

Objektno orjentisano programiranje – C++

Naredbe za kontrolu toka programa



Teme

- Blokovi naredbi
- Naredba if
- Naredba switch
- Petlja for
- Petlja while
- Petlja do-while
- Naredba break i continue
- Ugneždene petlje
- Naredba bezuslovnog skoka



Blokovi naredbi

- Delovi programa koji se uslovno izvode ili čije se izvršavanje ponavlja grupišu se u blokove naredbi tj. u jednu ili više naredbi unutar para vitičastih zagrada { }
- **Blokovi naredbi** se uobičajeno pišu **uvučeno**, zbog preglednosti programa
 - promenljive deklarisanе u bloku vidljive su samo unutar njega
- Ograničenje važnosti (engl. *scope*) lokalne promenljive

Blokovi naredbi (2)

```
#include <iostream>
int main() {
    using namespace std;
    {
        int i=1;
    }
    cout<<i; //greška
    return 0;
}
```

```
#include <iostream>
int main() {
    using namespace std;
    int i=10;
    {
        int i=1;
        cout<<i<<endl;
    }
    cout<<i<<endl;
    return 0;
}
```

1
10

Grananje naredbom IF

- Opšta forma naredbe je:
if (izraz) naredba;
else naredba; //nije obavezno
- If petlju može da kontroliše bilo koji validni logički izraz



Grananje naredbom IF (2)

```
#include <iostream>
int main() {
    using namespace std;
    int i=10;
    if (i<0)
        cout<<"broj je negativan"<<endl;
    if (i % 2)
        cout<<"broj je neparan"<<endl;
    else
        cout<<"broj je paran"<<endl;
    return 0;
}
```



Primer #1

```
#include <iostream>

int main(){
    using namespace std;
    int prviBroj, drugiBroj;
    cout << "Uneti veliki broj: ";
    cin >> prviBroj;
    cout << "\nUneti mali broj ";
    cin >> drugiBroj;
    if (prviBroj > drugiBroj)
        cout << "\nHvala!\n";
    else
        cout << "\nGreška, drugi broj je veći!";
    return 0;
}
```

```
Uneti veliki broj: 1000
Uneti mali broj 10
Hvala!
```

IF-ELSE-IF lestvica

```
if(uslov1)
    naredbe;
else if(uslov2)
    naredbe;
else if(uslov3)
    naredbe;
...
else naredbe;
```

- Moguće je pisati i ugneždene IF blokove; u tom slučaju, blok ELSE uvek pripada najbližem IF-u
- Uslovni izrazi se proveravaju odozgo naniže; čim se pronađe tačan, izvršava se naredba povezana sa njim, a ostatak lestvice se preskače
- Poslednji else služi kao default izbor, ako ništa pre njega nije ispunjeno

Primer #2

```
#include <iostream>

int main() {
    using namespace std;
    int prviBroj, drugiBroj;
    cout << "Uneti dva broja.\nPrvi: ";
    cin >> prviBroj;
    cout << "\nDrugi: ";
    cin >> drugiBroj;
    cout << "\n\n";
    if (prviBroj >= drugiBroj) {
        if ( (prviBroj % drugiBroj) == 0) // parno deljivi? {
            if (prviBroj == drugiBroj)
                cout << "Brojevi su isti!\n";
            else cout << "Parno su deljivi!\n";
        }
        else cout << "Nisu parno deljivi!\n";
    }
    else cout << "Drugi broj je veći!\n";
    return 0;
}
```

Uslovni operator ?

- Uslovni operator treba koristiti samo za jednostavna ispitivanja, kada naredba staje u jedan red; u suprotnom, kod postaje nečitljiv.

```
#include <iostream>
int main(){
using namespace std;
int x, y = 10;
    x = (y < 10) ? 30 : 40;
    cout <<"Vrijednost za x: " << x << endl;
    return 0;
}
```



Grananje toka naredbom switch

```
switch(izraz) {  
    case (konstanta1):  
        naredbe;  
        break;  
    case (konstanta2):  
        naredbe;  
        break;  
    case(konstanta3):  
        naredbe;  
        break;  
    ...  
    default  
        naredbe;
```

- Naredba switch se koristi umesto serije ugneženih IF- ELSE naredbi, kada se kontrola naredbe može vršiti izrazom tipa char ili int
- Blok default može stajati bilo gde unutar bloka switch, ali zbog preglednosti preporučljivo ga je staviti na kraj



Izvršavanje naredbe switch

- Izračunava se izraz
- Izračunata vrednost se redom upoređuje sa vrednostima u case delovima
- Za prvu pronađenu istu vrednost se izvršava odgovarajuća naredba u nastavku
- Ako nije pronađena nijedna ista vrednost, izvršava se naredba iza reči default, ako postoji
- Naredba break nije obavezna, ali bez nje grananje naredbom switch nema mnogo smisla

Naredba switch: Primer

```
#include <iostream>
#include <string>
using namespace std;
int main () {
    switch (x) {
        case 1:
            cout << "x = 1";
            break;
        case 2:
            cout << "x = 2";
            break;
        default:
            cout << "x ";
    }
    return 0;
}
```

//if ekvivalent

```
if (x == 1) {
    cout << "x = 1";
}
else if (x == 2) {
    cout << "x = 2";
}
else {
    cout << "x nepoznat";
}
```

Petlja for

for (inicijalizacija; uslov; inkrement) naredba;

- Uslovni izraz se uvek testira pre ulaska u petlju; petlja se izvršava sve dok je uslov ispunjen.

```
int main (){  
    int i;  
    for (i = 0; i < 5; i++)  
        cout << "U petlji! ";  
    cout << "\ni=: " << i << ".\n";  
    return 0;  
}
```

U petlji! U petlji! U petlji! U petlji! U petlji!
i=: 5.

Petlja for (2)

```
int main ()  
{  
    int i,j;  
    for (int i=0, j=0; i<3; i++, j++)  
        std::cout << "i: " << i << " j: " << j << endl;  
    system("pause"); → Na ekranu ispisuje „Press any key to continue ..  
    return 0;  
}
```

```
i: 0 j: 0  
i: 1 j: 1  
i: 2 j: 2  
Press any key to continue  
...
```

Petlja for (3)

```
int i = 0;
for( ; i < 5; ) {
    i++;
    cout << "U petlji! " << endl;
}
system("pause"); Na ekranu ispisuje „Press any key to continue ..
return 0;
}
```

```
U petlji!
U petlji!
U petlji!
U petlji!
U petlji!
Press any key to continue . .
```

Petlja for (4)

```
#include <iostream>
using namespace std;
int main (){
    int i=0; int max;
    cout << "Koliko poruka?";
    cin >> max;
    for (;;) { // petlja bez kraja
        if (i < max) { // test
            cout << "U petlji!\n";
            i++;
        }
    }
```

```
else
    break;
}
system("pause");
return 0;
}
```

Koliko poruka?3
U petlji!
U petlji!
U petlji!
Press any key to continue .

Naredba while

while (izraz) naredba;

- Koristi se za ponavljanje delova koda kod koga broj ponavljanja nije unapred poznat
- Svaka for petlja se može napisati kao while i obrnuto



Naredba while (2)

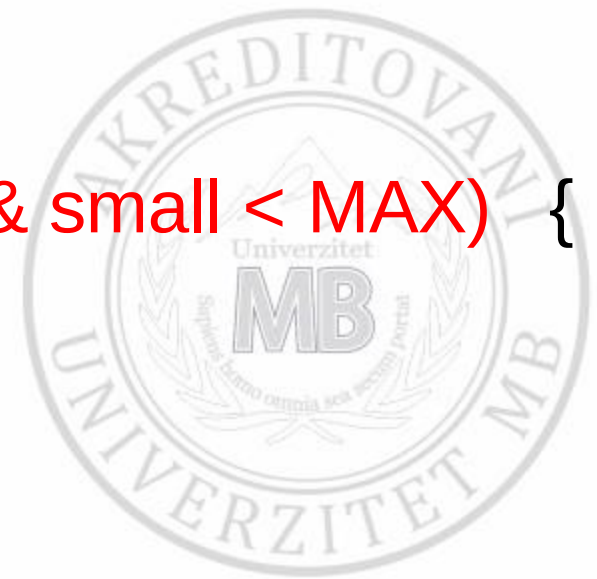
```
#include <iostream>
using namespace std;
int main () {
    int i = 0;
    while(i < 5) {
        i++;
        cout << "i =: " << i << "\n";
    }
    cout << "i = " << i << ".\n";
    return 0;
}
```

```
i =: 1
i =: 2
i =: 3
i =: 4
i =: 5
i = 5.
```



Naredba while (3)

```
#include <iostream>
using namespace std;
int main() {
    unsigned short small;  unsigned long  large;
    const unsigned short MAX=65535;
    ....
    while (small < large && large > 0 && small < MAX) {
        ....
        return 0;
    }
}
```



Blok do while

do

blok naredbi

while (izraz);

- Ova petlja je korisna kada je potrebno da se izvrši neka operacija, pa da se u zavisnosti od njenog ishoda ona eventualno ponavlja



Blok do while (2)

```
#include <iostream>
int main() {
    using namespace std;
    int i =10;
    do {
        cout << „C++\n”;
        i--;
    } while (i >0 );
    cout << " i is: " << i << endl;
    return 0;
}
```



Blok do while (3)

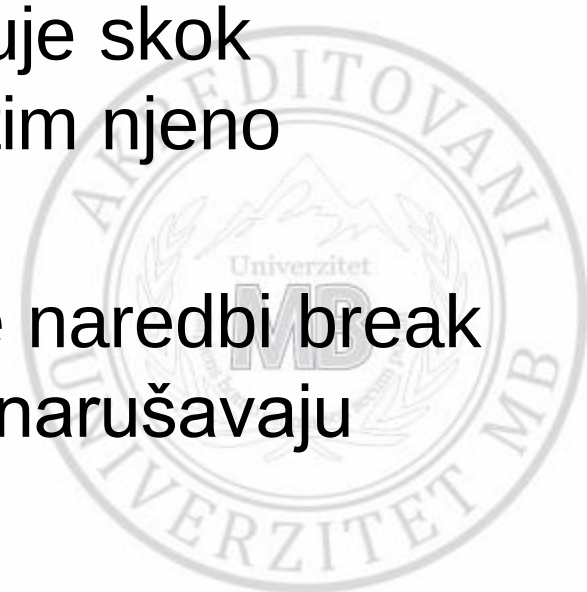
```
#include <iostream>
using namespace std;
int main () {
    using namespace std;
    int i =4;
    do {
        cout << "C++\n"; i--;
    } while (i >0 );
    cout << " i = " << i << endl;
    return 0;
}
```

```
C++
C++
C++
C++
i = 0
```

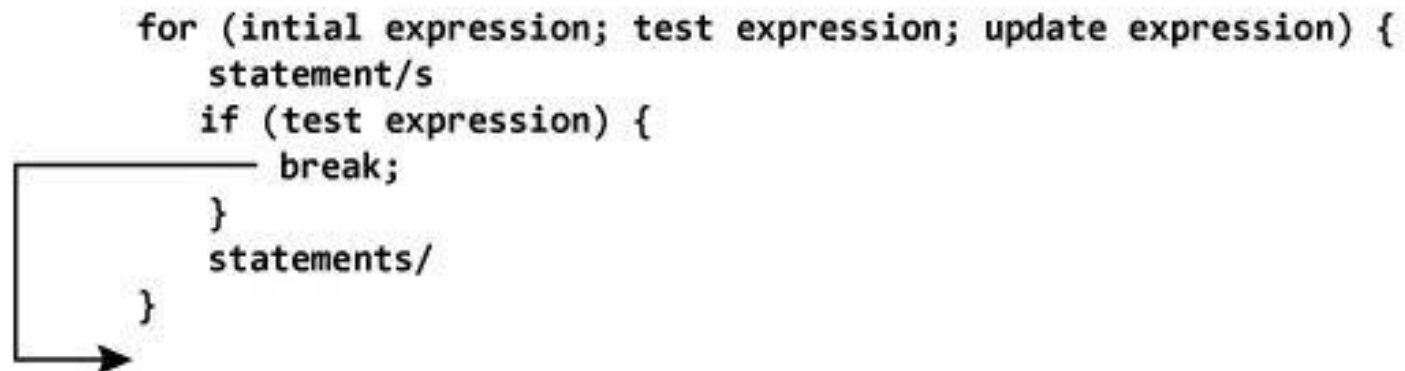
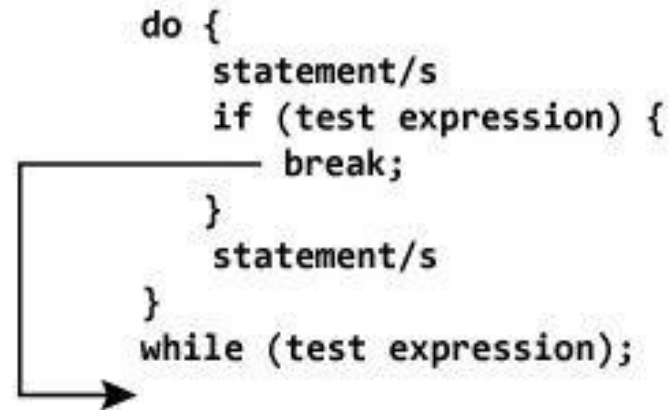
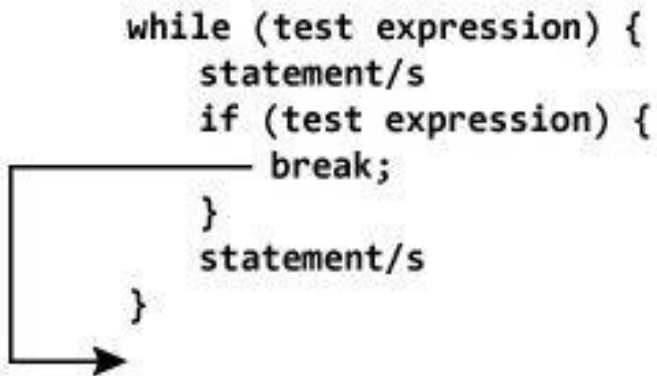


Naredbe break i continue

- **Naredba break** se osim u grananju switch može koristiti i za prekid izvršavanja petlji for, while i do while
- **Naredba continue** takođe uzrokuje skok programa na kraj petlje, ali se zatim njeno izvršavanje nastavlja
- Treba izbegavati često korišćenje naredbi break i continue u programima, jer one narušavaju strukturiranost programa



Naredba break



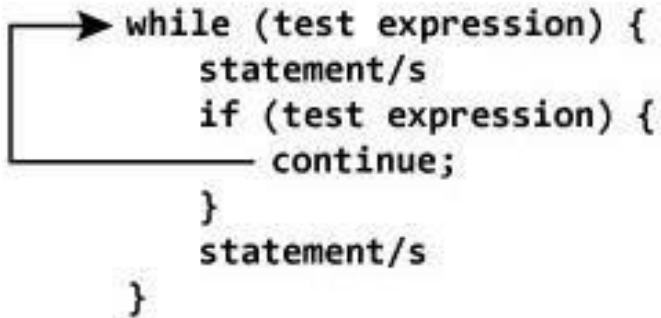
Naredba break (2)

```
float number, sum = 0.0;
while (true) {    // test, uvek istinit
    cout<<"Uneti broj: ";
    cin>>number;
    if (number != 0.0) {
        sum += number;
    }
    else {
        break; //number=0.0
    }
}
```

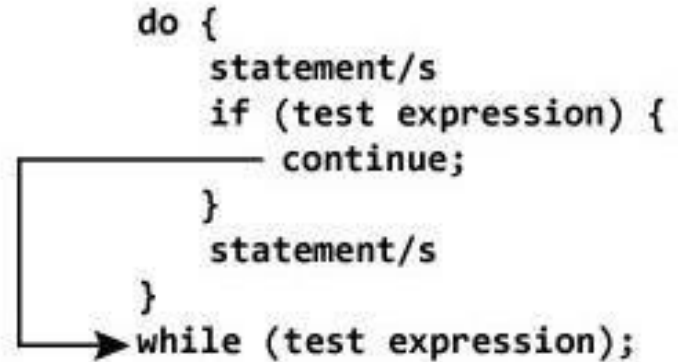


Naredba continue

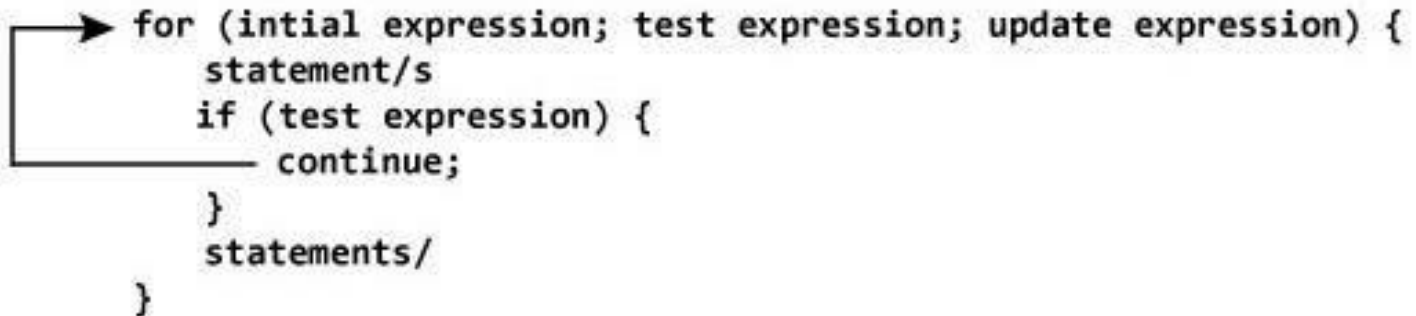
```
while (test expression) {  
    statement/s  
    if (test expression) {  
        continue;  
    }  
    statement/s  
}
```



```
do {  
    statement/s  
    if (test expression) {  
        continue;  
    }  
    statement/s  
} while (test expression);
```



```
for (initial expression; test expression; update expression) {  
    statement/s  
    if (test expression) {  
        continue;  
    }  
    statements/  
}
```



Naredba continue (2)

```
#include <iostream>
using namespace std;
int main() {
    for (int i = 1; i <= 10; ++i) {
        if ( i == 6 || i == 9) {
            continue;
        }
        cout<<i<<"\t";
    }
    return 0;
}
```



Ugneždene petlje

- Petlje koje se nalaze jedna u drugoj

```
for (izraz) {  
    for (izraz) {  
        for (izraz) {  
            for (izraz) {  
                .....  
            }  
        }  
    }  
}
```



Ugneždene petlje (2)

```
#include <iostream>
using namespace std;
int main() {
    for (int sat = 0; sat < 24; sat++) {
        for (int minut = 0; minut < 60; minut++) {
            for (int sekund = 0; sekund < 60; sekund++) {
                cout << sat << ":" << minut << ":" << sekund << endl;
            }
        }
    }
    return 0;
}
```

```
0:0:0
0:0:1
0:0:2
....
0:0:59
0:1:0
0:1:1
....
```

Ostale naredbe za skok

goto ime_oznake;

- Naredba *goto* omogućava bezuslovni skok na neku drugu naredbu unutar iste funkcije
- *ime_oznake* mora biti jedinstveno unutar funkcije
- Naredbom *return* prekida se izvršavanje funkcija

```
if(a == 0)
    goto deljenjeNulom;
```

```
deljenjeNulom:
    cout << "Deljenje nulom nije dozvoljeno!" << endl;
```

Zaključak

- **Blokovi naredbi:** grupa naredbi unutar para { }
- **if-else naredba**
- **if-else-if:** ugneždene IF naredbe
- **Uslovni operator ?:** koristi se samo za jednostavna ispitivanja kada naredba staje u jedan red
- **Naredba switch:** izračunata vrednost se redom upoređuje sa vrednostima u **case** delovima
- **Petlja for:** petlja se izvršava sve dok je uslov ispunjen (testira se pre ulaska u petlju)

Zaključak (2)

- **Naredba while:** koristi se za ponavljanje delova koda kod koga broj ponavljanja nije unapred poznat
- **Blok do-while:** izvrši se neka operacija pa se u zavisnosti od njenog ishoda ona eventualno ponavlja
- **Naredba break:** prekid izvršavanja petlji switch, for, while i do-while
- **Naredba continue:** skok programa na kraj petlje
- **Ugneždene petlje:** više petlji jedna u drugoj
- **Naredba goto:** bezuslovni skok na neku drugu naredbu unutar iste funkcije

Kraj prezentacije

HVALA NA PAŽNJI!

