Naredbe za kontrolu toka

Naredbe za kontrolu toka

- Nakon odslušanog bit ćete u stanju:
  - objasniti semantiku naredbi za kontrolu postupaka
  - navesti sintaksu naredbi `if`, `if-else` i `case` u programskom jeziku C
  - objasniti pojam iteracije
  - navesti primjere korištenja
  - navesti sintaksu naredbi `for`, `while` i `do-while` u programskom jeziku C
  - primijeniti navedene naredbe.

Naredba `if`

- Najjednostavnija naredba `if` ima oblik

  `if`(uvjet) naredba;

  gdje je uvjet aritmetički (logički) izraz.

Primjer:

```
int i, j, temp;
......
if (i > j){
        temp = i;
        i = j;
        j = temp;
 }
```

Naredba `if-else`

```
if(uvjet)
    naredba_1;
else
    naredba_2;
```

- Ako izraz uvjet ima vrijednost istine, onda se izvršava naredba_1 a u suprotnom naredba_2.

Primjer:

```
if (a > b) max = a;
else max = b;
```

Ugniježđene `if-else` naredbe

- Svaki `else` pripada sebi najbližoj `if` naredbi!

  Primjer:

```
if (n > 0)
        if (a > b) z = a;
else    /* Loš način pisanja */
        z = b;
```

  Pripadnost mijenjamo pomoću vitičastih zagrada:

```
if (n > 0) {
        if (a > b) z = a;}
else
        z = b;
```

Funkcija `exit`

- Funkcija

```
void exit(int status);
```

deklarirana je u datoteci zaglavlja `stdlib.h`. Ona zaustavlja izvršavanje programa i vrijednost status predaje operacijskom sustavu.
status = 0 znači da je program uspješno završen;
Vrijednost različita od nule signalizira da je program zaustavljen uslijed greške.

Primjer:

```
if (!x){
        printf("Djeljitelj je jednak nuli.\n");
        exit(-1);}
else y /= x;
```

Naredba `switch`

```
switch (izraz) {
    case konstanta_1:
                naredba_1;
                ......
                break;
    case konstanta_2:
                naredba_2;
                ......
                break;
            ......
    default:
            naredba;
}
```

Naredba `switch` (2)

- Izraz u naredbi `switch` mora imati cjelobrojnu vrijednost (`char`, `int` ili `enum`).
- Nakon ključne riječi `case` pojavljuju se cjelobrojne konstante ili konstantni izrazi.
- Na početku izvršavanja naredbe `switch` prvo se testira izraz. Ako je vrijednost izraza jednaka konstanta_i onda se izvršava naredba_i i sve naredbe koje dolaze nakon nje, sve do prve `break` naredbe ili kraja `switch` naredbe. Nakon toga program se nastavlja prvom naredbom iza `switch` naredbe.

Naredba `switch` (3)

- Ako izraz nije jednak niti jednoj konstanti, onda se izvršava naredba koja dolazi nakon ključne riječi `default` (ako postoji) i sve naredbe iza nje, sve do kraja `switch` naredbe.
- Slučaj `default` ne mora biti nužno prisutan u `switch` naredbi. Ako nije, i ako nema podudaranja vrijednosti izraza i navedenih konstanti, program se nastavlja prvom naredbom iza `switch` naredbe.

Primjer:

```
switch (operator) {
        case 'z':
                printf("%f\n", a + b);
                break;
        case 'o':
                printf("%f\n", a - b);
                break;
        case 'm':
                printf("%f\n", a * b);
                break;
        default:
                 printf("Nedopustena operacija!");
  }
```

Primjer:

```
unsigned i;
.....
switch (i) {
        case 1:
        case 2:
        case 3:
        case 4: printf("i < 5\n");
                break;
        case 5: printf("i = 5\n");
                break;
        default: printf("i > 5\n");
   }
```

Iteracija

- Iteracija – ponavljanje nečega dok nije postignut određeni, unaprijed zadani cilj, kao na primjer
  - broj ponavljanja je dosegnuo gornju granicu
  - varijabla je poprimila vrijednost `n`
  - pokazivač je poprimio vrijednost `NULL`
  - …
- Obično se prilikom ponavljanja mijenja neko stanje.

Iteracija (2)

- Primjeri ponavljanja:
  - nizanje ogrlice (crvena, plava i žuta perlica, 20 puta)
  - bacanje lopte sve dok tri puta za redom ne pogodimo koš
  - bacanje kocke sve dok ukupni zbroj ne postane veći od 50.
- U programiranju iteraciju realiziramo pomoću naredbi za ponavljanje (petlji).
- Petlje uobičajeno dolaze u dva osnovna oblika.

Petlja `while`

`while` (izraz) naredba;

Primjer:

```
brojac = 1;
while (brojac <  5)
{
    printf ("brojac = %d\n", brojac);
    ++brojac;
}
```

Primjer:

```
while (a != b) if (a > b ) a -= b; else b -= a;
```

Petlja `for`

`for` (izraz_1; izraz_2; izraz_3) naredba;

ekvivalentno je s

```
izraz_1;
while (izraz_2){
    naredba;
    izraz_3;
}
```

Petlja `for` (2)

Primjer:

```
for (brojac = 1; brojac  < 5; ++brojac)
    printf ("brojac = %d\n", brojac);
```

Primjer:

```
for (brojac = 1; brojac  < 5; ++brojac);
    printf ("brojac = %d\n", brojac);
```

Primjer: beskonačna petlja koja ne radi ništa.

```
for( ; ; );
```

Petlja `do-while`

```
do
        naredba;
while(izraz);
```

- Naredba će se izvršavati sve dok izraz ima vrijednost istine. Za razliku od petlji `while` i `for` vrijednost izraza se kontrolira na kraju prolaza kroz petlju. Petlja se stoga izvršava barem jednom.

Petlja `do-while` (2)

Primjer:

```
brojac = 1;
do
{
        printf ("brojac = %d\n", brojac);
        ++brojac;
}
while (brojac < 5);
```

Naredba `break`

- Naredba `break` služi za izlazak iz naredbe `switch` te zaustavljanje petlje. Može se koristiti unutar petlji `for`, `while` i `do-while`. Pri nailasku na naredbu `break` kontrola programa se prenosi na prvu naredbu iza naredbe unutar koje se `break` nalazi.

Primjer: izlaz iz beskonačne petlje.

```
while(1){
    scanf("%d", &i);
    if(i < 0) break;
.....
}
```

Naredba `continue`

- Naredba `continue` može se koristiti unutar petlji `for`, `while` i `do-while`. Pri nailasku na naredbu `continue` preostali dio tijela petlje se preskače i program nastavlja sa sljedećim prolazom kroz petlju.

Primjer:

```
while(k){
    scanf("%d", &i);
    if(i < 0) continue;
.....
}
```

Primjer:

```
#include <stdio.h>

int main()
{
   for(putchar('1'); putchar('2'); putchar('3')){
      putchar('4');
      break; /* continue; */
      putchar('5');
   }
   return 0;
}
```

Naredba `goto`

- Naredba `goto` prekida sekvencijalno izvršavanje programa i nastavlja izvršavanje s naredbom koja je označena labelom. Oblik joj je

  `goto` labela;

  gdje je labela identifikator koji služi za označavanje naredbe kojim se nastavlja program. Sintaksa joj je

  labela: naredba;

- Labela na koju se vrši skok mora biti unutar iste funkcije kao i `goto` naredba (pomoću naredbe `goto` se ne može izaći iz funkcije).

Naredba `goto` (2)

Primjer:

```
……
if ((fp = fopen ("rezultati_ispita", "r")) == NULL) goto error;
……
error: {
            printf("Ne mogu citati!\n");
            exit(-1);
        }
```