Tema 3

Nizovi, pokazivači

Deklaracija nizova

- Niz predstavlja kolekciju elemenata istog tipa
- Primer deklaracije celobrojnog niza od pet elemenata
  - `int niz[5];`

Pristupanje elementima niza

- `niz[0] = 4;`
- `niz[1] = 2 * niz[0];`
- `niz[2] = niz[0] * niz[1];`
- `niz[3] = 5;`
- `niz[4] = 7;`
- `a = niz[10];`
- Indeksiranje nizova u programskom jeziku C kreće od 0!

Inicijalizacija niza

- Primer inicijalizacije vrednosti niza nulama

```
int main()
{
    int i, niz[5];

    for(i = 0; i < 5;i++)
    {
        niz[i] = 0;
    }

    return 0;
}
```

Inicijalizacija niza prilikom deklaracije

- `int niz[5] = {1, 3, 2, 4, 5};`
- Ukoliko se ne navedu sve vrednosti, ostatak se inicijalizuje nulama
- Primer inicijalizacije niza nulama
  - `int niz[5] = {0};`
- Dimenzija niza se može izostaviti ukoliko se niz inicijalizuje
  - `int meseci[] = {31, 28, 31, 30, 31, 31, 30, 31, 30, 31};`
  - Koliko elemenata ima prethodni niz?

Unos elemenata niza

Primer unosa elemenata niza sa tastature:

```c
int main()
{
    int i, niz[5];

    for(i = 0;i < 5;i++)
    {
        printf("niz[%d] = ", i);
        scanf("%d", &niz[i]);
    }

    return 0;
}
```

- Obratiti pažnju da se indeksi niza kreću od 0 do 4!

Prikaz elemenata niza

Primer prikaza elemenata niza unetih sa tastature:

```
int main()
{
    int i, niz[5] = { 1, 2, 3, 4, 5 };

    printf("[");
    for(i = 0;i < 5;i++)
    {
        if(i > 0)
        {
            printf(", ");
        }
        printf("%d", niz[i]);
    }
    printf("]");

    return 0;
}
```

- Prikaz na ekranu: `[1, 2, 3, 4, 5]`

define pretprocesorska direktiva

- Koristi se kako bi se izbeglo pisanje konstantnih vrednosti
  - Smanjuje šansu za pravljenje greške prilikom izmena koda
  - Primena prilikom definisanja dimenzije niza (`MAX_SIZE`)
- Veličina niza je fiksna, šta ako želimo da iskoristimo samo deo?
  - Uvođenje posebne promenljive `n` koja to određuje
    - Unosi se sa tastature
  - `n` mora biti u granicama `0 < n <= MAX_SIZE`
    - Kako ograničiti korisnika da unese `n` u datim granicama?

Primer 1

Dat je niz od maksimalno 30 celobrojnih elemenata. Učitati n elemenata i ispisati ih po učitanom i obrnutom redosledu.

```c
#include <stdio.h>

#define MAX_SIZE 30

int main()
{
    int a[MAX_SIZE], i, n;

    do
    {
        printf("Unesite broj elemenata niza: ");
        scanf("%d", &n);
    } while (n <= 0 || n > MAX_SIZE);

    // unos elemenata niza
    for(i = 0;i < n;i++)
    {
        printf("niz[%d] = ", i);
```

```
        scanf("%d", &a[i]);
    }

    // ispis u unetom redosledu
    printf("[");
    for(i = 0;i < n;i++)
    {
        if (i > 0)
        {
            printf(", ");
        }
        printf("%d", a[i]);
    }
    printf("]\n");

    // ispis u redosledu obrnutom od unetog
    printf("[");
    for(i = n - 1;i >= 0;i--)
    {
        if (i < n - 1)
        {
            printf(", ");
        }
        printf("%d", a[i]);
```

```
    }
    printf("]\n");

    return 0;
}
```

Zadatak 1

Dat je niz `A` od maksimalno 30 celobrojnih elemenata. Učitati `n` elemenata, zatim učitati ceo broj `br`. Na standardnom izlazu ispisati broj pojavljivanja `br` u nizu `A`.

- Primer:
  - `A = [2, 5, 6, 2, 8, 9, 2]`
  - `br = 2`

Očekivani ispis:

```
Broj 2 se pojavljuje 3 puta u nizu A = [2, 5, 6, 2, 8, 9, 2].
```

Zadatak 2

Dat je niz od maksimalno 20 realnih elemenata. Učitati n elemenata, a zatim naći maksimalnu vrednost.

Primer 2

Primer Selection Sort algoritma

```
int j, min_idx, tmp;    // niz a, promenljive i i n su definisane

for(i = 0;i < n - 1;i++)
{
    min_idx = i;
    for(j = i + 1;j < n;j++)
    {
        if(a[min_idx] > a[j])
        {
            min_idx = j;
        }
    }
    if(min_idx != i)
    {
        tmp = a[i];
        a[i] = a[min_idx];
        a[min_idx] = tmp;
    }
}
```

- Sortiranje niza -ređanje elemenata niza u rastućem ili opadajućem redosledu
- Selection Sort traži element u podskupu elemenata desno od i-tog elementa, koji je najveći(najmanji) i menja vrednost njegovog i i-tog elementa
- Zamena sadržaja dva elementa niza radi se uz pomoć dodatne promenljive tmp (primer sa čašama čije tečnosti treba zameniti uz pomoć treće čaše)
- Selection Sort je samo jedan od algoritama, pogledati Bubble Sort, Selection Sort, Merge Sort, Quick Sort itd.
    - Krajnji rezultat je isti, samo način na koji se to postiže je drugačiji (bolji/gori u određenim situacijama)

Zadatak 3

Proširiti *Primer 1* sa sortiranjem nizova pre ispisa. Koristiti Selection Sort algoritam ili algoritam za sortiranje nizova po izboru (potražiti objašnjenje/implementaciju na internetu).

Zadatak 4

Napisati program koji pronalazi prvi element niza koji je najbliži srednjoj vrednosti niza celih brojeva. Niz može da ima najviše 20 elemenata.

Zadatak 5

Dat je niz X od maksimalno 25 celobrojnih elemenata. Učitati n elemenata u niz X i formirati nizove A i B, pri čemu su elementi niza A parni, a elementi niza B negativni elementi niza X. Ispisati nizove X, A i B.

Pokazivačka promenljiva

- Sadrži adresu promenljive
- Celobrojna vrednost, označava lokaciju u memoriji
- Može sadržati adresu neke druge promenljive ili adresu početka memorijskog bloka
- Tip pokazivačke promenljive, isto `int`, `double` itd.
- Prefiks "*" u imenu označava da se radi o pokazivačkoj promenljivoj

Rad sa pokazivačkim promenljivama

- Deklaracija
  - `int a = 5;`
  - `int b, c;`
  - `int *p1, *p2;`
  - Koje promenljive su pokazivačke?
- Dodela vrednosti
  - `p1 = &a;`
  - `p2 = &b;`
- Pristup lokaciji
  - `c = *p1;` // c <- a
  - `*p2 = 6;` // b <- 6

Referenciranje

- Unarni operator `&` daje adresu promenljive
- Izraz `p1 = &a` dodeljuje vrednost adrese promenljive `a` pokazivačkoj promenljivoj `p1`
  - `p1` "pokazuje" na promenljivu `a`
- Da bi se ištampala vrednost pokazivačke promenljive, koristi se format specifikator `%p`
  - Vrednost počinje sa "0x" i predstavlja celobrojnu vrednost memorijske lokacije u heksadecimalnom brojnom sistemu

Dereferenciranje

- Unarni operator `*`, omogućuje posredan pristup podatku pomoću adrese u pokazivačkoj promenljivoj
- Razlika između `*` kod deklaracije i operatora dereferenciranja!
  - Deklaracija pokazivačke promenljive `p1`
    - `int *p1;`
  - Dodela adrese promenljive `a` pokazivačkoj promenljivoj `p1`
    - `p1 = &a;`
  - Dodela vrednosti `3` promenljivoj čija adresa se nalazi u `p1`
    - `*p1 = 3;`

Specijalna konstanta NULL

- Nalazi se u `stdio.h`
- Predstavlja vrednost koja opisuje da pokazivačka promenljiva pokazuje "ni na šta"

Primer:

```
int *p;
p = NULL;
```

- Neinicijalizovana pokazivačka promenljiva nema vrednost `NULL`
  - Princip zauzeća memorijskog prostora je isti kao i kod promenljivih generalno
- Ukoliko je to neophodno, potrebno je inicijalizovati pokazivačku promenljivu na `NULL`

Primer 1

```c
#include <stdio.h>

int main()
{
    int i;
    int *pi;

    i = 7;
    pi = &i;

    printf("Vrednost promenljive i: %d\n", i);
    printf("Vrednost adrese u pi: %p\n\n", pi);

    printf("Vrednost adrese pi: %p\n", &pi);
    printf("Vrednost adrese u pi: %p\n\n", pi);

    printf("Vrednost adrese u pi: %p\n", pi);
    printf("Vrednost promenljive na adresi iz pi: %d\n\n", *pi);

    i = 10;
```

```
    printf("Vrednost adrese u pi: %p\n", pi);
    printf("Vrednost promenljive na adresi iz pi: %d\n\n", *pi);

    (*pi)++;

    printf("Vrednost adrese promenljive i: %p\n", &i);
    printf("Vrednost promenljive i: %d\n\n", i);

    return 0;
}
```

Pokazivači i nizovi

- Nizovi se mogu posmatrati kao pokazivači
- Prilikom definicije niza, zauzme se navedeni broj memorijskih lokacija
  - U nazivu promenljive niza (identifikatoru) nalazi se adresa početka njegovog memorijskog bloka

Primer 2

```c
#include <stdio.h>

#define SIZE 10

int main()
{
    int a[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };

    int *pa;
    int i;

    pa = a;

    printf("%d\n", *pa);
    printf("%d\n", *(a + 1));
```

```
    printf("[");
    for(i = 0;i < SIZE;i += 2)
    {
        if(i > 0)
        {
            printf(", ");
        }

        pa = a + i;
        printf("%d", *pa);
    }
    printf("]\n");

    return 0;
}
```

Vrlo česte greške

- Nemoguće je definisati pokazivač na konstantu ili izraz
- Nemoguće je promeniti adresu promenljive
  - Adresni opseg programa i promenljivih određuje isključivo operativni sistem
- Iz ovih razloga, sledeći izrazi su pogrešni:
  - `i = &3;`
  - `j = &(k + 5);`
  - `k = &(a == b);`
  - `&a = &b`
  - `&a = 150;`