30. TEKSTUALNE DATOTEKE

Datoteka je osnovna logička jedinica za zapis podataka preko jedinica masovne memorije koja se koristi za unos i izlaz velikog obima podataka u/iz programa.

Tekstualne datoteke sastoje se od niza znakova koji je znakovima za prelazak u novi red ('\n') podeljen u redove. Podela tekstualnih datoteka u redove je logička, a ne fizička organizacija. Tekstualne datoteke su samo dugački niz znakova. Sadržaj čine podaci koji odgovaraju karakterima nekog kodnog sistema: ASCII, EBCDIC, UTF-8, UTF-16.

U radu sa tekstualnim datotekama se primenjuju sledeće operacije:

- **Kreiranje datoteke** – formiranje nove datoteke.
- **Otvaranje datoteke** – uspostavljanje veze sa fizičkim fajlom datoteke. Tom prilikom se u operativnoj memoriji kreira određena struktura podataka koja omogućava efikasan pristup datoteci. Ta struktura sadrži određene podatke o trenutnom stanju i statusu datoteke.
- **Pristup datoteci** – realizuje prenos podataka u datoteku ili iz datoteke sa ili bez konverzije u toku prenosa.
- **Ispitivanje statusa datoteke** – dobijanje informacija o eventualnim greškama koje su nastale u toku prenosa podataka.
- **Zatvaranje datoteke** – završna radnja, kojom se raskida veza sa fizičkom datotekom. Tom prilikom poništavaju se strukture podataka stvorene prilikom otvaranja datoteke.

Otvaranje datoteke - Pre bilo kakve obrade podataka datoteka mora biti otvorena. To se postiže funkcijom **`fopen()`**, koja otvara datoteku i vraća pokazivač na tu datoteku (tip **`FILE`**), a ukoliko dođe do greške, funkcija vraća vrednost **NULL**. Opšti oblik funkcije `fopen()` je:

```
FILE fopen(const char *putanja, const char *access_mode);
```

gde je:

- **putanja** - niz znakova koji sadrži kompletnu putanju/stazu do navedene datoteke.
- **access_mode** - određuje način pristupa datoteci. Dozvoljene vrednosti za **access_mode** su:
  - **"r"** - otvara tekstualnu datoteku samo za čitanje.
  - **"w" -** otvara tekstualnu datoteku samo za pisanje, odbacujući postojeći sadržaj ako postoji bez opomene, ili kreirajući datoteku ako ona ne postoji.
  - **"a" -** otvara tekstualnu datoteku za pisanje, dodajući nove elemente na njen kraj, ili stvarajući potpuno novu datoteku ako je nema.
  - **"r+" -** otvara tekstualnu datoteku za ažuriranje, čitanje i pisanje, postavljajući pokazivač na početak datoteke.
  - **"w+" -** otvara tekstualnu datoteku za ažuriranje, čitanje i pisanje, odbacujući postojeći sadržaj ako postoji, ili kreira datoteku ako ona ne postoji.
  - **"a+" -** otvara tekstualnu datoteku za ažuriranje, čitanje i pisanje, dodajući nove elemente na njen kraj, ili stvara novu datoteku ako je nema.

Primer:

```
FILE *dat;
dat=fopen("datoteka.txt","r");
```

Nakon poziva funkcije fopen(), pokazivač dat pokazuje na datoteku, ali on u suštini ne pokazuje na stvarnu datoteku, već na strukturu **FILE.** Struktura **FILE** sadrži članove:

- pokazivač na prihvatnu memoriju (bafer);
- broj bajtova koje treba preneti;
- početnu adresu prihvatne memorije;
- stanje indentifikatora;
- brojna oznaka datoteke.

U ovom primeru deklaracijom pokazivača `dat` saopšteno je prevodiocu da je dat pokazivač koji će sadržati u sebi adresu promenljive tipa **`FILE`**. Funkcija `fopen()` traži datoteku `primer.txt` u direktorijumu gde se nalazi izvršni fajl programa. Ako ga pronađe, formiraće strukturu tipa **`FILE`**, upisaće brojnu oznaku datoteke, početnu adresu prihvatne memorije i inicijalizovaće pokazivač na prihvatnu memoriju. Nakon popunjavanja structure podataka, pokazivaču `dat` će biti prosleđena adresa strukture FILE. Druge funkcije će kasnije koristiti strukturu preko pokazivača `dat`, kako bi na osnovu početne adrese prihvatne memorije i pokazivača na prihvatnu memoriju smeštale podatke na prihvatnu memoriju.

Zatvaranje datoteke vrši se funkcijom **`fclose(),`** čiji je opšti oblik:

```
int fclose(FILE *dat);
```

Zatvaranje datoteke znači da se prekida veza koja postoji između pokazivača na datoteku i imena datoteke, tj. pokazivača na strukturu `FILE`, koji dobija vrednost `NULL`. Prilikom izvršavanja funkcije `fclose()`, a pre prekidanja veze između pokazivača na datoteku i imena datoteke, dolazi do ispisivanja sadržaja prihvatne memorije u datoteku. Funkcija vraća vrednost 0 u slučaju uspešnog zatvaranja datoteke, ili **EOF** u slučaju otkrivanja grešaka. Završetak izvršavanja programa dovodi do automatskog zatvaranja svih otvorenih datoteka. Programi rade optimalnije kada ima otvoreno manje datoteka (do oko 20 datoteka). Zatvorenoj datoteci ne možemo pristupiti bez ponovnog otvaranja. Kada se završi program, sve otvorene datoteke se zatvaraju.

Primer - Definisanje pokazivača na datoteku, otvaranje datoteke PRIMER.TXT za čitanje, a ako datoteka nije uspešno otvorena prikazati odgovarajuću poruku:

```
FILE *dat;
if ((dat = fopen("primer.txt","r")==NULL)
{
     printf("Datoteka primer.txt ne moze biti otvorena!");
     return;
}
fclose(dat);
```

Opšti oblici funkcija za rad sa tekstualnim datotekama:

- čitanje jednog znaka iz datoteke: `int fgetc(FILE *dat); int getc(FILE *dat); int getchar(void);`
- upisivanje jednog znaka u datoteku (znak): `int fputc(int znak, FILE *dat); int putc (int znak, FILE *dat); int putchar (int znak);`
- čitanje linije teksta iz datoteke: `char *fgets(char *tekst, int n, FILE *dat); char *gets(char *tekst);` maksimalan broj znakova kod `fgets` je n-1.

Pozicioniranje unutar datoteke - čitanje iz datoteke `dat` najviše `br` objekata veličine `vel` u memoriju od adrese `niz`:

```
int fread(void *niz, int vel, int br, FILE *dat);
```

Čitanje počinje od tekuće pozicije. Pprogramer mora da vodi računa o logičkoj strukturi datoteke (kako je upisana). Vrednost funkcije - broj pročitanih objekata:

```
n=fread(vektor, sizeof(int), n_max, podaci);
```

Upisivanje u datoteku `dat` `br` objekata veličine `vel` u memoriju od adrese `niz`:

```
int fwrite (void *niz, int vel, int br, FILE *dat);
int fseek(FILE *dat,long pomeraj,int reper);
void rewind(FILE *dat); pozicioniranje na početak datoteke
int feof(FILE *dat); ispitivanje indikatora kraja datoteke
int ferror(FILE *dat);
```

Pisanje počinje na poziciji gde je prethodni pistup završen. Ako trenutna pozicija nije na kraju datoteke, prepisuje se preko starih podataka. Ako do kraja nema mesta, datoteka se proširuje. Pozicioniranje unutar datoteke pretstavlja direktan pristup datoteci. Najčešće se za pozicioniranje unutar datoteke koriste funkcije **fseek()**, **ftell()** i **rewind()**. Primer opšteg oblika:

```
int fseek(FILE *dat, long offset, int mode);
```

Funkcija **fseek()** podešava poziciju datoteke za tok **dat**. Narednim čitanjem ili pisanjem pristupa se podacima na početku te nove pozicije. Datoteka se tretira kao niz bajtova, a funkcija **fseek** omogućava da se direktno pristupi do bilo kog bajta u datoteci. Argument **mode** označava polaznu tačku, i može imati jednu od tri vrednosti zapisanih u simboličkim konstantama:

- **SEEK_SET** početak datoteke,
- **SEEK_CUR** trenutna pozicija,
- **SEEK_END** kraj datoteke.

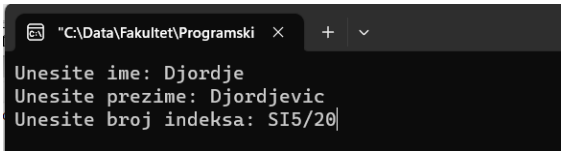
Simboličke konstante **SEEK_SET**, **SEEK_END** i **SEEK_CUR** definisane su u zaglavlju "**stdio.h".** Argument **offset** određuje koliko daleko u bajtovima treba da se pomeramo od početne tačke. Ako **offset** ima pozitivnu vrednost kreće se napred, ili negativnu vrednost kreće se nazad. U slučaju **tekstualne datoteke offset** može da ima vrednost nula ili vrednost koju je dala funkcija **`ftell()`**, dok **mode** mora imati vrednost **SEEK_SET**. Funkcija fseek vraća vrednost 0 ako je uspešno obavila posao, ili -1 u suprotnom (pokušaj pozicioniranja izvan granica datoteke). Funkcija fseek mora obavezno da se poziva pre prvog čitanja posle nekoliko upisivanja, odnosno pre prvog upisivanja posle nekoliko čitanja. Na primer prepisujemo jedan podatak preko drugog, a potom hoćemo da čitamo sledeći podatak. Bez obzira što se čita sledeći bajt mora se pozvati funkcija fseek.

Primer program koji koristi datoteke – U tekstualnu datoteku "studenti.txt", koja sadrži podatke o studentima: ime c[20], prezime c[20], brojindeksa c[20] upisati podatke o studentima koji se unose sa tastature.

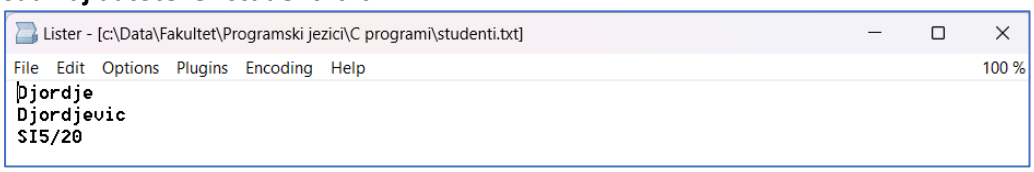
```
#include <stdio.h>
#include <string.h>
#define MAKS 20
int main()
{
    char str[1000];
    char ime[MAKS],prezime[MAKS],brind[MAKS];
    FILE *dat;
    dat=fopen("studenti.txt", "w");
    if (dat==NULL)
    {
        printf("Neuspesno otvaranje datoteke.\n");
        return 1;
    }
    else
```

```
    {
        printf("Unesite ime: ");
        gets(ime);
        printf("Unesite prezime: ");
        gets(prezime);
        printf("Unesite broj indeksa: ");
        gets(brind);
        fprintf(dat, "%s\n", ime);
        fprintf(dat, "%s\n", prezime);
        fprintf(dat, "%s\n", brind);
        fclose(dat);
    }
    return 0;
}
```

Rezultat izvršavanja:



Sadržaj datoteke "studenti.txt":

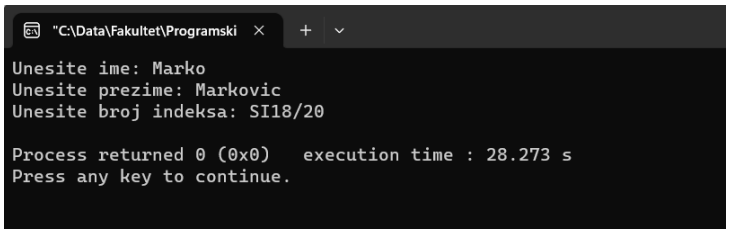


Primer program koji koristi datoteke – U tekstualnu datoteku "studenti.txt", koja sadrži podatke o studentima: ime c[20], prezime c[20], brojindeksa c[20] upisati podatke o studentima koji se unose sa tastature. Napomena: podaci se dodaju na kraj datoteke.

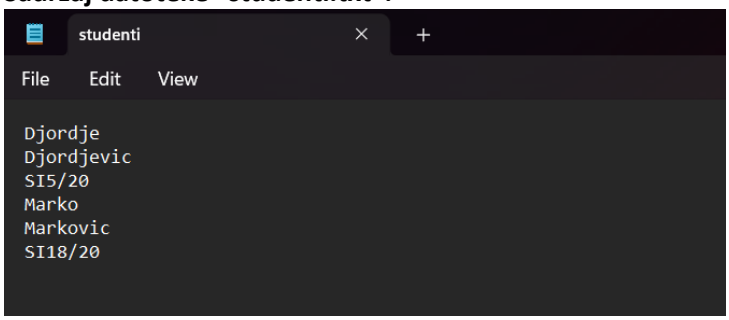
```
#include <stdio.h>
#include <string.h>
#define MAKS 20
int main()
{
    char str[1000];
    char ime[MAKS],prezime[MAKS],brind[MAKS];
    FILE *dat;
    dat=fopen("studenti.txt", "a");
    if (dat==NULL)
    {
        printf("Neuspesno otvaranje datoteke.\n");
        return 1;
    }
    else
    {
        printf("Unesite ime: ");
        gets(ime);
        printf("Unesite prezime: ");
        gets(prezime);
        printf("Unesite broj indeksa: ");
        gets(brind);
        fprintf(dat, "%s\n", ime);
        fprintf(dat, "%s\n", prezime);
        fprintf(dat, "%s\n", brind);
        fclose(dat);
```

```
    }
    return 0;
}
```

Rezultat izvršavanja:



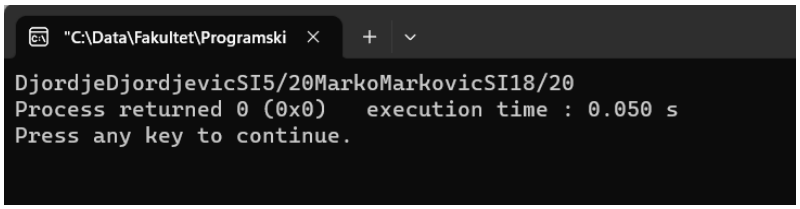
Sadržaj datoteke "studenti.txt":



Primer program koji koristi datoteke – Iz tekstualne datoteku "studenti.txt", koja sadrži podatke o studentima: ime c[20], prezime c[20], brojindeksa c[20] učitati sve podatke o studentima i prikaziti ih na ekranu tabelarno.

```
#include <stdio.h>
int main()
{
    char str[1000];
    FILE *dat;
    dat=fopen("studenti.txt", "r");
    if (dat==NULL)
    {
        printf("Neuspesno otvaranje datoteke.\n");
        return 1;
    }
    else
    {
        while (fscanf(dat, "%s", str)!=EOF)
            printf("%s",str);
        fclose(dat);
    }
    return 0;
}
```

Rezultat izvršavanja:




```
#include <stdio.h>
int main()
{
    char str;
    FILE *dat;
    dat=fopen("studenti.txt", "r");
    if (dat==NULL)
    {
        printf("Neuspesno otvaranje datoteke.\n");
        return 1;
    }
    else
    {
        while ((str=fgetc(dat))!=EOF)
           putchar(str);
        fclose(dat);
    }
    return 0;
}
```

Rezultat izvršavanja:

31. BINARNE DATOTEKE

Datoteka je osnovna logička jedinica za zapis podataka preko jedinica masovne memorije koja se koristi za unos i izlaz velikog obima podataka u/iz programa. **Binarne datoteke se** sastoje od niza bajtova čiji je sadržaj verna slika načina predstavljanja podataka u memoriji. U toku prenosa se ne primenjuje konverzija, već se vrši prosto prenošenje podataka bajt po bajt.

Za svaku datoteku koja se koristi u programu mora da postoji pokazivač na podatak tipa **FILE**. Neka je to pokazivač pod nazivom **datoteka**, pri tome deklaracija je:

```
FILE *dat;
```

Otvaranje datoteke se postiže funkcijom `fopen()`, koja otvara datoteku i vraća pokazivač na tu datoteku (tip **`FILE`**), a ukoliko dođe do greške, funkcija vraća vrednost **NULL**. Opšti oblik funkcije `fopen()` je:

```
FILE fopen(FILE *dat, pristup);
```

gde je:

- **putanja** - niz znakova koji sadrži kompletnu putanju/stazu do navedene datoteke.
- **pristup** - određuje način pristupa datoteci:
  - **"rb"** - otvara binarnu datoteku samo za čitanje.
  - **"wb" -** otvara binarnu datoteku samo za pisanje, odbacujući postojeći sadržaj ako postoji bez opomene, ili kreirajući datoteku ako ona ne postoji.
  - **"ab" -** otvara binarnu datoteku za pisanje, dodajući nove elemente na njen kraj, ili stvarajući potpuno novu datoteku ako je nema.
  - **"rb+" ili "r+b" -** otvara binarnu datoteku za ažuriranje, čitanje i pisanje, postavljajući pokazivač na početak datoteke.
  - **"wb+" ili "w+b" -** otvara binarnu datoteku za ažuriranje, čitanje i pisanje, odbacujući postojeći sadržaj ako postoji, ili kreira datoteku ako ona ne postoji.
  - **"ab+" -** otvara binarnu datoteku za ažuriranje, čitanje i pisanje, dodajući nove elemente na njen kraj, ili stvara novu datoteku ako je nema.

Čitanje iz binarne datoteke se postiže funkcijom `fread()`:

```
fread(&prom, sizeof(prom), 1, FILE *dat);
```

Pisanje u binarnu datoteku se postiže funkcijom `fwrite()`:

```
fwrite(&prom, sizeof(prom), 1, FILE *dat);
```

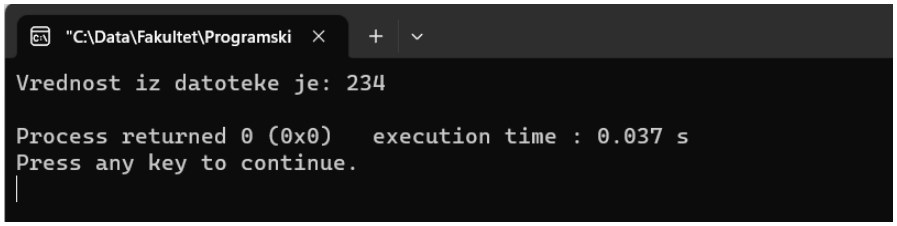
Zatvaranje datoteke vrši se funkcijom `fclose()`, čiji je opšti oblik:

```
int fclose(FILE *dat);
```

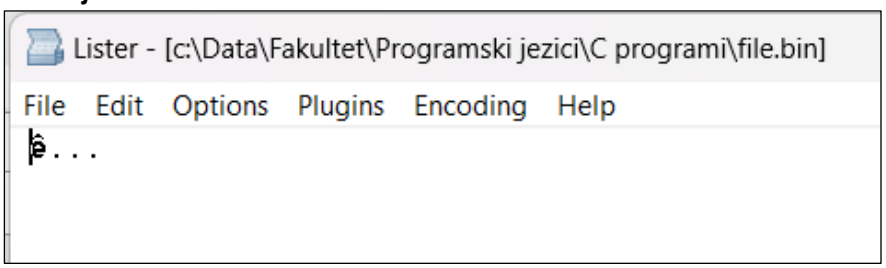
Primer:

```
#include <stdio.h>
int main()
{
    // kreiranje datoteke
    int x=234;
    FILE *dat=fopen("file.bin", "wb");
    if (dat!=NULL)
    {
        fwrite(&x, sizeof(x), 1, dat);
        fclose(dat);
    }
    // citanje iz datoteke
    x=7;
    dat=fopen("file.bin", "rb");
    if (dat!=NULL)
    {
        fread(&x, sizeof(x), 1, dat);
        fclose(dat);
    }
    // provera uradjenog
    printf("Vrednost iz datoteke je: %d\n", x);
    return 0;
}
```

Rezultat izvršavanja:



Sadržaj datoteke "file.bin":



32. STRUKTURA, DEFINICIJA STRUKTURE, DEKLARACIJA STRUKTURE, PRISTUP ČLANOVIMA STRUKTURE

Definicija strukture - Struktura predstavlja **izvedeni tip podatka koji sadrži promenljive istog ili različitog tipa**. Koristi se u slučajevima kada su podaci u međusobnoj semantičkoj vezi, jer se mogu grupisati pod istim imenom. Strukture omogućavaju kreiranje korisnički orjentisanih tipova podataka. Izuzetno su korisni i upotrebljivi pri organizaciji kompleksnijih podataka, naročito u većim i složenijim programima, s obzirom da nude mogućnost obrade grupe međusobno povezanih promenljivih kao jedne celine.

Opšti oblik deklaracije šablona strukture:

```
struct ime_strukture
      {
            tip_komponente1 naziv_komponente1;
            tip_komponente2 naziv_komponente1;
            …
            tip_komponenteN naziv_komponenteN;
      };
```

Strukturirani (složeni) tip podataka koji definiše korisnik ima sledeće **osobine**:

- sastoji se od više komponenti,
- komponente imaju identifikatore,
- komponente ne moraju biti istog tipa,
- struktura se smatra jednim objektom (može biti i vrednost funkcije).

Nazivi elemenata, tj. članova strukture se ne mogu koristiti kao samostalne promenljive, postoje samo kao deo strukture kao složenijeg objekta. Članovi strukture mogu biti bilo kog tipa podatka, osnovnih i izvedenih.

Pristup članovima strukture vrši se pomoću operatora člana strukture ".".

```
ime_strukture.naziv_promenljive1
```

Tip polja strukture, promenljive ne može da bude struktura koja se definiše, ali može da bude pokazivač na strukturu.

Primer – struktura student sa osnovnim podacima za jednog studenta:

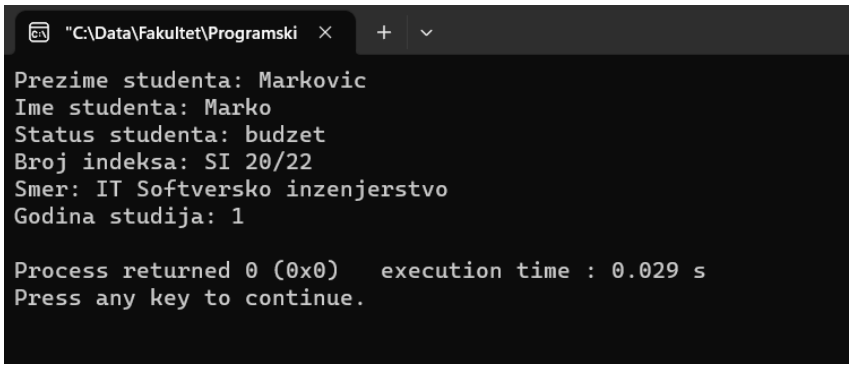
```
#include <stdio.h>
#include <string.h>

struct studenti //definicija strukture
{
      char prezime[30];
      char ime[20];
      char status[10];
      char brojIndeksa[10];
      char smer[50];
      int godinaStudija;
};

int main( )
{
      struct studenti Student; //deklaracija strukture
      strcpy(Student.prezime, "Markovic");
      strcpy(Student.ime, "Marko");
```

```
	strcpy(Student.status, "budzet");
	strcpy(Student.brojIndeksa, "SI 20/22");
	strcpy(Student.smer, "IT Softversko inzenjerstvo");
	Student.godinaStudija =1;
	printf("Prezime studenta: %s\n", Student.prezime); //pristup clanu
	printf("Ime studenta: %s\n", Student.ime);
	printf("Status studenta: %s\n", Student.status);
	printf("Broj indeksa: %s\n", Student.brojIndeksa);
	printf("Smer: %s\n", Student.smer);
	printf("Godina studija: %d\n", Student.godinaStudija);
	return 0;
}
```

Rezultat izvršavanja:



33. INICIJALIZACIJA STRUKTURE, NIZ STRUKTURA

Struktura predstavlja izvedeni tip podatka koji sadrži promenljive istog ili različitog tipa koji se mogu grupisati pod istim imenom.

Inicijalizacijom strukture obezbeđuje se da svi članovi strukture imaju definisanu početnu vrednost. Za numeričke tipove podataka uobičajno je dodeliti vrednost 0 (nula), dok je kod stringovnih promenljivih uobičajno je da se funkcijom `strcpy` koja se nalazi u "string.h" biblioteci, stringovnoj promenljivi upiše prazan string kao početna vrednost. Prazan string sadrži samo znak za kraja stringa, tj. '\0'. Kada član niza predstavlja pokazivač, moramo se postaviti na vrednost ništa - NULL.

Ako se sa `typedef` definiše strukturni tip, njegov identifikator se može koristiti kao oznaka tipa i bez službene reči `struct`.

Primer strukture radnik, definicija, deklaracija i inicijalizacija:

```
struct radnici // Definicija strukture radnici
{
      char prezime[30];
      char ime[20];
      unsigned starost;
      unsigned staz;
} Radnik;
Radnik A, B; //Deklaracija strukturne promenljive A i B tipa
Radnik.strcpy(A.prezime, ""); Inicijalizacija struk
promenljive A
strcpy(A.ime, "");
A.starost=0;
A.staz=0;
B=A; //Naredba dodele vrednosti za strukture
```

Prilikom naredbe dodele vrši se automatsko kopiranje odgovarajućih polja jedne strukture u drugu. U ovom slučaju vrednosti polja strukture A su prekopirane u strukturu B. Ovim se skraćuje kod programa i ne mora se koristiti funkcija `strcpy` da bi se polje koje predstavlja string prekopirao iz polja strukture A u polje strukture B.

Strukturna promenljiva se može inicijalizovati i prilikom deklaracije.

```
Radnik C={"", "", 0, 0};
```

Mogu se definisati i nizovi struktura. **Kada je programu potrebno koristiti više struktura istog tipa, tada se koristi niz**, pri čemu svaki član niza predstavlja strukturni tip podatka.

Ako u primeru želimo koristiti više radnika tada koristimo niz struktura. Niz struktura se definiše na isti način kao bilo koji niz. Svaki član niza je tipa strukture, a u ovom primeru je to Radnik.

```
Radnik X[50], x; //x je strukturna promenljiva
                 //dok X predstavlja niz struktura
```

Inicijalizacija niza struktura vrši se kao i inicijalizacija bilo kog niza. Prvo inicijalizujemo strukturnu promenljivu x:

```
strcpy(x.prezime, ""); // x je prazan string.
strcpy(x.ime, "");
x.starost = 0;
x.staz = 0;
```

Zatim se niz struktura inicijalizuje pomoću ciklusa tako što svaki element niza dobija vrednost x, koja je predhodno inicijalizovana.

```
unsigned i;
for(i=0;i<50;i++) X[i]=x;
```

Prilikom pristupa odgovarajućem polju i-tog člana niza strukture pišemo:

```
X[i].prezime = "Simic";
X[i].ime = "Sima";
X[i].starost = 44;
X[i].staz = 24;
```

Prilikom pristupa polju strukture, prvo se navodi naziv niza, potom indeks niza u zagradama [], a na kraju operator tačka '.' da bi se pristupilo članu strukture.

Razlike niza struktura u odnosu na običan niz:

- kod niza su elementi istorodne komponente - zasebni objekti, identifikuju se indeksima.
- vrednost funkcije ne može da bude niz (može pokazivač).

Niz struktura se koristi za složene objekte – logički povezane informacije.
Komponente niza struktura mogu biti vidljive samo unutar strukture kojoj pripadaju. Mogu se koristiti ista imena promenljivih u različitim strukturama.

Dozvoljena je dodela vrednosti (=) za promenljive istog strukturnog tipa. Dve strukturne promenjive su istog tipa ako su definisane istom naredbom i za njihovu definiciju se koristi isti identifikator tipa.

Strukture mogu pojaviti kao argumenti funkcije, **odnosno kao vrednost koju funkcija vraća** (pri dodeli vrednosti). Mogu se prenositi kao celine, prenosom komponenti ili prenosom adrese strukture (pokazivača na strukturu). Vrednost funkcije takođe može biti tipa strukture ili pokazivača na strukturu.

Strukture se ne mogu porediti.

Pokazivači na strukture - definisanje pokazivača na strukturu se realizuje kao i definicija pokazivača na bilo koju promenljivu.

```
struct radnici *sp;
```

Na ovaj način se smešta adresa strukturne promenljive u gore definisanu pokazivačku promenljivu.
Da bi se našla adresa strukturne promenljive, koristi se operator ˝&˝ operator pre imena strukture:

```
sp=&RadnikA;
```

Pristup članovima strukture pokazivača realizuje se pomoću operatora ˝->˝:

```
sp->prezime="";
sp->ime="";
```

Primer strukture koja se realizuje preko pokazivača:

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

struct radnici // Definicija strukture radnici
{
char prezime[30];
char ime[20];
unsigned starost;
unsigned staz;
};

void prikazRadnika(struct radnici *radnik);
```

```
void main()
{
	struct radnici Radnik;
	strcpy(Radnik.prezime, "Djukic");
	strcpy(Radnik.ime, "Djordje");
	Radnik.starost = 65;
	Radnik.staz = 40;
	prikazRadnika(&Radnik);
}

void prikazRadnika(struct radnici *radnik)
{
	printf("Prezime: %s\n", radnik->prezime);
	printf("Ime: %s\n", radnik->ime);
	printf("Starost: %d\n", radnik->starost);
	printf("Staz: %d\n", radnik->staz);
}
```

Rezultat izvršavanja: