Osnovi programiranja

2023/24

Datoteke

Skladišta podataka

- Sva skladišta podataka u računaru se mogu grubo podeliti na primarna i sekundarna, a njihova uloga je **privremeno** ili **dugotrajno** čuvanje podataka. Primarno skladište često nazivamo i **radna memorija**, ili skraćeno samo memorija. Radna memorija je jedino skladište podataka koje je direktno dostupno procesoru. Procesor čita i izvršava instrukcije smeštene u memoriji u kojoj su, takođe, smešteni i svi podaci sa kojima se operiše. Kao primarno skladište podataka današnji računari najčešće koriste takozvanu **RAM memoriju** (Random Access Memory).
- Da bi bilo moguće izvršavanje bilo kog programa, bez obzira u kom programskom jeziku je napisan, neophodno je da se on u celini ili delimično nalazi u primarnoj memoriji računara. Tako se i sve instrukcije programa napisanog u C-u pre izvršenja moraju naći u primarnoj memoriji. Takođe, prilikom deklarisanja svake promenljive, ona dobija odgovarajući prostor u radnoj memoriji u kojoj će biti smeštena njena vrednost. Međutim, po završetku izvršavanja programa, sve njegove instrukcije i podaci nad kojima je operisao se brišu iz radne memorije, kako bi se oslobodio prostor za druge programe i njihove podatke.

Skladišta podataka

- Drugi veliki nedostatak RAM memorije je taj što podaci u njoj postoje samo dok memorija ima odgovarajuće napajanje električnom energijom. To znači da se prilikom željenog ili neželjenog (nestanak struje i sl.) isključenja računara gube svi podaci koji su u tom trenutku bili smešteni u memoriji. Takođe, primarnu memoriju sa podacima u njoj nije moguće preneti na drugu lokaciju, jer je memorija fizički vezana za računar kako bi imala konstantno napajanje strujom.
- Svi navedeni nedostaci mogu se prevazići nekim od sekundarnih skladišta podataka. Sekundarna skladišta podataka kao što su magnetne trake, magnetni diskovi, optički diskovi, fleš memorije i sl. ne omogućavaju direktan pristup procesoru, već se njima pristupa posredstvom primarne memorije.
- Nedostatak sekundarnih skladišta u odnosu na primarna je višestruko manja brzina pisanja i čitanja podataka, što treba imati u vidu prilikom pisanja programa koji koriste podatke sa ovih uređaja, kako se ne bi ugrozile njihove performanse.

Organizacija podataka na sekundarnim skladištima

- U slučaju sekundarnih skladišta podaci su organizovani po takozvanim datotekama ili fajlovima (eng. file). Datoteka podrazumeva određeni memorijski prostor na sekundarnom skladištu podataka u kome može biti smeštena jedna ili više istorodnih ili raznorodnih informacija. Svaka datoteka na disku mora imati naziv, koji se najčešće sastoji od imena i nastavka (ekstenzije), razdvojenih tačkom. **Ekstenzija fajla** najčešće označava vrstu i format podataka koji su smešteni u tom fajlu.
- Tako, na primer, fajl Proba.txt ima ime Proba i ekstenziju txt, koja u ovom slučaju označava da se radi o tekstualnoj datoteci.
- Direktorijumi omogućavaju lakše snalaženje pri radu sa velikim brojem fajlova, jer su na taj način fajlovi grupisani u logičke celine (eng. folder)
- Dva fajla u različitim direktorijumima mogu imati isti naziv.
- *c:\Informatika\Predavanja\Proba.txt.*

Pristup podacima

- Prema načinu pristupanja podacima smeštenim u datotekama, sve datoteke možemo grubo podeliti na datoteke sa **sekvencijalnim** i **direktnim** pristupom.
- Kod sekvencijalnih datoteka čitanje podataka je moguće samo onim redom kojim su upisivani. To praktično znači da ako bismo želeli da očitamo podatak na desetom mestu, morali bismo prethodno da pročitamo svih devet podataka upisanih pre njega.
- U slučaju datoteka sa direktnim pristupom moguće je pročitati određeni podatak tako što mu se pristupa direktno, navođenjem njegove pozicije u datoteci.
- U zavisnosti od vrste datoteke koja se koristi, programski jezik C omogućava korišćenje datoteka i sa sekvencijalnim i sa direktnim pristupom.

Datoteke u C-u

- Da bi se iz programa napisanog u programskom jeziku C moglo pristupiti nekoj datoteci na disku, potrebno je deklarisati promenljivu datotečnog tipa, koja će praktično predstavljati vezu između C-a i konkretnog fajla na disku. Datotečni tip podatka u C-u je označen rezervisanom rečju **FILE**.

```
FILE *<naziv_promenljive>;
```

- Svi podaci se u datoteku upisuju u **binarnom obliku**, baš onako kako su predstavljeni u memoriji računara, tako da ovakve datoteke nazivamo **binarnim datotekama**. Ovakve datoteke nisu čitljive u tekstualnim editorima, već samo u programima koji zapisane podatke mogu da pročitaju u formatu u kome su oni i upisani u datoteku.
- Prilikom korišćenja svake datoteke pridružuje joj se **nevidljivi pokazivač**, koji određuje do koje pozicije se stiglo sa čitanjem podataka, odnosno koji će podatak biti očitan sledeći.

Povezivanje sa fajlom

- Datotečna promenljiva se povezuje sa konkretnim fajlom na disku korišćenjem funkcije `fopen` čiji su parametri naziv fajla koji želimo da koristimo i string koji označava operaciju koju želimo da vršimo nad fajlom.

```
FILE *<naziv_promenljive> = fopen(<naziv_fajla>,<oznaka_operacije>);
```

```
FILE *f = fopen("ulaz.bin", "rb");
```

ili

```
FILE *f = fopen("c:\\Informatika\\Predavanja\\ulaz.bin", "rb");
```

Upisivanje u datoteku

- Nakon što smo datotečnu promenljivu povezali sa fajlom na disku korišćenjem funkcije fopen i kao argument poslali oznaku operacije `"wb"`, datoteku otvaramo za upisivanje podataka

```
FILE *f = fopen("izlaz.bin", "wb");
```

- Ukoliko fajl sa kojim je povezana datotečna promenljiva nije postojao, kreira se prazan fajl koji je spreman za upis. Ako je fajl postojao, svi podaci u njemu se brišu, a pokazivač za upis novi podataka se postavlja na početak fajla.
- Upisivanje podataka u datoteku se vrši korišćenjem naredbe `fwrite`, koja kao prvi parameter prima promenljivu čiju vrednost želimo da upišemo, zatim koliko memorije zauzima promenljiva tog tipa, što dobijamo pozivom funkcije `sizeof` kojoj prosleđujemo tip promenljive koju upisujemo, treći parametar je koliko promenljivih upisujemo, a četvrti datoteka (datotečnu promenljivu) u koju se podaci upisuju.

```
fwrite(&<promenljiva>,sizeof(<tip_promenljive>),<broj_promenljivih>,<datotecna_promenljiva>);
```

Primer

```
float p = 5;
FILE *f = fopen("izlaz.bin", "wb");
fwrite(&p, sizeof(float), 1, f);

int a[5] = {1, 2, 9, 4, 7};
fwrite(a, sizeof(int), 5, f);
```

- Nakon upisivanja podatka u datoteku, pokazivač se pomera na sledeću poziciju i datoteka je spremna da primi nove podatke.
- Prilikom rada sa datotekama treba imati u vidu da se podaci ne upisuju istog trenutka u datoteku nakon pozivanja komande `fwrite`, već se čuvaju u posebnom delu memorije koji se naziva bafer. Sekundarna skladišta podataka su znatno sporija od radne memorije, pa se korišćenjem `bafera` izbegava često obraćanje ovim uređajima i kvarenje performansi programa. Podaci se prebacuju na sekundarno skladište tek kada se bafer napuni, čime se veća količina podataka upisuje odjednom na disk ili drugi uređaj.

Čitanje iz datoteke

- Da bismo otvorili datoteku za čitanje funkciji fopen kao oznaku operacije "rb“.

```
FILE *f = fopen("ulaz.bin", "rb");
```

- Ovako pozvana naredba fopen otvara datoteku za čitanje i pri tome pozicionira pokazivač na početak datoteke.
- Čitanje podataka iz datoteke se vrši korišćenjem naredbe fread, parametri koji se prosleđuju su isti kao kod fwrite, samo što sada kao prvi argument šaljemo promenljivu u koju želimo da upišemo pročitanu vrednost.

```
fread(&<promenljiva>,sizeof(<tip_promenljive>),<broj_promenljivih>,<datotecna_promenljiva>);
```

Primer

```
float p;
FILE *f = fopen("ulaz.bin", "rb");
fread(&p, sizeof(float), 1, f);

int a[5];
fread(a, sizeof(int), 5, f);
```

- Nakon čitanja podatka iz datoteke, pokazivač se pomera na sledeću poziciju i datoteka je spremna za učitavanje sledećeg podatka.
- Prilikom čitanja podataka iz datoteke neophodno je stalno proveravati da li se stiglo do kraja datoteke korišćenjem funkcije feof čiji je argument datotečna promenljiva:

```
feof(f);
```

- U slučaju da je pokazivač stigao do kraja datoteke funkcija vraća vrednost različitu od 0, što je znak da više nije moguće čitanje iz datoteke. U suprotnom, funkcija vraća vrednost 0.
- Nakon završetka rada sa datotekom neophodno je istu zatvoriti korišćenjem naredbe `fclose` čiji je jedini parametar datoteka koju želimo da zatvorimo:

```
fclose(f);
```

Primer 1

- Napisati program koji učitava podatke o studentima sa tastature i zapisuje ih u binarnu datoteku Studenti.dat.

```
#include <stdio.h>

typedef struct
{
    char ime[20];
    char adresa[20];
    int indeks;
} Student;
```

Primer 1

```
main()
{
    Student s;
    int n, i;
    FILE *f = fopen("Studenti.dat", "wb");
    printf("Unesite broj studenata:\n");
    scanf("%d", &n);
    for(i = 0; i < n; i++)
    {
        printf("Ime: "); scanf("%s", s.ime); getchar();
        printf("Adresa: "); gets(s.adresa);
        printf("Indeks: "); scanf("%d", &s.indeks);

        fwrite(&s, sizeof(Student), 1, f);
    }
    fclose(f);
}
```

Primer 2

- Napisati program koji iz postojeće datoteke Studenti.dat, formirane u prethodnom primeru, učitava podatke o studentima i ispisuje ih na ekranu u tabelarnom obliku.

```
#include <stdio.h>

typedef struct
{
    char ime[20];
    char adresa[20];
    int indeks;
} Student;
```

Primer 2

```
main()
{
    Student s;
    int n, i;
    FILE *f = fopen("Studenti.dat", "rb");
    while(!feof(f))
    {
        fread(&s, sizeof(Student), 1, f);
        printf("%10d%20s%30s\n", s.indeks, s.ime, s.adresa);
    }
    fclose(f);
}
```

Tekstualne datoteke

- Pored binarnih datoteka, C omogućava rad i sa posebnom vrstom datoteka koje se nazivaju **tekstualne datoteke**.
- Tekstualna datoteka je niz znakova zapisan na nekom sekundarnom skladištu podataka.
- Svaki znak je predstavljen odgovarajućim ASCII kodom, odnosno celim brojem koji zauzima jedan bajt. Tako se, na primer, veliko slovo "A" predstavlja ASCII kodom 65. Pored vidljivih znakova, postoje i znaci koji se ne ispisuju grafički, ali su neophodni za uređenje teksta.
- Za razliku od binarnih datoteka tekstualne datoteke se otvaraju sa drugim oznakama operacija.

Operacije	Oznaka operacije
čitanje	“r”
pisanje	“w”
dodavanje	“a”

Tekstualne datoteke

```
FILE *<naziv_promenljive> = fopen(<naziv_fajla>,<oznaka_operacije>);
```

```
FILE *f = fopen("ulaz.txt", "r");
```

- Pre korišćenja datotečne promenljive, neophodno ju je povezati sa odgovarajućim fajlom na disku korišćenjem funkcije `fopen`, kao i u slučaju binarnih datoteka. Samo što se ovog puta koriste druge oznake operacija.
- Važno je naglasiti da se pisanje i čitanje iz tekstualne datoteke vrši na skoro pa isti način kao i ispisivanje na ekran ili učitavanje sa tastature, sa dve razlike. Prva razlika je u tome što umesto `printf` za pisanje i `scanf` za čitanje koristimo `fprintf` za pisanje i `fscanf` za čitanje. Druga razlika je što kao prvi argument naredbama fprintf i fscanf mora navesti datoteka (datotečna promenljiva) u koju se upisuje, odnosno iz koje se čitaju podaci:

```
fscanf(f,"%d",&n);
fprintf(f,"%5d",n);
```

Primer 3

- Napisati program koji sa tastature učitava n prirodnih brojeva i upisuje ih u tekstualnu datoteku Matrica.txt, tako da u svakom redu bude k brojeva. Poslednji red u datoteci može imati i manje od k brojeva, ukoliko n nije celobrojan umnožak broja k.

```
#include <stdio.h>
main()
{
    int n, i, k, x;
    FILE *f = fopen("Matrica.txt", "w");
    printf("Unesite n i k:\n");
    scanf("%d%d", &n, &k);
    for(i = 0; i < n; i++)
    {
        scanf("%d", &x);
        fprintf(f,"%5d",x);
        if((i+1) % k == 0)
                fprintf(f,"\n");
    }
    fclose(f);
}
```

Primer 4

- Iz tekstualne datoteke primeru učitati brojeve i smestiti ih u odgovarajuću matricu. Prva dva broja u datoteci predstavljaju broj redova (m) i broj kolona (n), nakon čega sledi m*n celih brojeva. Elemente dobijene matrice odštampati na ekranu u matričnom obliku.

```
#include <stdio.h>
main()
{
    int a[100][100];
    int n, m, i, j;
    FILE *f = fopen("Matrica.txt", "r");
    fscanf(f,"%d%d",&m,&n);
    for(i = 0; i < m; i++)
        for(j = 0; j < n; j++)
                fscanf(f,"%d", &a[i][j]);
    for(i = 0; i < m; i++)
    {
        for(j = 0; j < n; j++)
        printf("%5d", a[i][j]);
        printf("\n");
    }
}
```