
Rekurzivne funkcije

nastavak

Primjer:

```
int f(unsigned n)
{
    if (n == 0) return 2;
    else return f(--n); /* f(n--) */
}
```

```
int main(void)
{
    printf("%d\n", f(4));

    return 0;
}
```

Broj particija (rastava) broja n

n = 4	n = 5
4	5
1 + 3	1 + 4
2 + 2	2 + 3
1 + 1 + 2	1 + 1 + 3
1 + 1 + 1 + 1	1 + 2 + 2
-	1 + 1 + 1 + 2
-	1 + 1 + 1 + 1 + 1
$p(4) = 5$	$p(5) = 7$

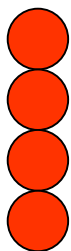
/*

Program učitava prirodni broj n i ispisuje broj particija broja n, tj. broj rastava broja n u sumu nepadajućih prirodnih brojeva.

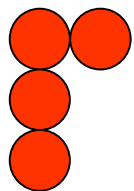
Osnovni rekurzivni algoritam zbrajanjem.

*/

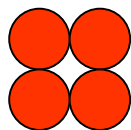
Broj rastava broja n (2)



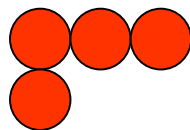
4



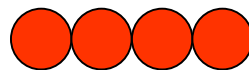
3 + 1



2 + 2



2 + 1 + 1



1 + 1 + 1 + 1

$$(1+x+x^2+x^3+x^4+\dots) (1+x^2+x^4+\dots) (1+x^3+x^6+\dots) (1+x^4+\dots)\dots = \dots p(4)x^4+\dots$$

$$p(100) = 190\,569\,292$$

Funkcija particije

```
#include <stdio.h>
```

```
int particije(int suma, int prvi)
```

```
{
```

```
    int i, broj = 0;
```

```
    if (suma == 0) return 1;
```

```
    /* else */
```

```
    for (i = prvi; i <= suma; ++i)
```

```
        /* Rekurzivni poziv - dodavanje sljedećeg pribrojnika */
```

```
        broj += particije( suma - i, i );
```

```
    return broj;
```

```
}
```

Funkcija main

```
int main(void)
{
    int n;

    printf(" Upiši prirodni broj n: ");
    scanf("%d", &n);

    printf("\n Broj particija p(%d) = %d.\n",
           n, particije(n, 1) );

    return 0;
}
```

Pozivi funkcije

Upiši prirodni broj n : 4

suma - $i = 3$, $i = 1$

suma - $i = 2$, $i = 1$

suma - $i = 1$, $i = 1$

suma - $i = 0$, $i = 1$

suma - $i = 0$, $i = 2$

suma - $i = 1$, $i = 2$

suma - $i = 0$, $i = 3$

suma - $i = 2$, $i = 2$

suma - $i = 0$, $i = 2$

suma - $i = 1$, $i = 3$

suma - $i = 0$, $i = 4$

```
for (i = prvi; i <= suma; ++i)
{
    printf(" suma - i = %d, i = %d\n", suma - i, i);
    broj += particije( suma - i, i );
}
```

Broj particija $p(4) = 5$.

Hanojski tornjevi

- Problem: Na štapu S (izvor, *source*) naslagano je n diskova različite veličine tako da veći disk nikada ne dolazi iznad manjeg.
Potrebno je, koristeći treći štap T (pomoćni, *temp*), uz minimalni broj operacija, preseliti sve diskove na štap D (odredište, *destination*), jedan po jedan. Pri tome se svaki disk smije postaviti ili na prazan štap ili na disk veći od njega.
 - Animacija:
<https://www.mathsisfun.com/games/towerofhanoi.html>
-

Algoritam

Preseliti (uz zadana ograničenja):

- $n-1$ disk sa štapa S na štap T koristeći štap D kao pomoćni;
 - najveći disk sa štapa S na štap D;
 - $n-1$ disk sa štapa T na štap D koristeći štap S kao pomoćni.
-
- Neka je a_n ukupni broj koraka (preseljenja). Tada je
 $a_1 = 1$, /* $a_0 = 0$ */
 $a_n = a_{n-1} + 1 + a_{n-1} = 2a_{n-1} + 1, n \geq 2.$

Funkcija hanojski_tornjevi

/* Funkcija vraća ukupni broj koraka. */

```
int hanojski_tornjevi(int n, char s, char d, char t)
{
    if (n == 1) return 1;
    else return
        hanojski_tornjevi(n-1, s, t, d)+
        hanojski_tornjevi(1, s, d, t)+
        hanojski_tornjevi(n-1, t, d, s);
}
```

Funkcija hanojski_tornjevi (2)

- Ukoliko oznake štapova nisu argumenti funkcije, tada rekurziju za ukupni broj koraka možemo jednostavnije zapisati na sljedeći način:

```
int hanojski_tornjevi(int n)
{
    if (n == 1) return 1;
    else return 2 * hanojski_tornjevi(n-1) + 1;
}
```

Struktura programa

Blokovska struktura programa

- Jedan blok naredbi čini svaki niz naredbi omeđen vitičastim zagradama (npr. tijelo funkcije).
 - Programski jezik C dozvoljava da se u svakom bloku deklariraju varijable.
 - Varijabla deklarirana unutar nekog bloka vidljiva je samo unutar tog bloka.
 - Varijabla deklarirana izvan bloka vidljiva je u unutarnjem bloku ako u njemu nije definirana varijabla istog imena.
-

Primjer: Ugniježđeni blokovi

```
{
    int a = 1, b = 2;
    printf ("vanjski blok: a = %d\n",a);
    printf ("vanjski blok: b = %d\n",b);
    {
        int a = 10; b=6;
        printf ("unutarnji blok: a = %d\n",a);
        printf ("unutarnji blok: b = %d\n",b);
    }
    printf ("vanjski blok: a = %d\n",a);
    printf ("vanjski blok: b = %d\n",b);
}
```

Atributi varijabli

- Sve varijable imaju tri atributa: tip, memorijski klasu i doseg.
 - Prema tipu imamo varijable tipa `int`, `float`, `char` itd.
 - Memorijska klasa (smještajni razred, eng. *storage class*) određuje kako će varijabla biti pohranjena u memoriji. Identifikatori (oznake) memorijske klase su ključne riječi: `auto`, `register`, `static` i `extern`.
 - Po dosegu (područje važenja, eng. *scope*) varijable se dijele na **vanjske** (globalne) i **unutarnje** (lokalne).
-