Programiranje II

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

13.03.08.

Sadržaj

1 Karakterske niske

1. Napisati funkciju `char* string_string(char* str, char* sub)` koja proverava da li se string `sub` nalazi u stringu `str` i vraća `NULL` ako se ne nalazi odnosno pokazivač na prvo pojavljivanje ako se nalazi.

```
char* string_string(char *str, char *sub)
{
    char *s, *t;

    /* Proveravamo da li sub pocinje redom na poziciji str, str+1,... */
    for (; *str; str++)
        /* Poredimo sub sa str pocevsi sve dok ne naidjemo na razliku */
        for (s = str, t = sub; *s == *t; s++, t++)
            /* Ako nismo naisli na razliku a ispitali smo sve karaktere
               niske sub */
            if (*(t+1) == '\0')
                return str;

    /* Nije nadjeno */
    return NULL;
}

main()
{
    char r[] = "racunari";

    printf("%d\n",string_string(r, "rac") - r);
    printf("%d\n",string_string(r, "ari") - r);
    printf("%d\n",string_string(r, "cun") - r);
    printf("%p\n",string_string(r, "cna"));
}
```

2 Pokazivači na funkcije

1. Napisati funkciju `sumiraj` koja računa sumu:

   `f(1)+f(2)+...+f(n)` za proizvoljno n koje se unosi sa standardnog ulaza i proizvoljnu funkciju sa potpisom `int f(int)`.

```
/* Program demonstrira upotrebu pokazivaca na funkcije */
#include <stdio.h>

/* Navodimo definicije nekoliko funkcija ciji je potpis
   int f(int); */
int kvadrat(int n)
```

```
{
    return n*n;
}

int kub(int n)
{
    return n*n*n;
}

int parni_broj(int n)
{
    return 2*n;
}


/* Funkcija izracunava sumu f(1) + f(2) + ... + f(n), gde je funkcija f
   data kao prvi argument funkcije sumiraj. Prvi argument funkcije
   sumiraj je tipa:

        int (*f) (int)

   sto je pokazivac na funkciju koja ima jedan argument tipa int i
   vraca vrednost tipa int.
*/
int sumiraj(int (*f) (int), int n)
{
    int i, suma;

    suma = 0;
    for (i=1; i<=n; i++)
        suma += (*f)(i);

    return suma;
}

main()
{
   /* Prilikom prosledjivanja parametara funkciji koja ocekuje
      pokazivac na drugu funkciju, dovoljno je navesti ime funkcije
      koja se poziva: sumiraj(kvadrat,3) */

   printf("Suma kvadrata brojeva od jedan do 3 je %d\n",
          sumiraj(kvadrat,3));
   printf("Suma kubova brojeva od jedan do 3 je %d\n",
          sumiraj(kub,3));
   printf("Suma prvih pet parnih brojeva je %d\n",
```

```
            sumiraj(parni_broj,5));
}
```

3 Dinamička alokacija memorije, funkcije *malloc* i *calloc*

1. Napisati program koji sa standardnog ulaza unosi prvo dimenziju niza pa zatim i elemente niza. Memoriju za niz alocirati dinamički, korišćenjem funkcije *malloc*. Nakon toga odrediti maksimalni element niza i sumu elemenata na parnim pozicijama.

```
/* Demonstracija dinamicke alokacije memorije koriscenjem funkcije malloc */
/* Program unosi niz proizvoljne dimenzije i nalazi najveci element niza
   i sumu elemenata na parnim pozicijama. */

#include <stdio.h>

/* Neophodno je ukljuciti zaglavlje "stdlib.h" zbog funkcije malloc */
#include <stdlib.h>

main()
{
    int n;

    /* Deklaracija
        int a[n];
    nije dozvoljena jer kompilator ne moze u vreme prevodjenja
    da odredi potrebnu kolicinu memorije. Umesto ovoga, memoriju cemo
    alocirati dinamicki tj. u fazi izvrsavanja programa kada bude poznata
    vrednost broja n. Zbog toga je potrebno upamtiti samo adresu pocetka
    alociranog bloka memorije sto cemo uraditi koriscenjem sledeceg
    pokazivaca:
    */

    int* a;

    int i, max, suma;

    printf("Unesi dimenziju niza : ");
    scanf("%d", &n);

    /* U ovom treunutku znamo koliko nam je memorije potrebno i pozivamo
       funkciju malloc za dinamicku alokaciju. malloc-u se prosledjuje
       kolicina potrebne memorije u bajtovima, a on vraca genericki
```

```
   pokazivac (void*) koji pre koriscenja treba konvertovati u
   odgovarajuci realni pokazivacki tip.
*/

a = (int*) malloc(n*sizeof(int));

/* U slucaju da nema dovoljno memorije malloc vraca NULL */
if (a == NULL)
{
    printf("Greska : Nema dovoljno memorije!\n");
    return 1;
}

/* Nadalje a koristimo kao obican niz */
for (i = 0; i<n; i++)
{
    printf("a[%d]=", i);
    scanf("%d", &a[i]);
}

/* Nalazimo maksimum */
max = a[0];
for (i = 1; i<n; i++)
    if (a[i] > max)
        max = a[i];

printf("Najveci element je %d\n", max);

/* Nalazimo sumu elemenata na parnim pozicijama.
   Brojac povecavamo za 2 da bismo sabrali samo elemente niza na
   pozicijama 0, 2, 4... */
suma = 0;
for(i=0; i<n; i+=2)
    suma += a[i];

/* Duzni smo da rucno alociranu memoriju rucno i oslobodimo */
free(a);
}
```

2. Ilustracija vraćanja niza iz funkcije. Dinamički alocirana memorija može biti povratna vrednost funkcije (za razliku od statički alocirane memorije).

```
/* Vracanje niza iz funkcije */
/* Program unosi niz proizvoljne dimenzije i nalazi najveci element */
```

```
#include <stdio.h>

/* Neophodno je ukljuciti stdlib.h */
#include <stdlib.h>

/* Moguce je vratiti pokazivac na dinamicki alocirani niz jer se on
   alocira na hip-u i taj prostor je rezervisan i posle zavrsetka rada
   funkcije */
int* CreateIntArray(int n)
{
    return (int*) malloc(n*sizeof(int));
}

/* Nije moguce vratiti pokazivac na lokalni niz jer se on alocira na steku
   i taj prostor nije rezervisan posle zavrsetka rada funkcije */
int* Create10ElementIntArray()
{
    int a[10];
    return a;
}

main()
{
    int n;

    int* a;
    int i, max;

    printf("Unesi dimenziju niza : ");
    scanf("%d", &n);

    /* Alociramo memoriju unutar funkcije pozivom funkcije malloc */
    a = CreateIntArray(n);

    /* U slucaju da nema dovoljno memorije malloc vraca NULL */
    if (a == NULL)
    {
        printf("Greska : Nema dovoljno memorije!\n");
        return 1;
    }

    /* Nadalje a koristimo kao obican niz */
    for (i = 0; i<n; i++)
    {
        printf("a[%d]=", i);
        scanf("%d", &a[i]);
```

```
    }

    /* Nalazimo maksimum */
    max = a[0];
    for (i = 1; i<n; i++)
        if (a[i] > max)
            max = a[i];

    printf("Najveci element je %d\n", max);

    /* Duzni smo da rucno alociranu memoriju rucno i oslobodimo */
    free(a);
}
```

3. Demonstracija funkcije *calloc* - funkcija popunjava sadržaj niza nulama.

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

#define BR_ELEM 10

main()
{
    int *m, *c, i;

    /* Niz m NE MORA garantovano da ima sve nule */
    m = malloc(BR_ELEM*sizeof(int));

    /* Niz c MORA garantovano da ima sve nule */
    c = calloc(BR_ELEM, sizeof(int));

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

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

    free(m);
    free(c);
}
```

4 Dinamička alokacija memorije, funkcija *realloc*

1. Napisati program koji sa standardnog ulaza unosi cele brojeve dok se ne unese -1 kao oznaka za kraj unosa. Broj celih brojeva nije unapred poznat. Zadatak resiti korišćenjem dinamičke alokacije.

```
/* Program demonstrira niz kome se velicina tokom rada povecava
   - verzija sa funkcijom realloc */

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

/* Konstanta koja nam odredjuje za koliko elemenata cemo prosirivati
   niz prilikom jednog poziva funkcije realloc */
#define KORAK 10

main()
{
    int* a = NULL;      /* Niz je u pocetku prazan i postavljamo pokazivac
                           na NULL da bi mogao biti prosledjen funkciji
                           realloc */
    int* pom;           /* Pomocni pokazivac koji ce nam sluziti da
                           preuzme povratnu vrednost funkcije realloc
                           pre nego budemo mogli da je dodelimo pokazivacu
                           a */

    int duzina = 0, alocirano = 0;  /* Duzina predstavlja broj popunjenih
                                       elemenata niza, dok alocirano
                                       predstavlja broj alociranih
                                       elemenata */

    int n, i;

    do
    {
        printf("Unesi ceo broj (-1 za kraj): ");
        scanf("%d", &n);

        /* Ukoliko nema vise slobodnih mesta, vrsi se prosirivanje */
        if (duzina == alocirano)
        {
            /* Niz se prosiruje za 10 elemenata vise */
            alocirano = alocirano + KORAK;

            /* Poziv funkcije realloc za a = NULL, ekvivalentan je pozivu
               funkcije malloc(alocirano*sizeof(int)); */
```

```
            pom = realloc(a, alocirano*sizeof(int));

            /* Ukoliko funkcija realloc vrati pokazivac koji se
               razlikuje od NULL realokacija je uspela i dodeljujemo
               vrednost pomocnog pokazivaca pokazivacu a */
            if (pom != NULL)
                a = pom;
            else
                {
                    /* Ako realloc vrati NULL (a kao drugi parametar joj
                       nije prosledjena duzina 0) to znaci da realokacija
                       nije uspela i prostor na koji pokazuje pokazivac
                       a nece biti oslobodjen pa to moramo uraditi rucno */
                    free(a);
                    exit 1;
                }
        }
        /* Sada kada sigurno ima mesta u nizu upisujemo procitanu
           vrednost na tekucu poziciju u nizu i uvecavamo duzinu niza
           za 1 */
        a[duzina++] = n;

    } while (n != -1);

    /* Ispisujemo niz a */
    printf("Uneto je %d brojeva. Alocirano je mesta za %d brojeva\n",
        duzina, alocirano);

    printf("Brojevi su : ");
    for (i = 0; i<duzina; i++)
        printf("%d ", a[i]);

    /* Oslobadjamo niz a */
    free(a);
}
```