Programiranje 2

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

March 26, 2007

Sadržaj

1

Programski jezik C

[1]

1.1 Rekurzija

Rekurzija je postupak rešavanja zadataka, u kome neka funkcija (direktno ili indirektno) poziva samu sebe. Pri tome se vodi računa da se svaki dublji poziv izvršava za jednostavniji problem od polaznog, a da se najjednostavniji problemi rešavaju direktno, tj. bez dalje upotrebe rekurzije (jednostavnost problema se definiše na pogodan način, prema prirodi samog problema, a najčešće kao veličina ulaznog argumenta rekurzivne funkcije).

Pri rešavanju zadatka, rekurziju je najbolje shvatiti i koristiti na sledeći način: Potrebno je da umemo da neposredno rešimo najjednostavniji slučaj datog problema, i da složenije slučajeve svedemo na jednostavnije, tj. da ih rešimo koristeći jednostavnije slučajeve. Pri tome ne treba da brinemo o tome kako će biti rešeni ti jednostavniji slučajevi na koje se složeni slučaj svodi - tu rekurzija radi za nas.

Primer 1 *Rekurzivna i iterativna varijanta funkcije koja stepenuje ceo broj na celobrojni pozitivan izlozilac.*

```
#include <stdio.h>

int stepen(int x, int k)
{

    printf("Racunam stepen(%d, %d)\n",x,k); /* Ilustracije radi! */

    if (k == 0)
        return 1;
    else
        return x*stepen(x, k-1);


    /* Krace se moze zapisati i kao

    return k==0 ? 1 : x*stepen(x, k-1); */
```

[1] Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~filip`, i zbirke zadataka "Programiranje i programiranje", Milan Vugdelija

```
}

/* Druga verzija prethodne funkcije.
Obratiti paznju na efikasnost u odnosu na prvu verziju!*/
int stepen2(int x, int k)
{
    printf("Racunam stepen2(%d, %d)\n",x,k); /* Ilustracije radi! */
    if (k == 1)
        return x;
    else
         /* Ako je stepen paran*/
        if((k%2)==0)
            return stepen2(x*x, k/2);
        else
            return x * stepen2(x*x, k/2);


}

/* Iterativna verzija prethodne funkcije */
int stepen_iterativno(int x, int k)
{
    int i;
    int s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return s;
}


/* Iterativna verzija prethodne funkcije */
int stepen_iterativno(int x, int k)
{
    int i;
    int s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return s;
}


main()
{
    printf("%d", stepen2(2, 8));
    printf("\n------------------\n");
    printf("%d", stepen(2, 8));
}

/* Izlaz iz programa:
```

```
Racunam stepen2(2, 8)
Racunam stepen2(4, 4)
Racunam stepen2(16, 2)
Racunam stepen2(256, 1)
256
------------------
Racunam stepen(2, 8)
Racunam stepen(2, 7)
Racunam stepen(2, 6)
Racunam stepen(2, 5)
Racunam stepen(2, 4)
Racunam stepen(2, 3)
Racunam stepen(2, 2)
Racunam stepen(2, 1)
Racunam stepen(2, 0)
256
*/
```

Primer 2 *Rekurzivna i iterativna varijanta računanja Fibonačijevih brojeva.*

```
/* Fibonacijev niz brojeva se definise kao niz u kome su prva dva broja
   jednaka 1, a za ostale brojeve vazi da je svaki sledeci zbir dva
   prethodna. tj.
    f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2)
*/

#include <stdio.h>
#include <stdlib.h>

/* Rekurzivna implementacija - obratiti paznju na neefikasnost funkcije */
int Fib(int n)
{
    printf("Racunam Fib(%d)\n",n); /* Ilustracije radi! */

    if((n==0)||(n==1))
        return 1;
    else
        return(Fib(n-1)+Fib(n-2));

    /* Krace moze kao:
    return (n == 0 || n == 1) ? 1 : Fib(n-1) + Fib(n-2); */
}


/* Iterativna verzija bez niza */
int Fib_iterativno(int n)
{
    /* Promenljiva pp cuva pretposlednji, a p poslednji element niza */
    int pp = 1, p = 1;
    int i;
    for (i = 0; i <= n-2; i++)
```

```
    {
        int pom = pp;
        pp = p;
        p = p + pom;
    }
    return p;
}


main()
{
    printf("Fib(5) = %d\n", Fib(5));

}

/* Izlaz iz programa:
Racunam Fib(5)
Racunam Fib(4)
Racunam Fib(3)
Racunam Fib(2)
Racunam Fib(1)
Racunam Fib(0)
Racunam Fib(1)
Racunam Fib(2)
Racunam Fib(1)
Racunam Fib(0)
Racunam Fib(3)
Racunam Fib(2)
Racunam Fib(1)
Racunam Fib(0)
Racunam Fib(1)
Fib(5) = 8
*/
```

Primer 3 *Rekurzivna i iterativna varijanta računanja faktorijela prirodnog broja.*

```
unsigned long faktorial(int n)
{

    if (n == 1)
        return 1;
    else
        return n*faktorial(n-1);


    /* Krace moze kao:
    return n == 1 ? 1 : n*faktorial(n-1); */
}

/* Iterativna verzija prethodne funkcije */
unsigned long faktorial_iterativno(int n)
{
```

```
    long f = 1;
    int i;
    for (i = 1; i<=n; i++)
        f *= i;
    return f;
}

main()
{
    printf("5! = %d\n", faktorial(5));
}
```

Primer 4 *Rekurzivna i iterativna varijanta računanja sume niza celih brojeva.*

```
int suma_niza(int a[], int n)
{
if (n == 1)
return a[0];
else
return suma_niza(a, n-1)+a[n-1];

/* Krace moze kao:
return n == 1 ? a[0] : suma_niza(a, n-1)+a[n-1];*/
}

/* Iterativna verzija prethodne funkcije */
int suma_niza_iterativno(int a[], int n)
{
int suma = 0;
int i;
for (i = 0; i<n; i++)
suma+=a[i];
return suma;
}
```

Primer 5 *Štampanje celog broja.*

```
#include<stdio.h>
void print_broj(long int n)
{
if(n<0)
{
putchar('-');
n=-n;
}
if(n/10)
print_broj(n/10);
putchar(n % 10 + '0');
}
int main()
{
```

```
long int b=-1234;
print_broj(b);
putchar('\n');
return 0;
}
```

Primer 6 *Rekurzivna varijanta binarne pretrage niza celih brojeva.*

```
#include <stdio.h>

/* Funkcija proverava da li se element x javlja unutar niza
   celih brojeva a.
   Funkcija vraca poziciju na kojoj je element nadjen odnosno
   -1 ako ga nema.

   !!!!! VAZNO !!!!!
   Pretpostavka je da je niz a uredjen po velicini
*/

int binarna_pretraga(int a[], int l, int d, int x)
{
    /* Ukoliko je interval prazan, elementa nema */
    if (l > d)
        return -1;


    /* Srednja pozicija intervala [l, d] */
    int s = (l+d)/2;

    /* Ispitujemo odnos x-a i srednjeg elementa */
    if (x == a[s])
        /* Element je pronadjen */
        return s;
    else if (x < a[s])
        /* Pretrazujemo interval [l, s-1] */
        return  binarna_pretraga(a, l, s-1, x);
    else
        /* Pretrazujemo interval [s+1, d] */
        return  binarna_pretraga(a, s+1, d, x);
}

main()
{
    int a[] = {3, 5, 7, 9, 11, 13, 15};
    int x;
    int i;

    printf("Unesi element kojega trazimo : ");
    scanf("%d",&x);
    i = binarna_pretraga(a, 0, sizeof(a)/sizeof(int)-1, x);
```

```
    if (i==-1)
        printf("Elementa %d nema\n", x);
    else
        printf("Pronadjen na poziciji %d\n", i);
}
```

Primer 7 *Sortiranje niza celih brojeva - QuickSort algoritam*

```
#include<stdio.h>

/* Funkcija menja mesto i-tom i j-tom elementu niza a */
void razmeni(int a[], int i, int j)
{
    int pom = a[i];
    a[i] = a[j];
    a[j] = pom;
}

/* Funkcija sortira deo niza brojeva a izmedju pozicija l i d*/
void quick_sort(int a[], int l, int d)
{
    int i, poslednji, pivot;

    /* Ukoliko je interval [l, d] prazan nema nista da se radi */
    if (l >= d)
        return;

     /* Srednji element uzimamo za pivot i postavljamo ga na pocetak */
    razmeni(a, l, (l + d)/2);

    /* Niz organizujemo tako da pivot postavimo na pocetak, zatim da iza
       njega budu svi elementi manji od njega, pa zatim svi elementi koji
       su veci od njega  tj. pivot < < < < > > > >*/

    /* poslednji je pozicija poslednjeg elementa niza za koji znamo da je
       manji od pivota. U pocetku takvih elemenata nema */
    poslednji = l;
    for (i = l+1; i <= d; i++)
        /* Ukoliko je element na poziciji i manji od pivota,
           postavljamo ga iza niza elemenata manjih od pivota,
           menjajuci mu mesto sa prvim sledecim */
        if (a[i] < a[l])
            razmeni(a, ++poslednji, i);

    /* Zahtevanu konfiguraciju < < < < piv > > > > dobijamo tako
       sto zamenimo mesto pivotu i poslednjem elementu manjem od njega */
        razmeni(a, l, poslednji);

    /* Pivot se sada nalazi na poziciji poslednji */
    pivot = poslednji;

    /* Rekurzivno sortiramo elemente manje od pivota */
```

```
    quick_sort(a, l, pivot-1);
    /* Rekurzivno sortiramo elemente vece pivota */
    quick_sort(a, pivot+1, d);
}


main()
{
    int a[] = {5, 8, 2, 4, 1, 9, 3, 7, 6};
    int n = sizeof(a)/sizeof(int);
    int i;

    quick_sort(a, 0, n-1);

    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");

}
```

Zadaci za vežbu:

Zadatak 1 *Napisati program koji upotrebom rekurzivne funkcije ispituje da li je dati string palindrom.*

Zadatak 2 *Za dati broj pomoću rekurzivne funkcije ispisati broj koji se piše istim ciframa ali u obrnutom poretku.*

Zadatak 3 *Napisati rekurzivnu funkciju koja vraća odgovor na pitanje da li je broj cifara njenog argumenta paran.*

Zadatak 4 *Napisati rekurzivnu funkciju koja računa zbir cifara na neparnim pozicijama, računajući skraja (sdesna nalevo).*