Funkcije

Slide 1

Funkcija je programska cjelina koja uzima neke ulazne podatke, izvršava odredjen niz naredbi i vraća rezultat svog izvršavanja.

Definicija funkcija ima oblik

```
tip_p ime_funkcije(tip_1 arg_1, ... ,tip_n arg_n)
{
      tijelo funkcije
}
```

gdje je `tip_p` tip podatka koji će funkcija vratiti kao rezultat svog izvršavanja. Unutar zagrada nalazi se deklaracija formalnih argumenata funkcije. Prvi argument, `arg_1`, je varijabla tipa `tip_1` itd. Deklaracije pojedinih argumenata medjusobno se odvajaju zarezima. Unutar vitičastih zagrada pojavljuje se tijelo funkcije koje se sastoji od deklaracija varijabli i izvršnih naredbi.

Slide 2

Programiranje (C) Ak. god. 2003/04

Naredba `return`

Funkcija vraća rezultat svog izvršavanja pomoću naredbe `return`. Opći oblik te naredbe je

```
return izraz;
```

Izraz se može staviti u oble zagrade ali to nije nužno. Vrijedi sljedeće pravilo:

- Funkcija može vratiti aritmetički tip, strukturu, uniju ili pokazivač ali ne može vratiti drugu funkciju ili polje.

Ako je tip izraza u naredbi `return` različit od tipa podatka koji funkcija vraća, izraz će biti konvertiran u tip podatka.

Slide 3

Primjer :

Sljedeća funkcija pretvara mala slova (engleske abecede) u velika.

```
char malo_u_veliko(char z) {
    char c;
    c= (z >= 'a' && z <= 'z') ? ('A' + z - 'a') : z;
    return c;
}
```

Poziv funkcije:

```
int main(void) {
    char malo, veliko;
    printf("Unesite malo slovo: ");  scanf("%c", &malo);
    veliko = malo_u_veliko(malo);
    printf("\nUneseno slovo = %c\n",veliko);
    return 0;
}
```

Slide 4

M. Jurak pred5.tex

Stvarni argument funkcije može biti izraz.

Primjer: funkcija sqrt iz <math.h>

```
double sqrt(double)
```

može biti pozvana na ovaj način:

```
double x,y;
.......
y=sqrt(2*x-3)
```

Tada y prima vrijednost $\sqrt{2x-3}$.

Slide 5

Višestruke return naredbe:

Ako se programski tok unutar funkcije grana, onda je moguće imati više return naredbi unutar iste funkcije.

Primjer:

```
char malo_u_veliko(char z)
{
    if(z >= 'a' && z <= 'z')
        return('A' + z - 'a');
    else
        return z;
}
```

Slide 6

Funkcija tipa void:

Kada funkcija ne vraća nikakvu vrijednost onda se za tip "vraćene vrijednosti" koristi ključna riječ void

Primjer:

```
void maximum(int x, int y)
{
    int z;
    z=(x>=y) ? x : y;
    printf("\nMaksimalna vrijednost =%d",z);
    return;
}
```

Slide 7

Funkcija bez argumenata:

```
tip_podatka f(void)
{
        tijelo funkcije
}
```

Ključna riječ void unutar zagrada označava da funkcija ne uzima argumente.

Poziv funkcije:

```
varijabla=f();
```

Zagrade su obavezne, one informiraju prevodilac da je simbol f ime funkcije.

Slide 8

Tijelo funkcije:

- Sastoji se od deklaracija varijabli i izvršnih naredbi.
- Deklaracije varijabli moraju prethoditi prvoj izvršnoj naredbi (standard C90).
- Standard C99 dozvoljava deklaraciju varijabli bilo gdje unutar tijela funkcije.

Slide 9

Deklaracija funkcije (1):

- Svaka bi funkcija prije svoga poziva u programu trebala biti *deklarirana*.
- Deklaracija funkcije govori koliko argumenata funkcija uzima, koji su njihovi tipovi i koji tip podatka funkcija vraća.
- Ako je funkcija definirana u istoj datoteci u kojoj se poziva i to prije svog prvog poziva, onda definicija služi kao deklaracija te posebna deklaracija nije potrebna.

Primjer:

Slide 10

Programiranje (C) Ak. god. 2003/04

```
#include <stdio.h>
void maximum(int x, int y)
{    int z;
     z=(x>=y) ? x : y;
     printf("\nMaksimalna vrijednost =%d\n",z);
}
int main(void)
{
    int x,y;

    printf("Unesite dva cijela broja: ");
    scanf("%d %d", &x,&y);
    maximum(x,y);
    return 0;
}
```

Slide 11

Deklaracija funkcije (2):

- Ukoliko je funkcija koju koristimo definirana iza mjesta na kome se poziva, moramo ju na početku datoteke deklarirati.

Deklaracija (ili prototip) funkcije ima oblik

```
tip_podatka ime_funk(tip_1 arg_1, ... ,tip_n arg_n);
```

- Imena argumenata mogu biti izostavljena.

Primjer:

Slide 12

M. Jurak pred5.tex

```
#include <stdio.h>
int main(void)
{   int x,y;
    void maximum(int x, int y);

    printf("Unesite dva cijela broja: ");
    scanf("%d %d", &x,&y);
    maximum(x,y);
return 0;
}

void maximum(int x, int y) {
    .........
}
```

Slide 13

ili

```
#include <stdio.h>
void maximum(int, int);

int main(void)
{   int x,y;
    printf("Unesite dva cijela broja: ");
    scanf("%d %d", &x,&y);
    maximum(x,y);
return 0;
}
void maximum(int x, int y) {
     ...........
}
```

Slide 14

Prijenos argumenata:

- Argumenti deklarirani u definiciji funkcije nazivaju se formalni argumenti.
- Izrazi koji se pri pozivu funkcije nalaze na mjestima formalnih argumenata nazivaju se stvarni argumenti.
- Prilikom poziva funkcije stvarni argumenti se izračunavaju (ako su izrazi) i kopiraju u formalne argumente. Funkcija prima kopije stvarnih argumenata što znači da ne može izmijeniti stvarne argumente.

Primjer:

Slide 15

```
#include <stdio.h>
void f(int x)
{ x+=1; printf("\nUnutar funkcije x=%d",x);
}
int main(void) {
    int x=5;
    printf("\nIzvan funkcije x=%d",x);
    f(x);
    printf("\nNakon poziva funkcije x=%d",x);
    return 0;
}
```

Rezultat izvršavanja programa će biti:

```
Izvan funkcije x=5
Unutar funkcije x=6
Nakon poziva funkcije x=5
```

Slide 16

Pravila:

- Broj stvarnih argumenata pri svakom pozivu funkcije mora biti jednak broju formalnih argumenata.
- Ako je funkcija ispravno deklarirana, tako da prevodilac pri pozivu funkcije zna broj i tip formalnih argumenata, tada se stvarni argumenti čiji se tip razlikuje od odgovarajućih formalnih argumenta konvertiraju u tip formalnih argumenata isto kao pri pridruživanju.
- Redosljed izračunavanja stvarnih argumenat nije definiran i može ovisiti o implemetaciji.

Slide 17

Primjer:

```
 int f(double);
 int main(void)
 {
     float x=2.0;
     printf("%d\n",f(2)); /* int -> double */
     printf("%d\n",f(x)); /* float -> double */
     return 0;
 }

int f(double x) {
   return (int) x*x;
}
```

Slide 18

Funkcije bez prototipa;

U programu se mogu koristiti i funkcije koje nisu prethodno deklarirane. U tom slučaju vrijedi:

- Prevodilac pretpostavlja da funkcija vraća podatak tipa `int` i ne pravi nikakve pretpostavke o broju i tipu argumenata.
- Na svaki stvarni argument cjelobrojnog tipa primijenjuje se integralna promocija (konverzija argumenata tipa `short` i `char` u `int`), a svaki stvarni argument tipa `float` konvertira se u tip `double`. Nakon toga broj i tip (konvertiranih) stvarnih argumenta mora se podudarati s brojem i tipom formalnih argumenata da bi poziv funkcije bio korektan.

Primjeri:

Slide 19

```
 int main(void)
 {
     float x=2.0;
     printf("%d\n",f(2)); /* greska */
     printf("%d\n",f(x)); /* O.K.   */
     return 0;
 }

int f(double x) {
   return (int) x*x;
}
```

Slide 20

```c
int main(void)
{
    float x=2.0;
    printf("%d\n",f(2));
    printf("%d\n",f(x));
    return 0;
}

double f(double x) {
/* greska: redefinicija simbola f */
   return  x*x;
}
```

Slide 21

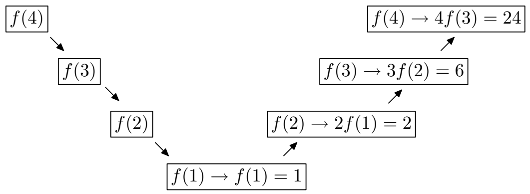
Rekurzivne funkcije

C dozvoljava da se funkcije koriste rekurzivno, odnosno da pozivaju same sebe. Na primjer za računanje $n!$ možemo napisati rekurzivnu funkciju

```c
long faktorijeli(long n) {
     if(n<=1) return 1;
     else return n*faktorijeli(n-1);
}
```

Funkcija (rekurzivno) poziva samu sebe $n-1$ puta kako bi izračunala $n!$.

Slide 22



Slide 23

Primjer:

```c
#include <stdio.h>
 void unos(void){
    char znak;
    if((znak=getchar())!='\n') unos();
    putchar(znak);
 }
int main(void) {
    printf("\n Unesite niz znakova: ");
    unos(); return 0;
}
```

dobit ćemo ovaj rezultat:

```
 Unesite niz znakova: Zdravo

ovardZ
```

Slide 24

Funkcije s varijabilnim brojem argumenata:

Funkcije `scanf` i `printf` primaju varijabilan broj argumenata. Datoteka zaglavlja `<stdarg.h>` sadrži niz definicija i makro naredbi koje nam omogućavaju pisanje funkcija s varijabilnim brojem argumenata.

vidi: Kernighan-Ritchie : The C Programming Language ...

Slide 25