

Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

March 22, 2006

Sadržaj

1		5
1.1	Pokazivači i nizovi (polja)	5
1.2	Zadaci za vežbu	10

1

1

1.1 Pokazivači i nizovi (polja)

U C-u postoji čvrsta veza između pokazivača i nizova. Bilo koja operacija koja se može ostvariti dopisivanjem indeksa niza može se uraditi i sa pokazivačima.

Deklaracija

```
int a[10];
```

definiše niz **a** veličine 10 koji predstavlja blok od 10 uzastopnih objekata nazvanih **a[0]**, **a[1]**, ..., **a[9]**.

Notacija **a[i]** odgovara i-tom elementu niza.

Ako je **pa** pokazivač na ceo broj

```
int *pa;
```

tada iskaz **pa = &a[0]**;

podešava da **pa** pokaže na nulti element niza **a**, odnosno **pa** sadrži adresu od **a[0]**.

Ako **pa** pokazuje na određeni element polja, onda po definiciji **pa+1** pokazuje na sledeći element, **pa+i** pokazuje na i-ti element posle **pa**. Stoga, ako **pa** pokazuje na **a[0]** tada

```
*(pa+1)
```

se odnosi na sadržaj od **a[1]**.

pa+i je adresa od **a[i]**, a

```
*(pa+i)
```

je sadržaj od **a[i]**.

Ovo sve važi bez obzira na tip ili veličinu elemenata u polju **a**.

Iskaz **pa=&a[0]** se može napisati kao **pa=a** jer je ime niza sinonim za lokaciju početnog elementa.

Primer 1 *Veza između pokazivača i nizova.*

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

```
void print_array(int* pa, int n);
```

```
main()
```

```
{
```

```
int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

```
int num_of_elements = sizeof(a)/sizeof(int);
```

```
int* pa;
```

```
/* Niz je isto sto i adresa prvog elementa */
```

¹Preuzeto sa sajta <http://www.matf.bg.ac.yu/~milena>

```
printf("Niz a : %p\n", a);
printf("Adresa prvog elementa niza a (&a[0]) : %p\n", &a[0]);
/* Niz a : 0012FF5C
Adresa prvog elementa niza a (&a[0]) : 0012FF5C */

/* Moguce je dodeliti niz pokazivacu odgovarajuceg tipa */
pa = a;

printf("Pokazivac pa ukazuje na adresu : %p\n", pa);
/* Pokazivac pa ukazuje na adresu : 0012FF5C */

/* Nizu nije moguce dodeliti pokazivacku promenljivu
   (nizove mozemo smatrati KONSTANTNIM pokazivacima na prvi element) */
/* a = pa; */

/* Pokazivace je dalje moguce indeksirati kao nizove */
printf("pa[0] = %d\n", pa[0]);
printf("pa[5] = %d\n", pa[5]);
/* pa[0] = 1
   pa[5] = 6 */

/* Medjutim, sizeof(pa) je samo velicina pokazivaca, a ne niza */
printf("sizeof(a) = %d\n", sizeof(a));
printf("sizeof(pa) = %d\n", sizeof(pa));
/* sizeof(a) = 40
   sizeof(pa) = 4 */

/* Pozivamo funkciju za stampanje niza i saljemo joj niz */
print_array(a, num_of_elements);
/* 1 2 3 4 5 6 7 8 9 10 */

/* Pozivamo funkciju za stampanje niza
   i saljemo joj pokazivac na pocetak niza */
print_array(pa, num_of_elements);
/* 1 2 3 4 5 6 7 8 9 10 */
}

/* Prosledjivanje niza u funkciju
   void print_array(int pa[], int n);
   je ekvivalentno prosledjivanju
   pokazivaca u funkciju
   void print_array(int* pa, int n);
   Izmedju ovih konstrukcija nema nikakve razlike.
*/
void print_array(int* pa, int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", pa[i]);
    putchar('\n');
}
```

Prilikom deklaracije treba praviti razliku između niza znakova i pokazivača:

```
char poruka[]="danas je lep dan!";
char *pporuka = "danas je lep dan!";
```

poruka je niz znakova koji sadrži dati tekst. Pojedine znake moguće je promeniti ali se poruka uvek odnosi na isto mesto u memoriji.

pporuka je pokazivač, koji je inicijalizovan da pokazuje na konstantnu nisku, on može biti preusmeren da pokazuje na nešto drugo, ali rezultat neće biti definisan ako pokušate da modifikujete sadržaj niske (jer je to konstantna niska).

Ako deklariramo

```
char *pporuka1 = "danas je lep dan!";
char *pporuka2 = "danas je lep dan!";
char *pporuka3 = "danas pada kisa";
```

tada će pokazivači pporuka1 i pporuka2 pokazivati na isto mesto u memoriji, a pporuka3 na neko drugo mesto u memoriji.

Ako uporedimo (pporuka1==pporuka3) upoređić se vrednosti pokazivača. Ako uporedimo (pporuka1 < pporuka2) upoređić se vrednosti pokazivača. Ako dodelimo pporuka1=pporuka3 tada će pporuka1 dobiti vrednost pokazivača pporuka3 i pokazivaće na isto mesto u memoriji. Neće se izvršiti kopiranje sadržaja memorije!!!

Primer 2 Vežba pokazivačke aritmetike.

```
#include <stdio.h>

/* Funkcija pronalazi x u nizu niz
   date dimenzije,
   bez koriscenja indeksiranja.
   Funkcija vraca pokazivac na
   poziciju pronadjenog elementa. */

int* nadjiint(int* niz, int n, int x)
{
    while (n-- >= 0 && *niz!=x)
        niz++;

    return (n>=0)? niz: NULL;
}

main()
{
    int a[]={1,2,3,4,5,6,7,8};
    int* poz=nadjiint(a,sizeof(a)/sizeof(int),4);

    if (poz!=NULL)
        printf("Element pronadjen na poziciji %d\n",poz-a);
}
```

Primer 3

```
int strlen(char *s)
{
    int n;
```

```

    for(n=0; *s != '\0'; s++) n++;
    return n;
}

```

Primer 4

```

/* Funkcija kopira string t u string s */
void copy(char* s, char* t)
{
    while (*s++==*t++)
        ;
}

/* Ovo je bio skraceni zapis za sledeci kod
    while(*t != '\0')
    {
        *s=*t;
        s++;
        t++;
    }
    *s = '\0';

*/

```

Primer 5

```

/* Vrsi leksikografsko poredjenje dva stringa.
    Vraca :
        0 - ukoliko su stringovi jednaki
        <0 - ukoliko je s leksikografski ispred t
        >0 - ukoliko je s leksikografski iza t
*/ int string_compare1(char *s, char *t) {
    /* Petlja tece sve dok ne naidjemo
    na prvi razliciti karakter */
    for (; *s == *t; s++, t++)
        if (*s == '\0') /* Naisli smo na kraj
                        oba stringa, a
                        nismo nasli razliku */
            return 0;

    /* *s i *t su prvi karakteri u kojima
    se niske razlikuju.
    Na osnovu njihovog odnosa,
    odredjuje se odnos stringova */

    return *s - *t;
}

```

```

/* Mozemo koristiti i sintaksu kao kod nizova */
int string_compare2(char *s, char *t) {
    int i;
    for (i = 0; s[i] == t[i]; i++)

```

```
        if (s[i] == '\\0')
            return 0;
    return s[i] - t[i];
}
```

Primer 6 Pronalazi prvu poziciju karaktera *c* u stringu *s*, i vraća pokazivač na nju, odnosno *NULL* ukoliko *s* ne sadrži *c*.

```
char* string_char(char *s, char c)
{
    int i;
    for (; *s; s++)
        if (*s == c)
            return s;

    /* Nije nadjeno */
    return NULL;
}
```

Primer 7 Pronalazi poslednju poziciju karaktera *c* u stringu *s*, i vraća pokazivač na nju, odnosno *NULL* ukoliko *s* ne sadrži *c*.

```
char* string_last_char(char *s, char c)
{
    char *t = s;
    /* Pronalazimo kraj stringa s */
    while (*t++)
        ;

    /* Krecemo od kraja i trazimo c unazad */
    for (t--; t >= s; t--)
        if (*t == c)
            return t;

    /* Nije nadjeno */
    return NULL;
}
```

Primer 8

/* Verzija funkcije strstr implementirane
bez koriscenja indeksiranja */

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

/* proverava da li se niska t nalazi unutar niske s*/

```
int sadrzi_string(char s[], char t[])
{
    int i;
    for (i = 0; s[i]; i++)
    {
        int j, k;
        for (j=0, k=0; s[i+j]==t[k]; j++, k++)
            if (t[k+1]=='\\0')
                return i;
    }
}
```

```
    }
    return -1;
}

/* proverava da li se niska t nalazi unutar niske s*/
char* sadrzi_string_pok(char* s, char* t)
{
    while(*s)
    {
        char *i, *j;
        for (i = s, j = t; *i == *j; i++,j++)
            if (*(j+1)=='\0')
                return s;
        s++;
    }
    return NULL;
}

/* Cita liniju sa stadnardnog ulaza i
vraca njenu duzinu */
int getline(char* line, int max)
{
    char *s=line;
    int c;
    while ( max-->0 && (c=getchar())!='\n' && c!=EOF)
        *s++ = c;

    if (c=='\n')
        *s++ = c;

    *s = '\0';
    return s - line;
}

main()
{
    char rec[]="zdravo";
    char linija[100];
    while (getline(linija, 100))
        if (sadrzi_string_pok(linija, rec))
            printf("%s",linija);
}
```

1.2 Zadaci za vežbu

Zadatak 1 Koristeći pokazivače napisati funkciju koja nadovezuje string *t* na kraj stringa *s*. (Pretpostavlja se da u *s* ima dovoljno prostora.)

Zadatak 2 januar 2006. (I grupa) Napisati funkciju koja za celobrojni niz dimenzije *n*, prover-

ava da li među elementima niza postoje neka dva koja su jednaka.

Zadatak 3 januar 2006. (II grupa) Napisati funkciju koja za dve niske koje se prenose kao parametri utvrđuje da li su anagrami ili ne. Dve niske su anagrami ako se sastoje od istog broja istih karaktera. Na primer, niske "anagram" i "ramgana" jesu anagrami, dok "anagram" i "angrm" nisu.

Zadatak 4 januar 2006. (II grupa) Napisati funkciju koja u datom celobrojnomo nizu A dužine n pronalazi (ako postoji) takav par indeksa (i, j) da je zbir članova niza sa indeksima od i do j jednak zadatom broju m .

Zadatak 5 Napisati funkciju koja vraća prvu poziciju u niski $s1$ na kojoj se pojavljuje znak iz $s2$ ili -1 ako $s1$ ne sadrži ni jedan znak iz $s2$. Ako je $s1$ pera a $s2$ navip onda funkcija treba da vrati poziciju 0. Ako je $s1$ zeleno a $s2$ nana onda funkcija treba da vrati poziciju 4.

Zadatak 6 (a) Napisati funkciju koja ispituje da li je jedna reč prefiks druge reči.
(b) Napisati program koji za svaku liniju teksta koja se unosi sa standardnog ulaza a koja nije duža od 100 karaktera proverava da li je neka reč njen prefiks i štampa odgovarajuću poruku.

Zadatak 7 (a) Napisati funkciju koja ispituje da li je jedna reč sufiks druge reči.
(b) Napisati program koji za svaku liniju teksta koja se unosi sa standardnog ulaza a koja nije duža od 100 karaktera proverava da li je neka reč njen sufiks i štampa odgovarajuću poruku.

Zadatak 8 Učitava se linija po linija teksta. Odšampati svaku od tih linija tako da ima veliko slovo na početku rečenice i sva mala unutar rečenice(., ?, !).