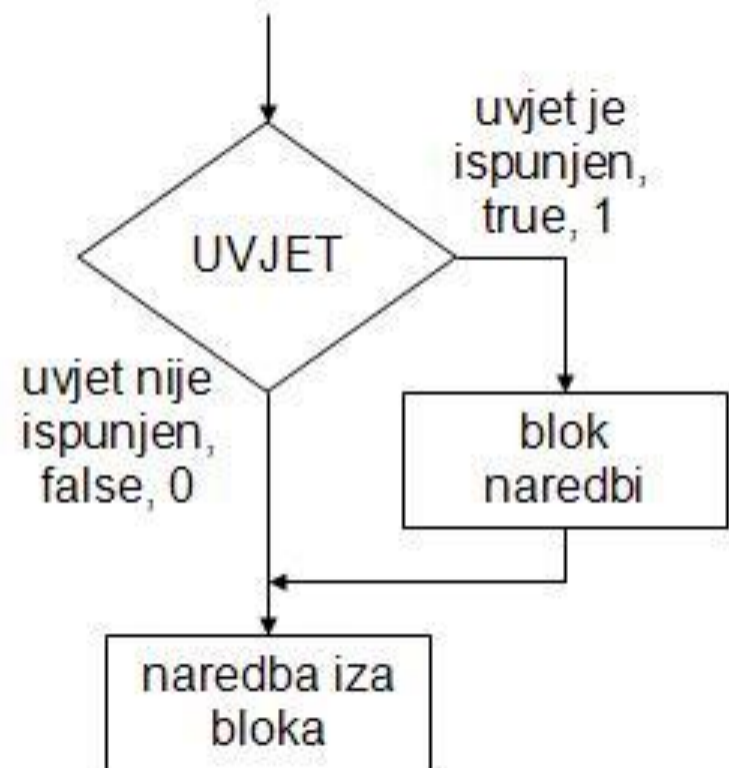
Programska struktura grananja može se ostvariti naredbama:

- *if*
- *if – else*
- *if – else if - else*
- *switch – case*



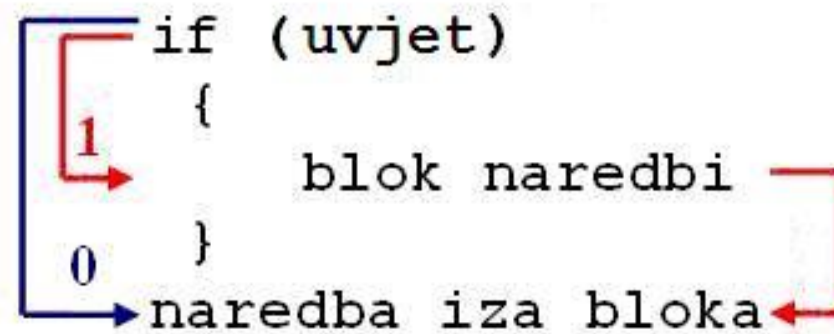
JEDNOSTRUKO UVJETNO GRANANJE

- jednostruko uvjetno grananje omogućava izvršenje bloka naredbi samo ako je zadani uvjet ispunjen
- ako uvjet nije ispunjen izvršava se prva naredba nakon bloka



JEDNOSTRUKO UVJETNO GRANANJE - IF

- za jednostruko uvjetno grananje rabi se naredba if



- uvjet je logički izraz, zapisuje se unutar para okruglih zagrada
- na kraju naredbe if ne stavlja se znak ;



PRIMJER 2

Treba unijeti cijeli broj različit od 0 pa provjeriti da li je negativan ili pozitivan. U oba slučaja ispisati apsolutnu vrijednost broja.

Ispis neka bude oblika:

Upisi cijeli broj razlicit od 0:
Broj...je.... Njegova apsolutna vrijednost je....



PRIMJER 2

- ako je ($a < 0$) izvršit će se prvi blok naredbi
- ako uvjet nije zadovoljen, prvi blok naredbi se preskače i izvođenje se programa nastavlja od prve naredbe iza bloka, a to je provjera drugog uvjeta ($a > 0$)
- ako nije ispunjan niti drugi uvjet (za $a = 0$), drugi blok naredbi se preskače i izvođenje se nastavlja od naredbe `return 0`



PRIMJER 2

```
#include<iostream>
using namespace std;
int main()
{
    int a;
    cout<<"Upisi cijeli broj razlicit od 0: ";
    cin>>a;
    if(a<0)
    {
        cout<<"Broj "<<a<<" je negativan. \
        Njegova apsolutna vrijednost je "<<-a<<endl;
    }
    if(a>0)
    {
        cout<<"Broj "<<a<<" je pozitivan. \
        Njegova apsolutna vrijednost je "<<a<<endl;
    }
    return 0;
}
```

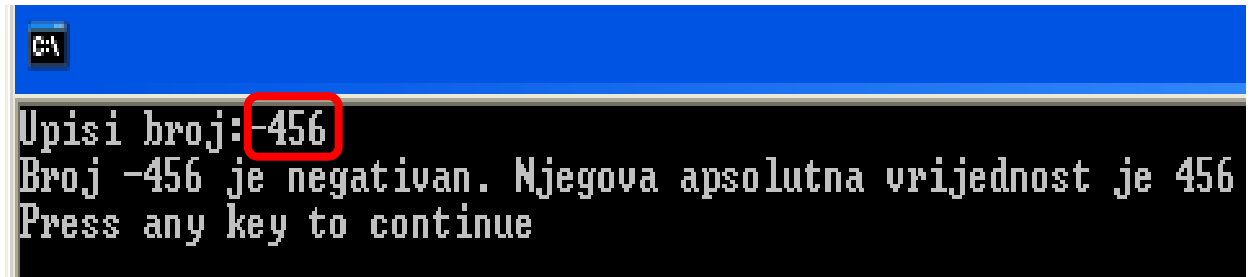
1. BLOK

2. BLOK



PRIMJER 2

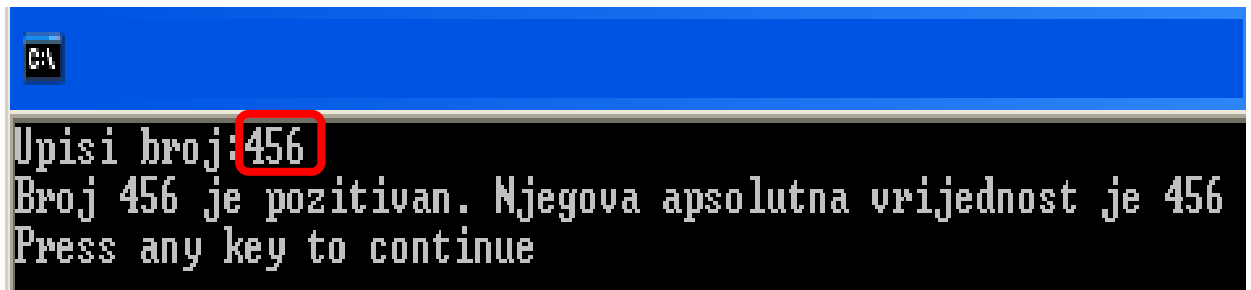
Za ($a < 0$):



```
C:\n
Upisi broj: -456
Broj -456 je negativan. Njegova apsolutna vrijednost je 456
Press any key to continue
```

A screenshot of a Windows command prompt window with a blue title bar. The text is displayed in a monospaced font. The input '-456' is highlighted with a red rectangle. The output shows the number is negative and its absolute value is 456, followed by a prompt to press any key to continue.

Za ($a > 0$):

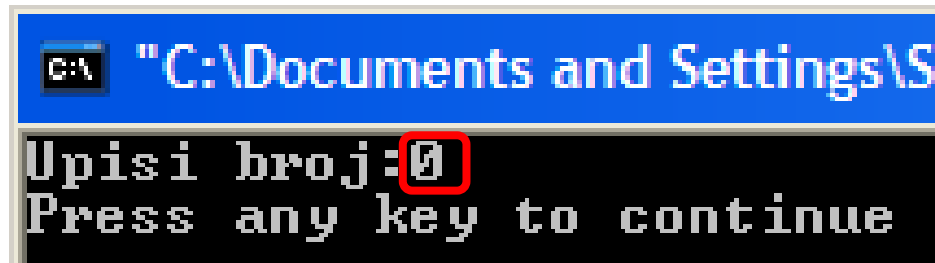


```
C:\n
Upisi broj: 456
Broj 456 je pozitivan. Njegova apsolutna vrijednost je 456
Press any key to continue
```

A screenshot of a Windows command prompt window with a blue title bar. The text is displayed in a monospaced font. The input '456' is highlighted with a red rectangle. The output shows the number is positive and its absolute value is 456, followed by a prompt to press any key to continue.

PRIMJER 2

- ako nije ispunjen niti prvi niti drugi uvjet (unesena je 0), program se prekida

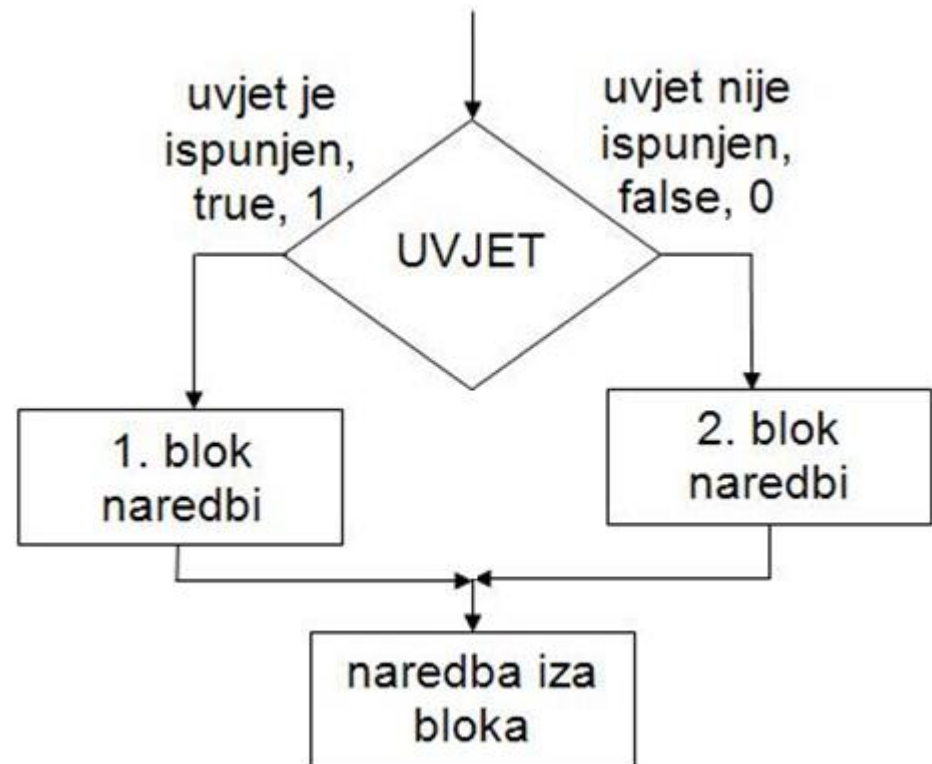


```
C:\ "C:\Documents and Settings\S...  
Upisi broj:   
Press any key to continue
```



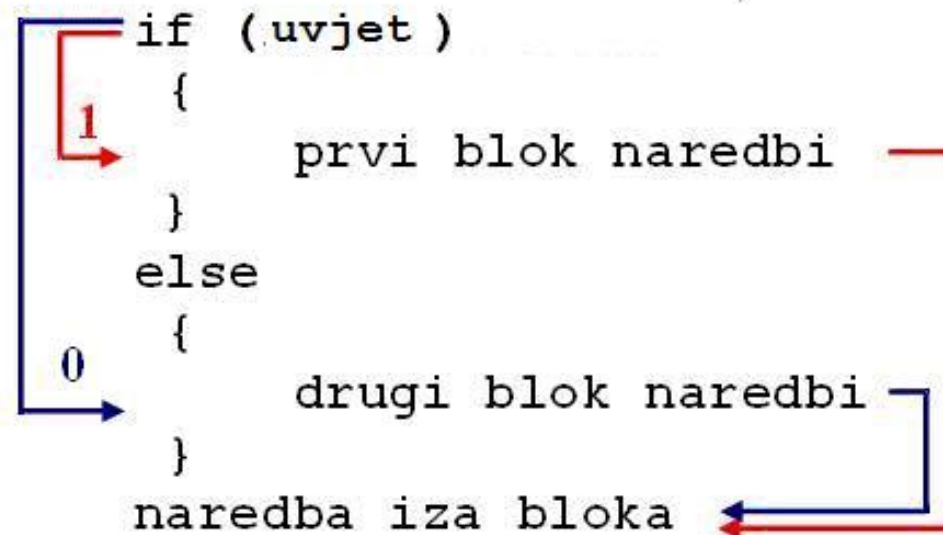
DVOSTRUKO UVJETNO GRANANJE

- dvostruko uvjetno grananje omogućava da se ovisno o ispunjenju postavljenog uvjeta izvodi jedan od dva neovisna bloka naredbi



DVOSTRUKO UVJETNO GRANANJE

- za dvostruko uvjetno grananje rabi se naredba if - else



DVOSTRUKO UVJETNO GRANANJE

- ako je vrijednost uvjeta logička istina izvodi se prvi blok
- nakon njegova završetka izvođenje se nastavlja od prve naredbe iza drugog bloka
- ako je vrijednost uvjeta logička neistina, preskače se prvi blok i izvodi se drugi blok (iza naredbe else)
- nakon njegova završetka izvođenje se nastavlja od prve naredbe iza drugog bloka



PRIMJER 3

Primjer 2 treba riješiti uporabom dvostrukog uvjetnog grananja

Ispis neka bude oblika:

Upisi cijeli broj razlicit od 0:

Broj...je....Njegova apsolutna vrijednost je....



PRIMJER 3

- primjer je riješen uz pomoć *if – else* naredbe
- ako je ($a < 0$) izvršit će se prvi blok naredbi
- ako je ($a > 0$), preskače se prvi blok i izvodi se drugi blok naredbi



PRIMJER 3

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

```
    int a;
    cout<<"Upisi cijeli broj razlicit od 0: ";
    cin>>a;
```

```
    if(a<0)
```

1. BLOK

```
{
```

```
    cout<<"Broj "<<a<<" je negativan. \
    Njegova apsolutna vrijednost je "<<-a<<endl;
```

```
}
```

```
    else
```

2. BLOK

```
{
```

```
    cout<<"Broj "<<a<<" je pozitivan. \
    Njegova apsolutna vrijednost je "<<a<<endl;
```

```
}
```

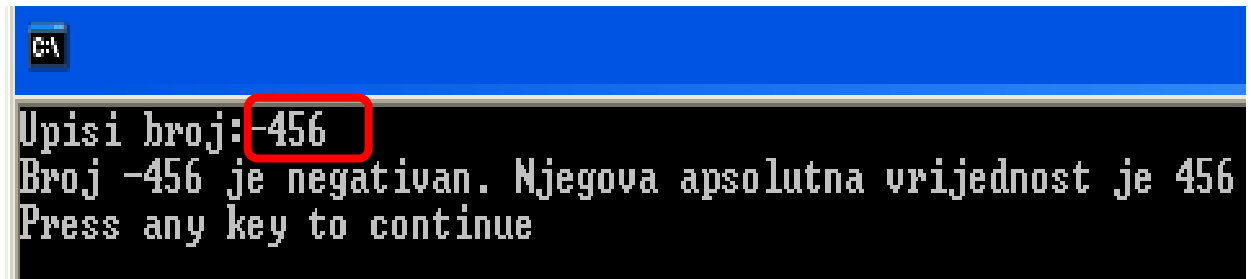
```
    return 0;
```

```
}
```



PRIMJER 3

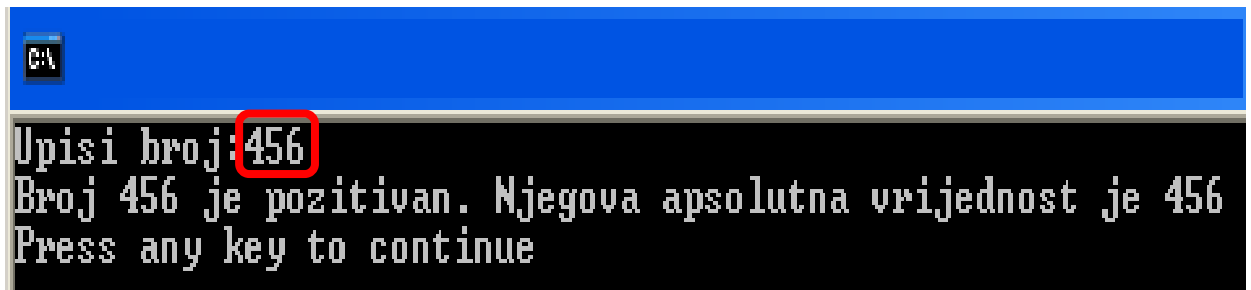
Za ($a < 0$):



```
CA
Upisi broj: -456
Broj -456 je negativan. Njegova apsolutna vrijednost je 456
Press any key to continue
```

A screenshot of a Windows command prompt window with a blue title bar. The window contains text in a monospaced font. The first line is 'Upisi broj:' followed by '-456', which is enclosed in a red rectangular box. The second line reads 'Broj -456 je negativan. Njegova apsolutna vrijednost je 456'. The third line is 'Press any key to continue'.

Ako uvjet nije ispunjen:



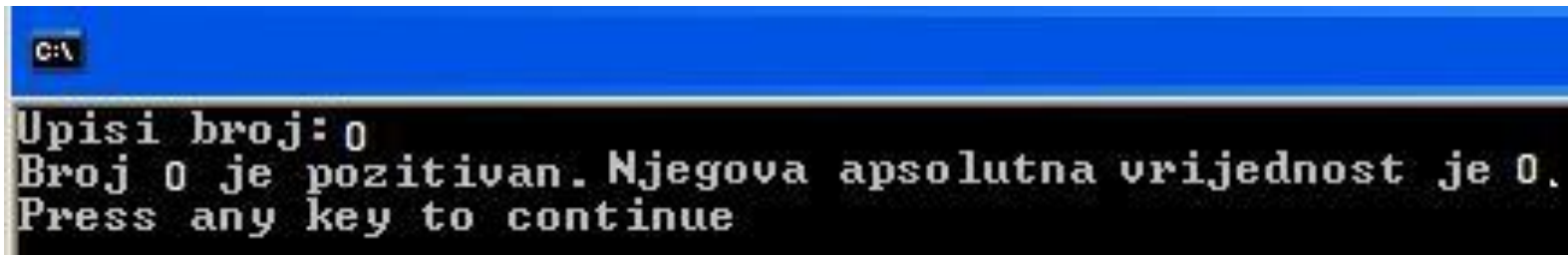
```
CA
Upisi broj: 456
Broj 456 je pozitivan. Njegova apsolutna vrijednost je 456
Press any key to continue
```

A screenshot of a Windows command prompt window with a blue title bar. The window contains text in a monospaced font. The first line is 'Upisi broj:' followed by '456', which is enclosed in a red rectangular box. The second line reads 'Broj 456 je pozitivan. Njegova apsolutna vrijednost je 456'. The third line is 'Press any key to continue'.

POGREŠKA UNOSA

Što ako korisnik ne pročita uputu pažljivo, pa unese broj 0?

Rezultat neće biti ispravan:



```
C:\>  
Upisi broj: 0  
Broj 0 je pozitivan. Njegova apsolutna vrijednost je 0.  
Press any key to continue
```

- bilo bi dobro izbjeći takvu situaciju



POGREŠKA UNOSA

- uneseni broj treba provjeriti, ako je unesena 0, korisnika valja upozoriti, a potom korisnik mora unijeti novi broj koji je različit od 0
- da bi to bilo moguće, potrebna je naredba koja omogućava nastavak odvijanja programa od odabrane naredbe



NAREDBA GOTO

- naredba goto omogućava nastavak odvijanja programa od odabrane naredbe
- naredba na koju se želi skočiti, od koje se želi nastaviti odvijanje programa, može biti bilo gdje u programu

Opći oblik naredbe:

goto oznaka_naredbe;



NAREDBA GOTO

- naredbu od koje se želi nastaviti odvijanje programa treba označiti oznakom iza koje dolazi znak dvotočke

```
oznaka_naredbe:naredba;
```

```
.....
```

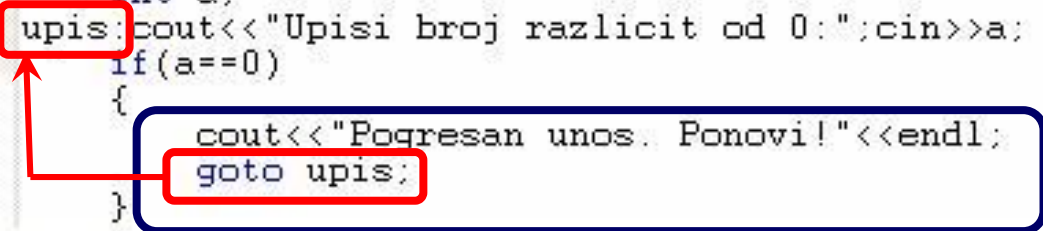
```
.....
```

```
goto oznaka_naredbe;
```



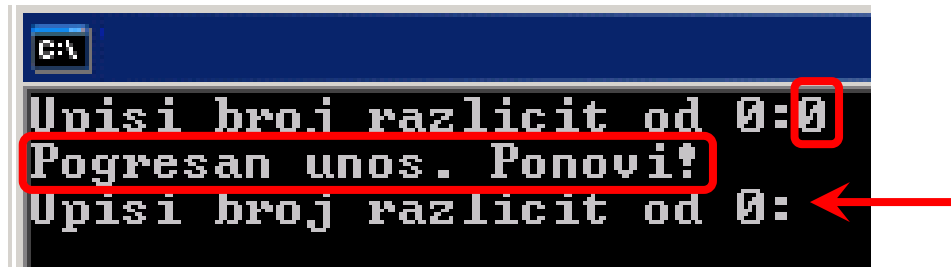
PRIMJER 3 - DOPUNA

```
#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)
    {
        cout<<"Broj "<<a<<" je negativan. \
        Njegova apsolutna vrijednost je "<<-a<<endl;
    }
    else
    {
        cout<<"Broj "<<a<<" je pozitivan. \
        Njegova apsolutna vrijednost je "<<a<<endl;
    }
    return 0;
}
```



PRIMJER 3 - DOPUNA

- za `a==0`



```
GA
Upisi broj razlicit od 0:0
Pogresan unos. Ponovi!
Upisi broj razlicit od 0:
```

The screenshot shows a DOS-style command prompt window with a blue title bar and a black background. The text is displayed in a white monospaced font. The first line is 'Upisi broj razlicit od 0:0', where the second '0' is enclosed in a red square. The second line is 'Pogresan unos. Ponovi!', which is enclosed in a red rectangle. The third line is 'Upisi broj razlicit od 0:', where the final colon is pointed to by a red arrow from the right. The window's title bar contains the text 'GA'.

GOTO

- čestom uporabom naredbe goto teško je slediti tijek odvijanja programa što otežava otkrivanje pogrešaka
- naredbu goto stoga treba izbjegavati i nastojati zadatak riješiti na drugi način, pomoću petlji



UVJETNA NAREDBA – KRATKI OBLIK

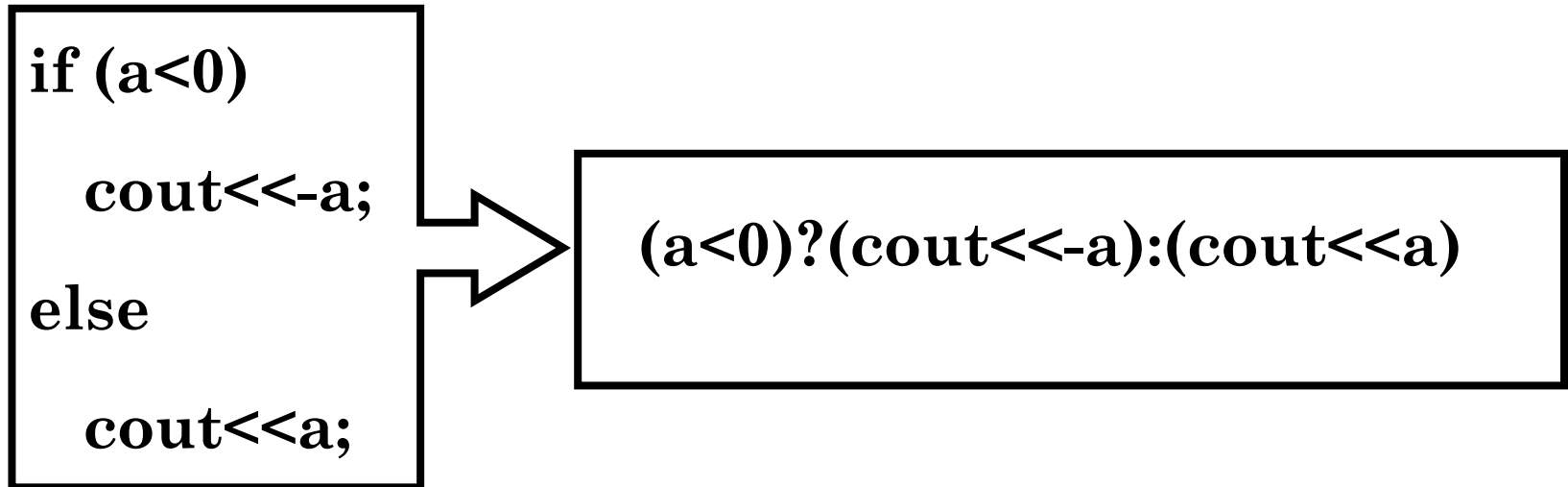
- ako su uvjet i naredbe uvjetnih blokova kratki, umjesto *if - else* naredbi može se koristiti skraćeni oblik zapisa

(uvjet) ? (1.blok naredbi) : (2. blok naredbi)

- koristi se kada uvjet i naredbe blokova stanu u jedan redak

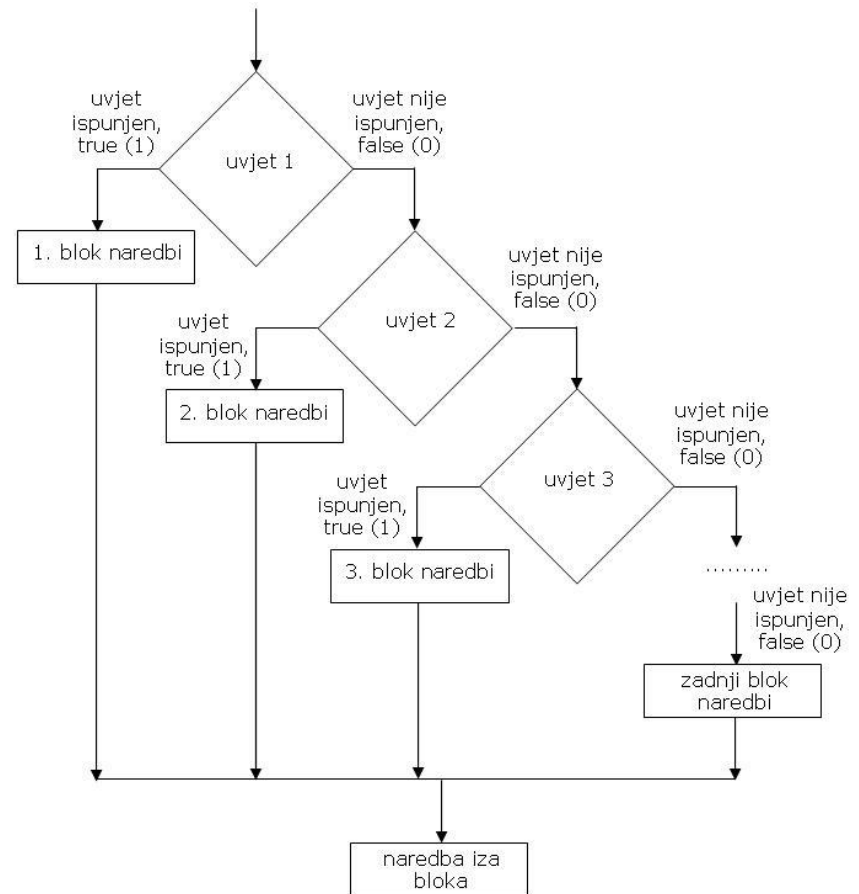


UVJETNA NAREDBA – KRATKI OBLIK



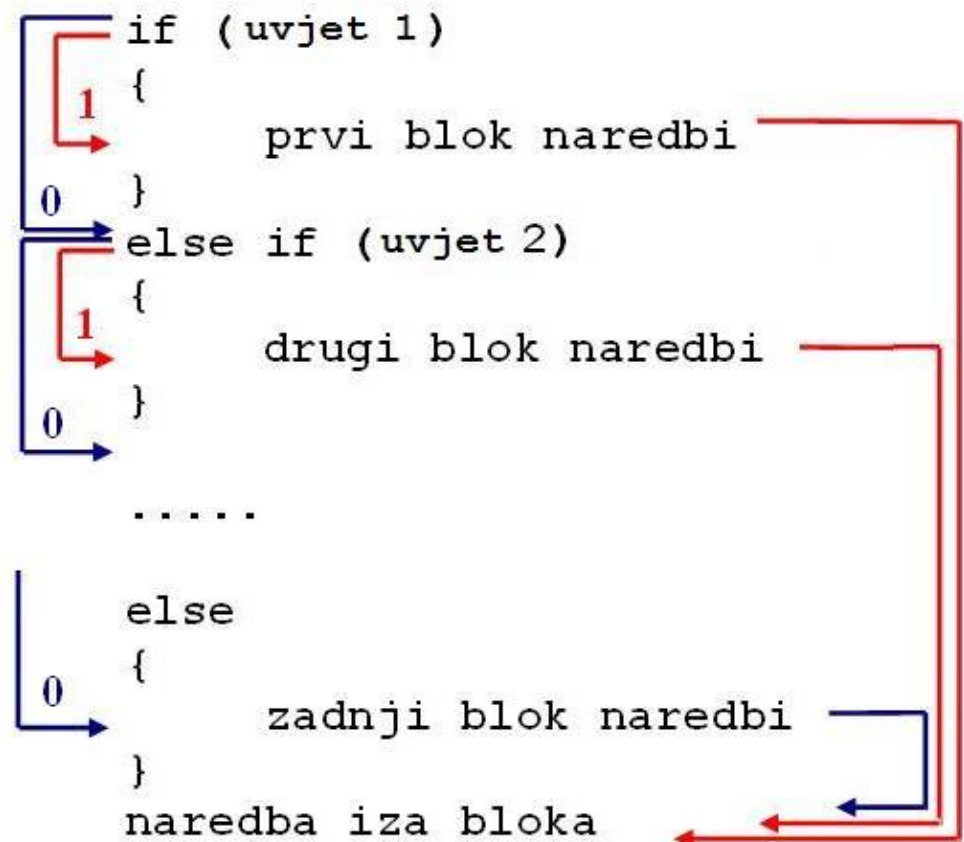
VIŠESTRUKO UVJETNO GRANANJE

- višestruko grananje omogućava ispitivanje više uvjeta
- ovisno o ispunjenju postavljenih uvjeta izvodi se odgovarajući blok naredbi



VIŠESTRUKO UVJETNO GRANANJE

- za višestruko uvjetno grananje rabi se naredba `if – else if - else`
- broj postavljenih uvjeta nije ograničen



VIŠESTRUKO UVJETNO GRANANJE

- ako je vrijednost prvog uvjeta logička istina, izvodi se prvi blok naredbi
- nakon njegova završetka izvođenje se nastavlja od prve naredbe iza zadnjeg bloka naredbi
- ako je vrijednost prvog uvjeta logička neistina, provjerava se drugi uvjet
- ako je on logička istina, izvodi se drugi blok naredbi, a potom prva naredba iza zadnjeg bloka naredbi



VIŠESTRUKO UVJETNO GRANANJE

- ako je vrijednost drugog uvjeta logička neistina, provjerava se treći uvjet, itd.
- provjere se tako redom nastavljaju sve do naredbe else
- ako do tada niti jedan od uvjeta nije imao vrijednost logičke istine, izvršit će se zadnji blok naredbi koji se nalazi iza naredbe else



PRIMJER 4

Primjer 2 treba riješiti uporabom višestrukog uvjetnog grananja

Ispis neka bude oblika:

Upisi broj:

Broj...je....Njegova apsolutna vrijednost je....

ili

Unio si 0. Apsolutna vrijednost od 0 je 0.



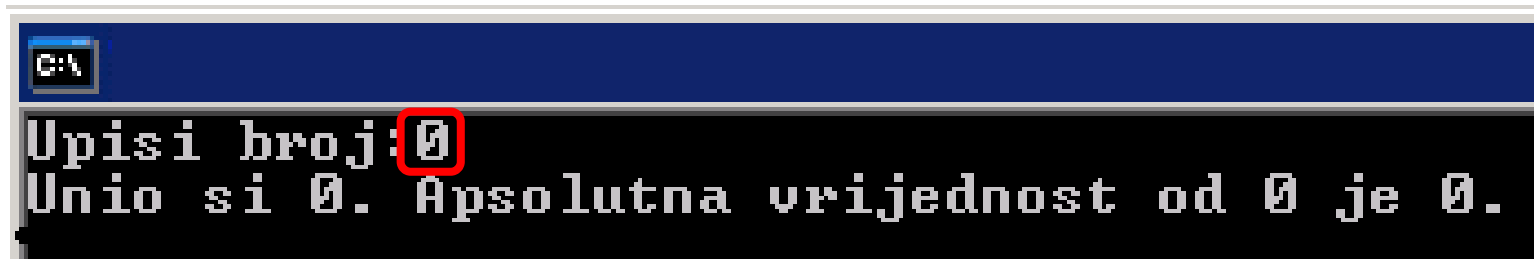
PRIMJER 4

```
#include<iostream>
using namespace std;
int main()
{
    int a;
    cout<<"Upisi cijeli broj: ";
    cin>>a;
    if(a<0)
    {
        cout<<"Broj "<<a<<" je negativan.\n";
        cout<<"Njegova apsolutna vrijednost je "<<-a<<endl;
    }
    else if(a>0)
    {
        cout<<"Broj "<<a<<" je pozitivan.\n";
        cout<<"Njegova apsolutna vrijednost je "<<a<<endl;
    }
    else
    {
        cout<<"Unio si 0. Apsolutna vrijednost od 0 je 0."
        <<endl;
    }
    return 0;
}
```



PRIMJER 4

Ako se unese 0, poruka će biti:



```
04
Upisi broj:0
Unio si 0. Apsolutna vrijednost od 0 je 0.
```

- u ostalim slučajevima, provjera je ista kao u primjeru 3



PRIMJER 5

Treba upisati prirodni broj pa provjeriti da li je veći ili manji od 100, te da li je paran ili neparan

Ispis neka bude oblika:

Upisi prirodni broj:
Uneseni broj je ... od 100 i



PRIMJER 5

- višestrukim uvjetnim grananjem provjerava se da li je broj veći, manji ili jednak 100
- dvostrukim uvjetnim grananjem (neovisno o tome da li je broj veći ili manji od 100) provjerava se parnost broja
- parnost se provjerava operatorom modulo (ostatak dijeljenja s 2 se uspoređuje s 0)



PRIMJER 5

```
#include<iostream>
using namespace std;
int main()
{
    int a;
    upis:cout<<"Upisi prirodni broj:"<<cin>>a;
    if(a<=0)
    {
        cout<<"Pogresan unos. Ponovi!"<<endl;
        goto upis;
    }
    if (a<100)
    {
        cout<<"Uneseni broj je manji od 100 i ";
        if (a%2==0)
            cout<<" paran je."<<endl;
        else
            cout<<" neparan je."<<endl;
    }
    else if (a>100)
    {
        cout<<"Uneseni broj je veci od 100 i ";
        if (a%2==0)
            cout<<" paran je."<<endl;
        else
            cout<<" neparan je."<<endl;
    }
    else
    {
        cout<<"Unesen je broj 100, on je paran"<<endl;
    }
    return 0;
}
```

PRIMJER 5

Provjera:

```
C:\n
Upisi prirodni broj:435
Uneseni broj je veci od 100 i  neparan je.
Press any key to continue_

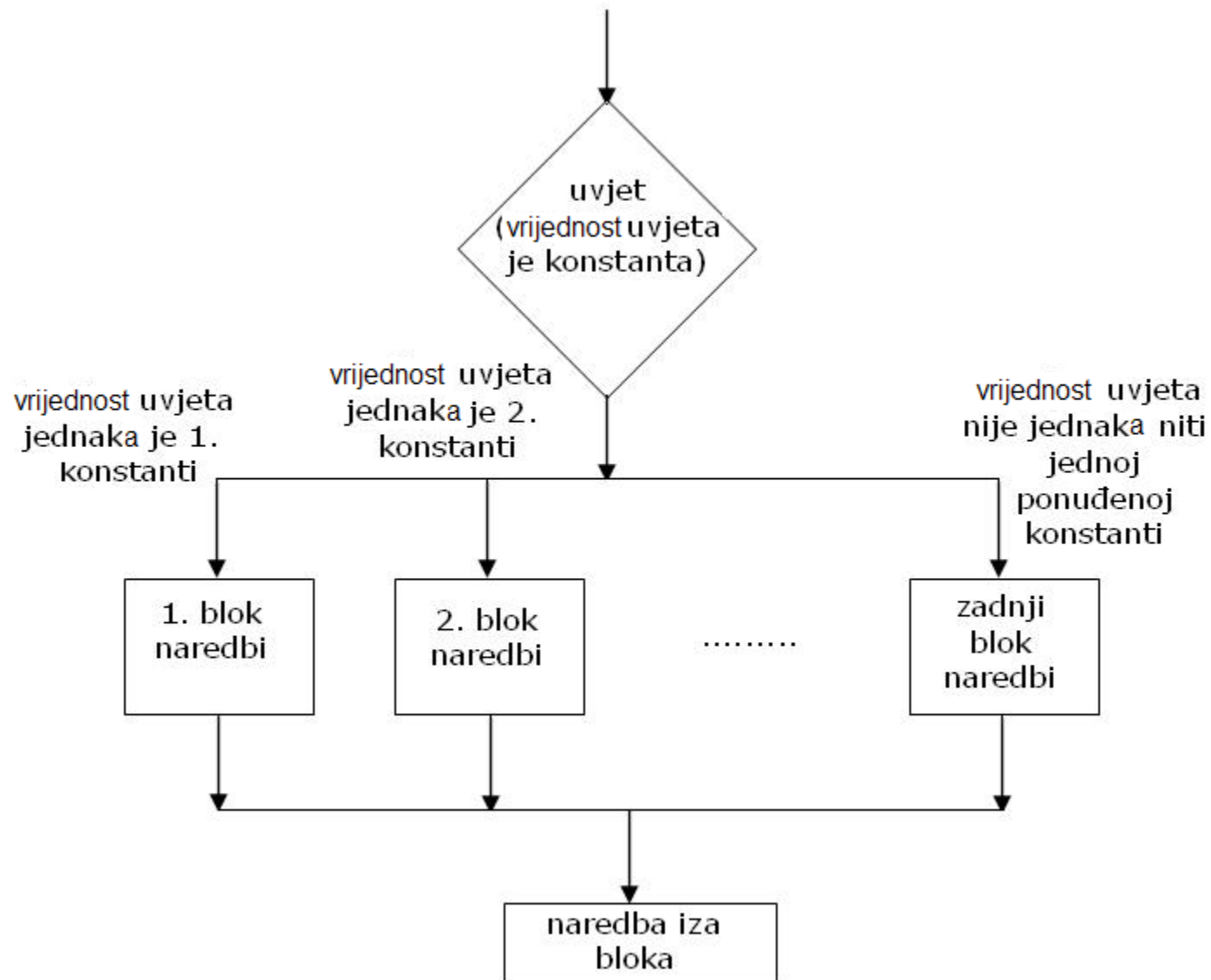
C:\n
Upisi prirodni broj:86
Uneseni broj je manji od 100 i  paran je.
Press any key to continue_

C:\n
Upisi prirodni broj:-9
Pogresan unos. Ponovi!
Upisi prirodni broj:_
```

GRANANJE SWITCH-CASE

- naredba switch-case omogućava jednostruko grananje koje ovisi o vrijednosti postavljenog uvjeta
- za razliku od višestrukog grananja naredbom if u kojoj su uvjeti logički izrazi, u naredbi switch-case uvjet je cjelobrojan izraz ili cjelobrojna varijabla
- izračun cjelobrojnog izraza ili vrijednost cjelobrojne varijable je cijeli broj (cjelobrojna konstanta)





GRANANJE SWITCH-CASE

- vrijednost se uvjeta uspoređuje s nizom zadanih cjelobrojnih konstanti: konst1, konst2, konst3, itd.
- ako je vrijednost uvjeta jednaka nekoj od zadanih konstanti, izvršava se blok naredbi pridružen toj konstanti


```
switch (uvjet)
{
    case (konst1):
        prvi blok naredbi
    break;
    case (konst2):
        drugi blok naredbi
    break;
    case (konst3):
        treći blok naredbi
    break;
    ...
    default:
        zadnji blok naredbi
}
```



GRANANJE SWITCH-CASE

- po izvršenju bloka naredbi (kao primjer uzet je prvi blok), naredba `break` označava izlaz iz bloka `switch-case` i nastavlja program prvom naredbom nakon `switch-case` bloka

```
switch (uvjet)
{
    case (konst1):
        prvi blok naredbi
        break;
    case (konst2):
        drugi blok naredbi
        break;
    case (konst3):
        treći blok naredbi
        break;
    ...
    default:
        zadnji blok naredbi
}
```



GRANANJE SWITCH-CASE

- ako vrijednost uvjeta nije jednaka niti jednoj od ponuđenih konstanti, izvršava se blok naredbi pridružen naredbi default
- u slučaju izostavljanja naredbe default program će nastaviti izvršavanje prvom naredbom nakon switch-case bloka

```
switch (uvjet)
{
    case (konst1):
        prvi blok naredbi
    break;
    case (konst2):
        drugi blok naredbi
    break;
    case (konst3):
        treći blok naredbi
    break;
    ...
    default:
        zadnji blok naredbi
}
```



PRIMJER 6

Treba izračunati ukupni otpor za otpore R1 i R2, ovisno o tome da li su spojeni serijski ili paralelno. Za odabir serijskog spoja korisnik upisuje 1, a za paralelnog 2.

Ispis neka bude oblika:

Otpor R1 (u omima):

Otpor R2 (u omima):

Za serijski spoj otpora upisi 1, a za paralelni 2:

Ako se otpori od ... oma i ... oma spoje u ...
ukupni je otpor ... oma.



PRIMJER 6

- valja načiniti dvije grane (serijski spoj-1, paralelni-2)
- ovisno o tome što korisnik upiše (broj 1 ili broj 2) izvršit će se jedna od dvije grane
- u slučaju da korisnik unese vrijednost koja nije 1 ili 2, ispisat će se upozorenje



```

#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 spoj otpora u seriju upisi 1, a za paralelu 2:";
    cin>>i;
    switch (i)
    {
        case 1:
            R=R1+R2;
            cout<<"Ako se otpori od "<<R1<<" oma i "
            <<R2<<" oma spoje u seriju ukupni je otpor "
            <<R<<" oma."<<endl;
            break;
        case 2:
            R=(R1*R2)/(R1+R2);
            cout<<"Ako se otpori od "<<R1<<" oma i "
            <<R2<<" oma spoje u paralelu ukupni je otpor "
            <<R<<" oma."<<endl;
            break;
        default:
            cout<<"Pogresan unos. Unesi 1 ili 2"<<endl;
    }
    return 0;
}

```

PRIMJER 6

PRIMJER 6

Provjera:

C:\

```
Otpor R1 (u omima):2  
Otpor R2 (u omima):4  
Za spoj otpora u seriju upisi 1, a za paralelu 2:1  
Ako se otpori od 2 oma i 4 oma spoje u seriju ukupni je otpor 6 oma.
```

C:\

```
Otpor R1 (u omima):2  
Otpor R2 (u omima):4  
Za spoj otpora u seriju upisi 1, a za paralelu 2:2  
Ako se otpori od 2 oma i 4 oma spoje u paralelu ukupni je otpor 1.33333 oma.
```

PRIMJER 6

- ako se ne upiše broj 1 ili 2 ispisat će se poruka o pogrešci

```
C:\n
Otpor R1 (u omima):2
Otpor R2 (u omima):4
Za spoj otpora u seriju upisi 1, a za paralelu 2:6
Pogresan unos. Unesi 1 ili 2
```



HVALA NA PAŽNJI!

