Programiranje 1
Programski jezik C
7. čas

Mirko Spasić

Nizovi

- Niz možemo inicijalizovati zadavanjem vrednosti elemenata niza u vitičastim zagradama. Dimenzija niza se određuje na osnovu broja inicijalizatora.
  int a[] = {1, 2, 3, 4, 5, 6};
- Ovo važi i za niske karaktera
  char s[] = {‘a’, ‘b’, ‘c’};
- Prethodni primer je ekvivalentan sledećem:
  char s[] = {97, 98, 99};

Operator SIZEOF

- Operator sizeof() je unarni operator koji se može primenjivati na podatke i tipove.
- Ako se primeni na tip T, tada je vrednost izraza sizeof(T) broj bajtova koji zauzima jedan podatak tog tipa.
- Ako se primeni na podatak (ili proizvoljan izraz), tada sizeof(izraz) ima vrednost broja bajtova koje zauzima podatak, imajući u vidu tip podatka (izraza).
- Operator sizeof daje vrednost tipa int.
- Vrednost izraza sizeof se računa u vreme prevođenja programa, jer je već tada jasno koja je veličina objekta na koji se primenjuje.

Dimenzija niza

- Dimenziju niza koga smo inicijalizovali vitičastim zagradama, i nismo eksplicitno precizirali dimenziju niza, možemo odrediti na sledeći način:
  int a_br_elem = sizeof(a)/sizeof(int);
  int s_br_elem = sizeof(s)/sizeof(char);
  gde je a niz celobrojnih vrednosti, a s niz karaktera.

Nizovi - primer

- Napisati funkciju koja računa euklidsku mormu normu vektora. (fja treba da prima kao argument niz čiji su elementi tipa double, i dimenziju niza). Euklidska norma vektora, tj. Niza, je koren sume kvadrata svih elemenata niza. Napisati potom i program koji testira ovu funkciju.

Veza između nizova i pokazivača

- U C-u postoji jaka veza između pokazivača i nizova.
- Deklaracija

  int a[10];

  definiše niz a veličine 10, odnosno blok od 10 susednih objekata u memoriji sa imenima a[0], a[1], a[2], a[3], a[4], a[5], a[6], a[7], a[8], a[9].
- Zapis a[i] ukazuje na i-ti element niza a.

Veza između nizova i pokazivača

- Ako je pa pokazivač na neki ceo broj deklarisan pomoću

```
int *pa;
```

tada dodeljivanje

```
pa = &a[0];
```

podešava pa da pokazuje na nulti element niza a, to jest, pa sadrži adresu od a[0].

Veza između nizova i pokazivača

- Kako je ime niza sinonim za adresu početnog elementa niza, dodela

  ```
  pa = &a[0];
  ```

  se može napisati i kao

  ```
  pa = a;
  ```

- Ako pa pokazuje na određeni element nekog niza, tada pa+1 pokazuje na naredni element pa tako pa+i pokazuje i elemenata iza pa, a pa-i na i elemenata ispred pa.

Veza između nizova i pokazivača

- Dakle, ako pa pokazuje na a[0] onda je

  *(pa+i)

  zapravo sadržaj i-tog elementa niza to jest isto je što i a[i].

  *(pa+i) ⇔ a[i]⇔ pa[i]

- pa+i je isto što i &pa[i].

  pa+i ⇔ &pa[i] ⇔ a+i

- Ime niza se ponaša kao pokazivač na početni element niza.

Veza između nizova i pokazivača

- Razlika između pokazivača pa i a je u tome što a nije pokazivačka promenljiva, pa samim tim ne možemo da joj promenimo vrednost.
- Drugim rečima, izraz pa++ je dozvoljen (pomera pokazivač pa za jednu poziciju u nizu udesno)
- a++ nije dozvoljen, jer a nije promenljiva pokazivačkog tipa, već tipa "niz celih brojeva" koja je samo automatski konvertovana u pokazivač na prvi član niza.

Veza između nizova i pokazivača - primer

```
#include <stdio.h>
int main() {
	int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
	*p = a;
	int i;
	printf("Vrednost izraza a (tipa int *): %p\n", a);
	printf("Vrednost izraza p (tipa int *): %p\n", p);
	printf("Vrednost izraza *a (tipa int): %d\n", *a);
	printf("Vrednost izraza *p (tipa int): %d\n", *p);
	for(i = 0; i < 10; i++) {
		printf("Vrednost izraza &a[%d] (tipa int *): %p\n", i, &a[i]);
		printf("Vrednost izraza &p[%d] (tipa int *): %p\n", i, &p[i]);
		printf("Vrednost izraza (a + %d) (tipa int *): %p\n", i, a + i);
		printf("Vrednost izraza (p + %d) (tipa int *): %p\n", i, p + i);
		printf("Vrednost izraza a[%d] (tipa int): %d\n", i, a[i]);
		printf("Vrednost izraza p[%d] (tipa int): %d\n", i, p[i]);
		printf("Vrednost izraza *(a + %d) (tipa int): %d\n", i, *(a + i));
		printf("Vrednost izraza *(p + %d) (tipa int): %d\n", i, *(p + i));
	}
	return 0;
}
```

Veza između nizova i pokazivača - primer

- Ispis prethodnog primera:

Vrednost izraza a (tipa int *): 0xbfd1f11c
Vrednost izraza p (tipa int *): 0xbfd1f11c
Vrednost izraza *a (tipa int): 0
Vrednost izraza *p (tipa int): 0
Vrednost izraza &a[0] (tipa int *): 0xbfd1f11c
Vrednost izraza &p[0] (tipa int *): 0xbfd1f11c
Vrednost izraza (a + 0) (tipa int *): 0xbfd1f11c
Vrednost izraza (p + 0) (tipa int *): 0xbfd1f11c
Vrednost izraza a[0] (tipa int): 0
Vrednost izraza p[0] (tipa int): 0
Vrednost izraza *(a + 0) (tipa int): 0
Vrednost izraza *(p + 0) (tipa int): 0

Veza između nizova i pokazivača - primer

Vrednost izraza &a[1] (tipa int *): 0xbfd1f120
Vrednost izraza &p[1] (tipa int *): 0xbfd1f120
Vrednost izraza (a + 1) (tipa int *): 0xbfd1f120
Vrednost izraza (p + 1) (tipa int *): 0xbfd1f120
Vrednost izraza a[1] (tipa int): 1
Vrednost izraza p[1] (tipa int): 1
Vrednost izraza *(a + 1) (tipa int): 1
Vrednost izraza *(p + 1) (tipa int): 1
Vrednost izraza &a[2] (tipa int *): 0xbfd1f124
Vrednost izraza &p[2] (tipa int *): 0xbfd1f124
Vrednost izraza (a + 2) (tipa int *): 0xbfd1f124
Vrednost izraza (p + 2) (tipa int *): 0xbfd1f124
Vrednost izraza a[2] (tipa int): 2
Vrednost izraza p[2] (tipa int): 2
Vrednost izraza *(a + 2) (tipa int): 2
Vrednost izraza *(p + 2) (tipa int): 2
...

Primer – Obrtanje nizova (indeksna sintaksa)

```
#include <stdio.h>
#define MAX 100

int main() {
	int n;
	int a[MAX];
	int i,j;
	/* Unosimo broj elemenata */
	printf("Uneti broj elemenata niza (<= 100): ");
	scanf("%d", &n);

	/* Proveravamo da li je prekoraceno ogranicenje */
	if(n > MAX)
		n = MAX;

	/* Unosimo elemente niza */
	printf("Uneti elemente niza:\n");
	for(i = 0 ; i < n ; i++)
		scanf("%d", &a[i]);

	/* Prikaz niza */
	printf("Uneli ste niz:\n");
	for(i = 0 ; i < n ; i++)
		printf("%d ", a[i]);
	printf("\n");

	/* Obrcemo niz */
	for(i = 0, j = n - 1 ; i < j ; i++, j--) {
		int t = a[i];
		a[i] = a[j];
		a[j] = t;
	}

	/* Prikaz niza */
	printf("Niz nakon obrtanja:\n");
	for(i = 0 ; i < n ; i++)
		printf("%d ", a[i]);
	printf("\n");
	return 0;
}
```

Primer – Obrtanje nizova (pokazivačka sintaksa)

```
#include <stdio.h>
#define MAX 100

int main() {
	int n;
	int a[MAX];
	int *p,*q;
	/* Unosimo broj elemenata */
	printf("Uneti broj elemenata niza (<= 100): ");
	scanf("%d", &n);

	/* Proveravamo da li je prekoraceno ogranicenje */
	if(n > MAX)
		n = MAX;

	/* Unosimo elemente niza */
	printf("Uneti elemente niza:\n");
	for(p = a ; p - a < n ; p++)
		scanf("%d", p);

	/* Prikaz niza */
	printf("Uneli ste niz:\n");
	for(p = a ; p - a < n ; p++)
		printf("%d ", *p);
	printf("\n");

	/* Obrcemo niz */
	for (p = a, q = a + n - 1 ; p < q ; p++, q--) {
		int t = *p;
		*p = *q;
		*q = t;
	}

	/* Prikaz niza */
	printf("Niz nakon obrtanja:\n");
	for(p = a ; p - a < n ; p++)
		printf("%d ", *p);
	printf("\n");
	return 0;
}
```

Nizovi - primer

- Program za uneti datum utvrđuje redni broj tog dana u datoj godini. Program demonstrira upotrebu nizova, kao i veze nizova i pokazivača.

```
#include <stdio.h>
int main() {
    int obicna[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
    int prestupna[] = {31,29,31,30,31,30,31,31,30,31,30,31};
    int dan, mesec, godina;
    int i, *tekuca;
    int dan_u_godini = 0;

    printf("Uneti datum u formatu dd:mm:yyyy: ");
    scanf("%d:%d:%d", &dan, &mesec, &godina);
```

```
    /* Proveravamo da li je godina prestupna */
    if(godina % 400 == 0 || (godina % 100 != 0 && godina % 4 == 0))
        tekuca = prestupna;
    else
        tekuca = obicna;

    /* Sumiramo dane protekle prethodnih meseci. Indeks ide od 0 do
    mesec - 2, zato sto indeksi u C-u pocinju od 0. Npr, za dan u mesecu
    martu (treci mesec) treba sabrati broj dana u januaru i februaru (tj.
    indeks u petlji treba ici od 0 do 1, ukljucujuci i jednicu). */
    for(i = 0 ; i < mesec - 1 ; i++)
        dan_u_godini += tekuca[i];

    /* Nakon toga jos treba dodati broj proteklih dana u tekucem mesecu */
    dan_u_godini += dan;

    printf("Uneti datum je %d. dan u godini\n", dan_u_godini);
    return 0;
}
```

Prenos niza u funkciju

- Nizovi se prenose u funkciju tako što se prenese adresa njihovog početka. Iz tog razloga oni se MOGU MENJATI u okviru funkcije.
  povratni_tip ime_funkcije (tip niz[], int dimenzija);
  što je ekvivalentno sledećem prenosu (pokazivačka sintaksa)
  povratni_tip ime_funkcije (tip *niz, int dimenzija);
- Sve nizove osim niski karaktera (stringova) neophodno je prenositi zajedno sa dimenzijom niza.

Primeri za vežbu

- Napisati funkciju koja ispisuje niz brojeva, a zatim i program koji je testira.
- Napisati funkciju koja računa skalarni proizvod dva niza brojeva, a zatim i program koji je testira.
- Napisati program koji unosi realne brojeve sa ulaza, i računa zbir i proizvod elemenata, kao i najveći i najmanji element u nizu. Brojeve čuvati u nizu za koji se pretpostavlja da je dimenzije manje od 100