Funkcije

nastavak

Funkcije

- Nakon odslušanog bit ćete u stanju:
  - objasniti načine prijenosa argumenata: prijenos po
    - vrijednosti (*call by value*)
    - adresi (*call by address*)
  - dati primjere korištenja
  - objasniti pojam
    - rekurzije
    - rekurzivne funkcije
  - nabrojati prednosti i ograničenja rekurzivnih funkcija.

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.
- 1. Funkcija prima kopije vrijednosti stvarnih argumenata, što znači da ne može izmijeniti stvarne argumente.

Primjer: prijenos po vrijednosti

```
#include <stdio.h>
void f(int x){
    x += 1;
    printf("Unutar funkcije x = %d", x);
}

int main(){
    int x = 5;
    printf("Prije poziva funkcije x = %d", x);
    f(x);
    printf("Nakon poziva funkcije x = %d", x);
    return 0;
}
```

Prijenos argumenata (2)

- 2. Funkcija prima adrese stvarnih argumenata, što znači da može izmijeniti stvarne argumente, tj. sadržaje na tim adresama.

Primjer: prijenos po adresi

```
#include <stdio.h>
void f(int *x){
    *x += 1;
    printf("Unutar funkcije x = %d", *x);
}
```

Primjer: prijenos po adresi

```
int main(){
    int x = 5;
    printf("Prije poziva funkcije x = %d", x);
    f(&x);
    printf("Nakon poziva funkcije x = %d", x);
    return 0;
}
```

Primjer:

```
#include <stdio.h>
void f(int x, int y){
    x += y; y += x;
    printf("Unutar funkcije x = %d, y = %d", x, y);
}

int main(){
    int x = 2, y = 3;
    printf("Prije poziva funkcije x = %d, y = %d", x, y);
    f(y, x + y);
    printf("Nakon poziva funkcije x = %d, y = %d", x, y);
    return 0;
}
```

Primjer: (2)

```
#include <stdio.h>
void f(int *x, int *y){
    *x += *y; *y += *x;
    printf("Unutar funkcije x = %d, y = %d", *x, *y);
}

int main(){
    int z, x = 2, y = 3;
    z = x + y;
    printf("Prije poziva funkcije x = %d, y = %d", x, y);
    f(&y, &z);
    printf("Nakon poziva funkcije x = %d, y = %d", x, y);
    return 0;
}
```

Pravila

- Broj stvarnih argumenata pri svakom pozivu funkcije mora biti jednak broju formalnih argumenata.
- Ako je funkcija ispravno deklarirana (prevodilac pri pozivu zna broj i tip argumenata) stvarni argumenti čiji se tip razlikuje od odgovarajućih formalnih argumenata pretvaraju se u tip formalnih argumenata isto kao i pri pridruživanju.
- Redoslijed izračunavanja stvarnih argumenata nije definiran i ovisi o implementaciji.

Rekurzija

- Rekurzivni proces (rekurziju) najčešće definiramo pomoću rekurzivne relacije:

  `f(0)=1` (inicijalizacija rekurzije)
  `f(n)=n*f(n-1),n≥1.`

- f(4)=4*f(3)=4*3*f(2)=4*3*2*f(1)=4*3*2*1*f(0)=4*3*2*1*1
- Redoslijed izračunavanja:
  1. 1 * 1 = 1
  2. 2 * 1 = 2
  3. 3 * 2 = 6
  4. 4 * 6 = 24.

Rekurzija (2)

- Neke poznate rekurzivne relacije:

$f(0)=0$
$f(n)=n+f(n-1), n \geq 1$

$f(0)=0,\ f(1)=1$ (inicijalizacija rekurzije)
$f(n)=f(n-1)+f(n-2), n \geq 2$

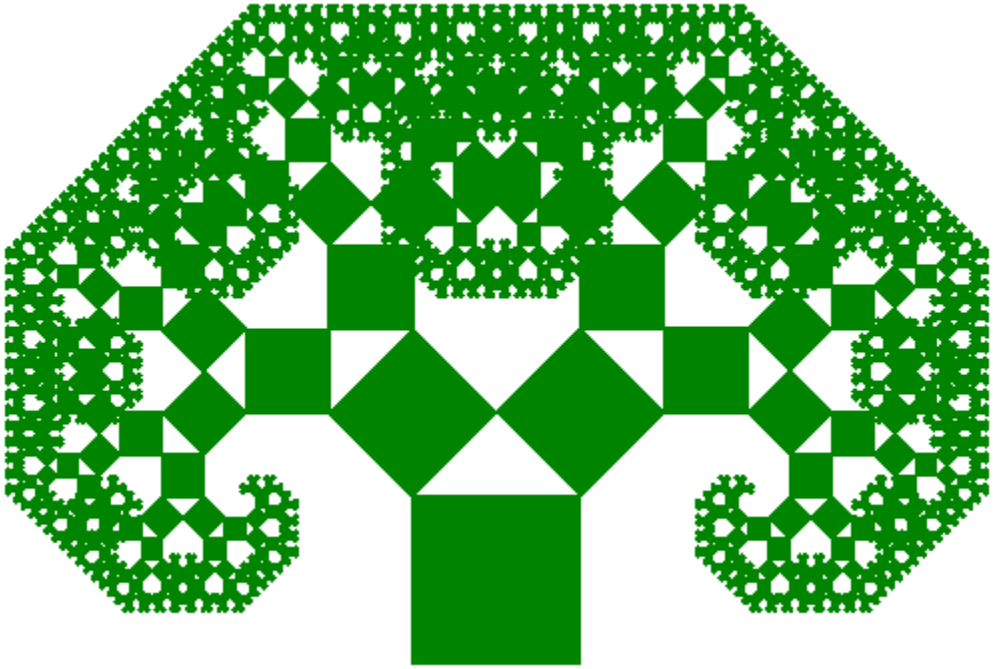
$f(0)=0$
$f(n)=f(n-1)+1+f(n-1), n \geq 1.$

Rekurzija (3)

- Primjer vizualne rekurzije:

Rekurzija (4)

- Pitagorino stablo:



Rekurzija (5)

- Primjer jezične rekurzije:

Dječak nije mogao spavati pa mu je majka ispričala priču o žabici,
koja nije mogla spavati pa joj je majka ispričala priču o medi,
koji nije mogao spavati pa mu je majka ispričala priču o maci
…
i maca je zaspala;
i medo je zaspao;
i žabica je zaspala;
i dječak je zaspao.

Rekurzivne funkcije

- Rekurzivne funkcije:
  - u definiciji funkcije koristimo funkciju koju definiramo.
- Rekurzivno rješenje je često
  - elegantnije (kraće).
- Može biti
  - sporije
  - memorijski zahtjevnije.

Rekurzivne funkcije (2)

- Programski jezik C dozvoljava rekurzivne funkcije.

  Primjer: funkcija koja rekurzivno računa n!

```
int faktorijeli(int n)
{
    if(n == 0) return 1;
    else return n * faktorijeli(n - 1);
}
```

Rekurzivne funkcije (3)

Primjer: funkcija koja rekurzivno ispisuje cijeli broj n.

```
void pd(int n)
{
    if(n < 0){
        putchar('-');
        n = -n;
    }
    if(n/10)
        pd(n/10);
    putchar(n%10 + '0');
}
```

Primjer:

```
#include <stdio.h>
void f(void)
{
   char c;
   if((c = getchar()) != '\n')
   f( );
   putchar(c);
}
int main( )
{
   f( );
   return 0;
}
```

Funkcije s varijabilnim brojem arg.

- Funkcije `scanf` i `printf` imaju varijabilni 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* …

Primjer:

```
void f(){
}

int main(void){

   f(2); f('7', 8); f("abc");
   return 0;
}
```

```
void f(void){
}

int main(void){

   f(2); f('7', 8); f("abc");
   return 0;
}
```

error: too many arguments to function 'f'