Tema 7

Datoteke i dinamička alokacija memorije

Datoteke

- Sekvenca bajtova sačuvana na disku
  - Životni vek datoteke nije direktno uslovljen trajanjem programa
- Binarne i tekstualne
- Rad isključivo preko pokazivača na tip `FILE`
- Funkcije za rad sa datotekama nalaze se u `stdio.h` biblioteci
- Osnovne funkcije za rad sa datotekama
  - `fopen` - otvaranje datoteke
    - Režimi pristupa
  - `fclose` - zatvaranje datoteke
    - Pokazivač tipa `FILE` se ne postavlja na `NULL` vrednost!

Tekstualne datoteke

- Ektenzija `.txt`
- Sadržaj datoteke je isključivo tekst
- Funkcije za rad sa tekstualnim datotekama
  - Univerzalne funkcije za rad sa svim tipovima podataka
    - `fscanf` - formatirano čitanje sadržaja datoteke
    - `fprintf` - formatirani ispis sadržaja u datoteku
  - Specijalizovane funkcije za rad sa nizovima znakova
    - `fgets` - učitavanje određenog broja znakova iz datoteke u niz karaktera
    - `fputs` - ispis određenog broja znakova iz niza znakova u datoteku

Primer 1

Učitavanje iz i zapis u tekstualnu datoteku.

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

#define MAX_SIZE 30

typedef struct tacka_st
{
    double x;
    double y;
} TACKA;


FILE *safe_fopen(char *, char *, int);
void ucitaj_tacke(FILE *, TACKA *, int *);
void ispisi_koordinate(FILE *, TACKA *, int);
```

```c
int main(int argc, char **argv)
{
    TACKA tacke[MAX_SIZE];
    int n;
    FILE *ulazna, *izlazna;

    if(argc != 3)
    {
        puts("Primer poziva programa: ./a.out in.txt out.txt");
        exit(EXIT_FAILURE);
    }

    ulazna = safe_fopen(argv[1], "r", 2);
    ucitaj_tacke(ulazna, tacke, &n);
    fclose(ulazna);

    izlazna = safe_fopen(argv[2], "w", 3);
    ispisi_koordinate(izlazna, tacke, n);
    fclose(izlazna);

    return 0;
}
```

```c
FILE *safe_fopen(char *ime, char *rezim, int kod_greske)
{
    FILE *pf = fopen(ime, rezim);

    if(pf == NULL)
    {
        printf("Datoteku %s nije moguce otvoriti.\n", ime);
        exit(kod_greske);
    }

    return pf;
}

void ucitaj_tacke(FILE *ulazna, TACKA *t, int *pn)
{
    int i = 0;

    while(fscanf(ulazna, "%lf %lf",
                 &t[i].x, &t[i].y) != EOF)
    {
        i++;
    }

    *pn = i;
```

```
}

void ispisi_koordinate(FILE *izlazna, TACKA *t, int n)
{
    int i;

    for(i = 0;i < n;i++)
    {
        fprintf(izlazna, "(%.2lf, %.2lf)\n", t[i].x, t[i].y);
    }
}
```

Primer 1 - dodatna pojašnjenja

- `typedef` za preimenovanje tipa (zarad kraćeg zapisa)
- Funkcija `exit` deo je biblioteke `stdlib.h`
  - Za razliku od `return` okončava program iz bilo koje funkcije
  - Konstante definisane u `stdlib.h`
    - `EXIT_SUCCESS` - uspešno izvršen program
    - `EXIT_FAILURE` - neuspešno izvršen program
- Pomoćna funkcija `safe_fopen`
  - Provera da li je uspešno otvorena datoteka je neophodna
  - U slučaju neuspešno otvorene datoteke, `fopen` vraća `NULL`
    - Izlazi iz programa pomoću funkcije `exit` sa prosleđenim kodom greške (celobrojna vrednost, često navedena u tekstu zadatka)
    - Primer: ne postoji fajl koji se otvara u režimu za čitanje `"r"`

Primer 1 - poziv programa i rezultat

Za poziv programa:

```
./a.out tacke.txt koordinate.txt
```

Gde je sadržaj ulazne datoteke `tacke.txt`:

```
1 2
2 3
3 4
4 5
```

Na osnovu kog je dobijen sadržaj izlazne datoteke `koordinate.txt`:

```
(1.00, 2.00)
(2.00, 3.00)
(3.00, 4.00)
(4.00, 5.00)
```

Zadatak 1

Napisati program koji iz tekstualne datoteke učitava `n` elemenata strukture tipa `auto_st` koja se sastoji iz sledećih polja:

- Marka automobila (jedna reč, do 20 karaktera)
- Kubikaža (prirodan broj)
- Godište (prirodan broj)

Ime ulazne tekstualne datoteke i kubikaža zadaju se kao argumenti komandne linije. Na osnovu zadatih informacija pronaći najnoviji auto sa kubikažom ne većom od zadate i rezultat upisati u tekstualnu datoteku, koja ima naziv oblika `<marka_automobila>.txt`.

Dinamička alokacija memorije

- Koristi se u slučaju da potrebna količina memorije koju program koristi nije unapred poznata
- Upravljanje preko pokazivača
- Potrebno je zatražiti dodatnu memoriju od operativnog sistema i nakon korišćenja je osloboditi
    - Ako se memorija ne oslobodi, nastaje problem *Memory Leak*

Korišćene funkcije

Dinamičko zauzimanje memorije, bez inicijalizacije:

```
void *malloc(size_t size);
```

Dinamičko zauzimanje memorije, inicijalizacija na nulu:

```
void *calloc(int n, size_t size);
```

Prebacivanje dinamički zauzete memorije u veću/manju dinamički zauzetu memoriju:

```
void *realloc(void *ptr, size_t size);
```

Oslobađanje dinamički zauzete memorije:

```
void free(void *ptr);
```

Funkcije se nalaze u biblioteci `stdlib.h`

Primer 1

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

int main()
{
    int n;  // Dimenzija niza
    int *a;
    int i, max;

    do
    {
        printf("Unesi dimenziju niza: ");
        scanf("%d", &n);
    } while(n <= 0);

    // Sada kada znamo koliko je memorije potrebno
    // pozivamo funkciju malloc za dinamicku alokaciju
    // memorije
    a = (int *) malloc(n * sizeof(int));
```

```
    if(a == NULL)
    {
        printf("Greska: Nema dovoljno memorije!\n");
        return 1;
    }

    // Nadalje a koristimo sa indeksnom notacijom
    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);

    // Obavezno oslobadjamo dinamicki alociranu memoriju
    free(a);

    return 0;
}
```

Zadatak 1

Modifikovati `Primer 1` tako da je moguće unositi proizvoljan broj prirodnih brojeva. Umesto unosa `n` na početku, postaviti ga na početnu vrednost 5. Svaki put kad se kapacitet niza popuni, proširiti `n` množenjem sa 2 i pomoću funkcije `realloc` napraviti, novi, veći niz te dimenzije. Unos podataka se završava kada korisnik unese `-1`.

Primer rada programa:

```
Unositi prirodne brojeve. Uneti -1 za kraj unosa.
Broj: 3
Broj: 5
Broj: 7
Broj: 2
Broj: 6
Broj: 10
Broj: 35
Broj: 23
Broj: 11
Broj: -1

Najveci element je: 35
```