Kontrola toka programa

Slide 1

Izrazi i naredbe

Izrazi:

```
x=3   n++   printf(....)
```

Naredbe (završavaju znakom `;`):

```
x=3;
n++;
printf(....);
```

Složena naredba (blok naredbi) omeđena je vitičastim zagradama:

```
{x=3;
 n++;
 printf(....);}
```

Tamo gdje se može pojaviti jedna naredba može se pojaviti i blok naredbi.

Slide 2

`if` naredba

Najjednostavnija `if` naredba ima oblik

```
if(uvjet) naredba;
```

gdje je `uvjet` aritmetički (ili pokazivački) izraz. Naredba prvo izračunava izraz `uvjet` i ako je njegova vrijednost različita od nule izvršava se `naredba`. Ako izraz `uvjet` daje nulu, onda se `naredba` ne izvršava i program se nastavlja prvom naredbom iza `if` naredbe.

Primjer:

```
int x;
....
if(x>0) printf("\n x= %d \n",x);
```

Slide 3

`if-else` naredba

```
if(uvjet)
    naredba1;
else
    naredba2;
```

Ako izraz `uvjet` ima vrijednost istine, onda se izvršava `naredba1` a u suprotnom `naredba2`.

Primjer:

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

Slide 4

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, a vrijednost različita od nule signalizira da je program zaustavljen usljed greške.

Slide 5

`if` naredba i uvjetni operator

Sljedeće dvije naredbe su ekvivalentne:

1)

```
max = a>b ? a : b
```

2)

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

Slide 6

Programiranje (C) Ak. god. 2003/04

Ugniježdene `if -- else` naredbe

`if-else` naredbe mogu se ugnijezditi.

Pomoću dvije `if - else` naredbe dobivamo

```
if(uvjet1)
    naredba1;
else if(uvjet2)
    naredba2;
else
    naredba3;
```

Slide 7

```
#include <stdio.h>
int main(void)
{  float a,b; char operacija;

   printf("Upisati prvi broj: "); scanf(" %f",&a);
   printf("Upisati drugi broj: "); scanf(" %f",&b);
   printf("Upisati operaciju: zbrajanje(z), oduzimanje(o),\n")
   printf("                     mnozenje(m),dijeljenje(d) :");
   scanf(" %c",&operacija);

   if(operacija==’z’) printf("%f\n",a+b);
   else if(operacija==’o’) printf("%f\n",a-b);
   else if(operacija==’m’) printf("%f\n",a*b);
   else if(operacija==’d’) printf("%f\n",a/b);
   else printf("Nedopustena operacija!\n");  return 0;
}
```

Slide 8

M. Jurak pred4.tex

Kojem `if` pripada `else`?

Svaka `else` naredba pripada sebi najbližoj `if` naredbi!

```
if(n>0)
  if(a>b) z=a;
else  /*   Los stil  */
    z=b;
```

Pripadnost mijenjamo pomoću vitičastih zagrada:

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

Slide 9

`switch` naredba : (primjer s dvije `case` konstante)

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

Slide 10

- Izraz u `switch` naredbi mora imati cjelobrojnu vrijednost (`char`, `int` ili `enum`).
- Nakon ključne riječi `case` pojavljuju se cjelobrojne konstante ili konstantni izrazi.
- Izvršavanje `switch` naredbe: testira se **`izraz`**. Ako je **`izraz`** = **`konstanta_i`** program se nastavlja naredbom **`naredba_i`** i svim naredbama koje dolaze nakon nje sve do prve **`break`** naredbe ili kraja `switch` naredbe. Nakon toga program se nastavlja prvom naredbom iza `switch` naredbe.
- Ako **`izraz`** nije jednak niti jednoj konstanti, onda se izvršava samo **`naredba`** koja dolazi nakon ključne riječi **`default`** (ako postoji) i sve naredbe iza nje, sve od kraja `switch` naredbe.
- Slučaj **`default`** ne mora nužno biti prisutan u `switch` naredbi. Ako nije i ako nema podudaranja izraza i konstanti, program se nastavlja prvom naredbom iza `switch` naredbe.

Slide 11

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

Slide 12

Efekt ispuštanja naredbe `break` je "propadanje kôda" u niži `case` blok.

```
unsigned i;
.....
switch(i) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5:  printf("i < 6\n");
           break;
  case 6: printf("i = 6\n");
           break;
 default:  printf("i > 6\n");
 }
```

Slide 13

`while` petlja

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

`naredba` će se izvršavati sve dok `izraz` ima vrijednost istine (različit od nule).

Primjer:

```
i=0;
while (i<10){
   printf("%d\n",i);
   i++;
}
```

Slide 14


```
/* Srednja vrijednost upisanih brojeva (razlicitih od 0). */
   int main(void){  /* Broj sumanada je nepoznat */
       int i=0;
       double sum=0.0,x;

       printf(" Upisite niz brojeva !=0, i nulu za kraj.\n");
       printf(" x[0]= "); scanf("%lf",&x);
       while (x!=0.0){
          sum+=x;
          printf(" x[%d]= ",++i);
          scanf("%lf",&x);
       }
       sum/=i;
       printf(" Srednja vrijednost = %f\n",sum);
       return 0;
  }
```

Slide 15

Petlja `for`

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

ekvivalentna je s

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

Beskonačna petlja koja ne radi ništa:

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

Slide 16

```
#include <ctype.h>

/* atoi : prevodi string s[] u cijeli broj */
int atoi(char s[])
{
   int i,n,sign;

   for(i=0;isspace(s[i]);i++) ;  /* preskace sve bjeline */
   sign=(s[i]=='-')?-1:1;
   if(s[i]=='+' || s[i]=='-') i++; /* preskoci predznak */
   for(n=0; isdigit(s[i]);i++)
        n=10*n+(s[i]-'0');
   return sign*n;
}
```

Slide 17

Funkcije u `<ctype.h>` vraćaju vrijednost tipa `int` različitu od nule ako je uvjet ispunjen, odnosno nulu ako nije:

```
int isalnum(int c)
int isalpha(int c)
int iscntrl(int c)
int isdigit(int c)
int isgraph(int c)
int islower(int c)
int isupper(int c)
int isspace(int c)
```

Funkcije konverzije

```
int tolower(int c)
int toupper(int c)
```

Slide 18

Programiranje (C) Ak. god. 2003/04

Zarez (,) koji separira dva izraza je operator. Izrazi separirani zarezom izračunavaju se slijeva na desno i rezultat čitavog izraza je vrijednost desnog izraza. Na primjer,

```
i=(i=3,i+4);
```

daje `i=7`. Operator zarez koristi se uglavnom u `if` naredbi.

```
#include <string.h>
/* invertiraj : invertiraj znakovni niz */
void invertiraj(char s[])
{
   int c,i,j;

   for(i=0,j=strlen(s)-1;i<j;i++,j--) {
     c=s[i]; s[i]=s[j]; s[j]=c;
   }
}
```

Slide 19

`do - while` petlja

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

`naredba` će se izvršavati sve dok `izraz` ima vrijednost istine. Za razliku od `while` petlje vrijednost izraza se kontrolira na kraju prolaza kroz petlju. Petlja se stoga izvršava barem jednom.

Primjer:

```
i=0;
do{
   printf("%d\n",i);
   i++;
} while (i<10);
```

Slide 20

M. Jurak pred4.tex

Naredba `break`

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

Primjer:

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

`while(1)` je beskonačna petlja. Iz nje se izlazi ukoliko se učita negativan broj.

Slide 21

Naredba `continue`

Naredba `continue` koristi se unutar `for`, `while` i `do-while` petlje. Nakon nailaska na `continue` preostali dio tijela petlje se preskače i program nastavlja sa sljedećim prolazom kroz petlju.

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

Sada nam u dijelu kôda koji obradjuje pozitivne brojeve treba neki drugi način izlaza iz petlje.

Slide 22

Naredba `goto`

`goto` naredba prekida sekvencijalno izvršavanje programa i nastavlja izvršavanje s naredbom koja je označena labelom koja se pojavljuje u `goto`. Oblik joj je

```
goto labela;
```

gdje je `label` identifikator koji služi za označavanje naredbe kojom se nastavlja program. Sintaksa je

```
labela: naredba;
```

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

Slide 23

Primjer

```
   scanf("%d",&i);
   while{i<=100){
     .......
     if (i<0) goto error;
     .........
     scanf("%d",&i);
   }
   .......
    /* detekcija greske */
error: {
          printf("Greska : negativna vrijednost!\n");
          exit(-1);
       }
```

Slide 24

Naredbe `break` i `continue` mogu se izvesti pomoću `goto` naredbe. Na primjer, kôd

```
for(...) {
  .......
  if (...) continue;
  ........
}
```

je ekvivalentan s

```
for(...) {
  .......
  if (...) goto cont;
  .......
  cont: ;
}
```

Slično vrijedi i za `continue` unutar `while` i `do-while` petlje.

Slide 25