

2. Podaci u C-u

Funkcija scanf()



autori:

Krunoslav Husak, dipl. ing.

Krunoslav Kulhavi, dipl. ing.

© TemplatesWise.com

Funkcija scanf()

- Funkcija `scanf()` očitava podatke s tipkovnice i pridružuje ih jednoj od varijabli u programu.
- Kao i `printf()` i ova funkcija koristi formatni string (kontrolni tekst) da bi opisala oblik i tip podatka koji se unosi u program.

Funkcija scanf()



- Formatni stringovi u funkciji scanf() koriste iste držače mjesta kao i funkcija printf().

- Npr., naredba:

```
scanf ("%d", &x) ;
```

očitava cijeli broj (*int*) s tipkovnice i pridružuje ga *int* varijabli x.

Funkcija scanf()



- Simbol & je adresni operator (*address-of operator*).
- Za razliku od printf() funkcije kojoj kao argument zadajemo ime varijable, scanf() zahtijeva memorijsku adresu varijable što je razlog korištenja & operatora.

Funkcija scanf()



- Jednom funkcijom `scanf()` može se unijeti više od jedne vrijednosti pomoću više držača mjesta u formatnom stringu i više imena varijabli:

```
scanf("%d %f", &x, &rata);
```

Funkcija scanf()



- Kada se pomoću ove funkcije unosi više varijabli u program, rabi se među njima prazno mjesto (white space).
- To može biti jedno mjesto razmaka, nova linija ili tipka *tab*.

Funkcija scanf()

- **Primjer:**

```
#include <stdio.h>
int main(void)
{
    int a;
    printf("Unesite a: ");
    scanf("%d", &a);
    return 0;
}
```

3. Kontrola tijeka programa

Petlje



for petlja



- *for* petlja je naredba koja izvršava blok naredbi određeni broj puta. Petljom se naziva jer se vraća na određeni dio programa više puta.
- *for* petlja ima ovakvu strukturu:

```
for (inicijalizacija brojača ; uvjet ; promjena vrijednosti)
{
    // naredbe
}
```

for petlja



- **Inicijalizacija brojača** je dio u kojem se postavlja inicijalna vrijednost kontrolne varijable petlje
- **Uvjet** je uvjet koji kontrolna varijabla mora zadovoljiti da bi se izvršio blok naredbi
- **Promjena vrijednosti** dio u kojem se mijenja stanje kontrolne varijable

for petlja



- Primjer 1.:

```
...  
int i;  
for (i = 0; i < 10; i++)  
{  
    printf("%d", i);  
}  
...
```

for petlja



- Primjer 2.:

```
...  
int i;  
for (i = 10; i > 0; i--)  
{  
    printf("%d", i);  
}  
...
```

for petlja



- Primjer 3.:

```
int i, j;
for (i = 0; i < 10; i++)
{
    for (j = 0; j < 10; j++)
    {
        printf("%d*d=%d", i, j, i*j);
    }
}
```

while petlja



- ***while*** petlja izvršava blok naredbi dok god se određeni uvjet ne ispuni. Ona ima sljedeći oblik:

```
while (uvjet)
{
    // naredbe
}
```

while petlja



- **To je petlja s provjerom uvjeta na vrhu**
- **Ne mora se izvršiti niti jedanput**
- uvjet može biti bilo koji C izraz, a naredba može biti jedna ili više njih (blok naredbi).
- Kada program dođe do *while* petlje, izraz uvjet se izvodi. Ako je uvjet *istinit*, blok naredbi se izvršava. Izvršenje se vraća na prvi korak odnosno na ponovno izvršenje uvjeta. Ako je uvjet *lažan* *while* petlja se prekida.

while petlja



- Primjer 1.:

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

Što bi se desilo kada bi izbacili *i++;* naredbu?

while petlja



- Primjer 2.:

```
int i;  
i = 10;  
while (i > 0)  
{  
    printf("%d\n", i);  
    i--;  
}
```

do...while petlja



- ***do...while*** petlja izvršava blok naredbi dok god se određeni uvjet ispunjava. Ona ima sljedeći oblik:

```
do  
{  
    // naredbe  
}  
while (uvjet)
```

do...while petlja



- Kad program dođe do ove petlje, događa se sljedeće:
 - izvršava se naredba (ili više njih)
 - ispituje se **uvjet**. Ako je on istinit (1) izvršenje se vraća na početak a ako je lažan (0) prekida se izvršenje petlje.

do...while petlja



- Naredbe koje se nalaze unutar ove petlje izvršavaju se bar jednom, zbog toga što je uvjet na **kraju** petlje.
- Za razliku, *for* i *while* petlje uvjet ispituju na početku petlje pa se naredbe u njima ne moraju izvršiti niti jednom.

petlje



- Zadatak 1.:
Kolika je vrijednost varijable **j** nakon što se izvrši slijedeći program

```
...  
int i, j;  
j = 0;  
for (i = 0; i < 5; i++)  
{  
    j = j + 2;  
}  
...
```

petlje



- Zadatak 2.:
Kolika je vrijednost varijable **j** nakon što se izvrši slijedeći program

```
...  
int i, j;  
j = 0;  
for (i = 5; i < 5; i++)  
{  
    j = j - 1;  
}  
...
```

petlje



- Zadatak 3.:
Kolika je vrijednost varijable *i* nakon što se izvrši slijedeći program


```
...  
int i = 0;  
do  
{  
    i++;  
}  
while (i < 5)  
...
```

petlje




- Zadatak 4.:
Kolika je vrijednost varijable *i* nakon što se izvrši slijedeći program

```
...  
int i = 5;  
do  
{  
    i++;  
}  
while (i < 5)  
...
```




3. Kontrola tijeka programa

Naredba *if*



© TemplatesWise.com



- Naredbe se u C programu izvode odozgo prema dolje. Naredbe za kontrolu programa modificira redoslijed izvođenja programa
- U osnovnom obliku if naredba izvršava naredbe ovisno o rezultatu uvjetnog izraza

if

```
if (izraz)
{
    naredbe;
}
```

- Ako izraz daje istinu (1) naredbe se izvršavaju a ako izraz daje laž (0) naredbe se ne izvršavaju

If (prošireni oblik) - primjer

```
if (izraz)
{
    //nar. se izvršavaju ako je izraz istina
}
else
{
    //nar. se izvršavaju ako je izraz laž
}
```

If (else if oblik)



```

if (uvjet 1)
    //nar. se izvršavaju ako je uvjet 1 istina
else if (uvjet 2)
    //nar. se izvršavaju ako je uvjet 2 istina
else if (uvjet 3)
    //nar. se izvršavaju ako je uvjet 3 istina
    .....
else if (uvjet n )
    //nar. se izvršavaju ako je uvjet n istina
  
```

if



- if naredba se često koristi s operatorima usporedbe (<, >, ==, !=, ...).

- To je kao da kažete:

“Izvrši slijedeću izjavu ili izjave samo ako je izraz istinit”

if



- Primjeri:
- ```
if (a > 10)
{
 a = 5;
}
```
- ```
if (a == 5)
{
    a = 6;
}
```

Naredba switch-case



- koristi se kod višestrukog grananja programa. Omogućava izbor
- jednog između više mogućih puteva daljnjeg izvođenja programa,

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


petlje



- Petlje (*for*, *while* i *do...while*) izvršavaju blok naredbi nikada, jednom ili više puta ovisno o uvjetu ispunjenja.
- Kako izaći iz petlje ranije ako želite veću kontrolu nad svojim programom?
- Naredbe: ***break*** i ***continue***

break



- Naredba ***break*** može se staviti jedino u tijelo (blok) petlje. Ona uzrokuje da program preskoči dio programskog koda