Tema 3

Nizovi, pokazivači, char tip i stringovi

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:

```
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);

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

```
    printf("[");
    for(i = 0;i < n;i++) {
        if(i > 0) {
            printf(", ");
        }
        printf("%d", a[i]);
    }
    printf("]\n");

    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

- Sortiranje niza -ređanje elemenata niza u rastućem ili opadajućem redosledu.

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;
            }
        }

        tmp = a[i];
        a[i] = a[min_idx];
        a[min_idx] = tmp;
    }
```

- Pogledati Bubble Sort, Selection Sort, Merge Sort, Quick Sort itd.

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;`

char tip

- Promenljiva tipa `char` koristi se za čuvanje tekstualnih znakova
- Znakovi mogu biti:
  - slova
  - cifre
  - specijalni znaci
  - neštampajući (beli) znaci
- Može se koristiti i za čuvanje malih, celobrojnih vrednosti
  - U slučaju `signed` tipa od -128 do 127
  - U slučaju `unsigned` tipa od 0 do 255
- Primeri `char` konstanti: `'a'`, `'A'`, `'1'`, `'?'`, `'\n'`

char je samo broj

- Svakom karakteru pridružuje se numerički kod
- Postoje različiti skupovi kodova (standardi):
    - ASCII (American Standard Code for Information Interchange)
        - Najčešće korišćen
    - EBCDIC
        - zastareo, danas se retko koristi
    - Unicode
        - predstavnik novijih skupova karaktera
- Koristićemo ASCII

Primer 1

Korišćenje char kao znakovnog tipa i za malu numeričku vrednost

```c
#include <stdio.h>

int main() {
    char znak;

    printf("Unesite znak: ");
    scanf("%c", &znak);

    printf("Znak kao karakter je: %c\n", znak);
    printf("Numericka vrednost znak-a je: %d\n", znak);
    printf("Karakter posle %c je %c\n", znak, znak + 1);

    return 0;
}
```

Stringovi

- Nizovi karaktera sa specijalnom oznakom za kraj stringa
  - `'\0'` terminator, označava kraj stringa, "žrtvovana" jedna lokacija kako ne bi bilo potrebe da se posebno prati zauzeće niza
- Unos i ispis sa standardnog ulaza
  - `scanf` i `printf` sa format specifikatorom `%s`
  - Specijalizovane funkcije za unos i ispis stringova `gets` i `puts`
- Biblioteka sa funkcijama za rad sa stringovima `string.h`
  - `strlen` - vraća dužinu stringa
  - `strcat` - spajanje stringova
  - `strcpy` - kopiranje stringa
  - `strstr` - pokazivač na prvu pojavu stringa unutar većeg stringa
  - `strcmp` - poređenje stringova
- Za više informacija o funkciji, koristiti `man` stranice, na primer: `man strlen`

Primer 1

Funkcije za unos i ispis stringova

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

#define MAX_STRING 101

int main() {
    char str1[MAX_STRING], str2[MAX_STRING];

    printf("Unesite prvi string: ");
    scanf("%s", str1);
    __fpurge(stdin);

    printf("Unesite drugi string: ");
    fgets(str2, MAX_STRING, stdin);
    int duzina_str2 = strlen(duzina_str2);
    if(duzina_str2 < MAX_STRING - 1) str2[duzina_str2 - 1] = '\0';
```

```
    printf("\n%s\n", str1);
    printf("%s\n\n", str2);

    puts(str1);
    puts(str2);

    return 0;
}
```

Primer ispisa na standardnom izlazu:

```
Unesite prvi string: Primer prvog stringa
Unesite drugi string: Primer drugog stringa

Primer
Primer drugog stringa

Primer
Primer drugog stringa
```

- `scanf` je funkcija osetljiva na whitespace karaktere

Primer 2

Korišćenje funkcije `strcat` za spajanje stringova

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

#define MAX_STRING 31

int main() {
    char str1[MAX_STRING] = "Veni, ";
    char str2[] = "vidi, ";

    printf("%s\n", strcat(str1, str2));
    strcat(str1, "vici.");
    printf("%s\n", str1);

    return 0;
}
```

Primer 3

Korišćenje funkcije `strcpy` za kopiranje stringova

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

#define MAX_STRING 31

int main() {
    char str1[MAX_STRING];

    strcpy(str1, "Znanje");
    puts(str1);
    puts(strcpy(str1, "Imanje"));

    return 0;
}
```

Napomene

- Nikada ne koristiti operator dodele vrednosti (=) nad stringovima!
    - Ukoliko se pokuša dodela stringa drugom stringu, to će biti sintaksna greška
    - Dodela pokazivačkoj promenljivoj tipa `char *` je moguća, ali u pitanju je referenca na isti string
        - Svaka modifikacija preko pokazivačke promenljive promeniće originalni string
- Korišćenjem funkcije `strcpy`, vrši se kopiranje sadržaja jednog stringa u drugi
    - Karakteri će biti kopirani jedan po jedan
    - Izmena sadržaja u drugom stringu neće uticati na prvi string

Primer 4

Korišćenje funkcije `strstr` za prebrojavanje pojave stringa unutar većeg stringa

```
int broj_pojavljivanja(char *str_veliki, char *str_mali) {
    int br = 0;

    while(strstr(str_veliki, str_mali) != NULL) {
        br++;
        str_veliki = strstr(str_veliki, str_mali)
                     + strlen(str_mali);
    }

    return br;
}
```

- Funkcija `strstr` vraća povratnu vrednost `NULL` kada mali string ne postoji unutar većeg stringa

Primer 5

Korišćenje funkcije `strcmp`

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

#define MAX_STRING 101

int main() {
    char str1[MAX_STRING] = "";

    while(strcmp(str1, "exit") != 0) {
        fgets(str1, MAX_STRING, stdin);
        int duzina_str1 = strlen(str1);
        if(duzina_str1 < MAX_STRING - 1) str1[strlen(str1) - 1] = '\0';
        printf("Unet je string: %s\n", str1);
    }

    return 0;
}
```

Napomene

- Nikada ne koristiti relacioni operator provere jednakosti (==) za poređenje stringova!
    - Uporediće se adrese nizova, relacioni izraz će biti tačan samo ukoliko se poredi string sa samim sobom
- Stringovi su nizovi karaktera, njihov sadržaj je moguće porediti samo karakter po karakter
    - Što funkcija `strcmp` u osnovi i radi, a vodi računa i o dužini poređenih stringova

Zadatak 1

Napisati program koji ispisuje string obrnutim redosledom od unetog.

- Koristiti funkciju `strlen`

Zadatak 2

Napisati program koji proverava da li je uneti string palindrom. Ukoliko je string palindrom, u pitanju su reči ili rečenice koje se mogu isto čitati s leva na desno i obrnuto.

- Napraviti program tako da prvo radi za reči koje su palindromi (na primer, "rotor")
- Proširiti funkcionalnost tako da je moguće isto uraditi i za skup reči (na primer, "Ana voli Milovana")
    - Ignorisati razmake i odgovarajuća mala i velika slova smatrati istim prilikom provere
- Proširiti funkcionalnost tako da je moguće isto uraditi i za rečenice (na primer, "Ana voli Milovana.")
    - Ignorisati znake interpunkcije

Zadatak 3

Napisati program koji kao parametre uzima jedan string i karakter. Program ispisuje broj pojavljivanja karaktera u stringu.

- Za primer stringa `"tatatatira"` i karaktera `'a'`, broj pojavljivanja karaktera je 4

Zadatak 4

Napisati program koja modifikuje string tako što njegova mala slova transformiše u velika i obrnuto.

- Znakove koji nisu slova program ne treba da obrađuje, već ih ostavlja takve kakvi su