Programski jezici i strukture podataka

6

STRUKTURE

Strukture u C-u

- Jezik C ima skup osnovnih tipova podataka i od njih se na nekoliko načina dobijaju tipovi višeg nivoa.
- Podatke koji imaju neku međusobnu vezu možete da grupišete pod isto ime i da im pristupite pomoću tog imena i izbora unutrašnjeg člana.

Deklaracije struktura

- Opšti oblik deklaracije strukture u jeziku C je sledeći:

```
struct oznaka {
                tip imeprom;
                };
```

- Deklaracija strukture započinje ključnom reči ***struct***.
- Opciona ***oznaka*** daje konkretnoj strukturi jedinstveno ime.
- Lista deklaracija promenljivih specificira članove strukture.
- Proizvod ovakve deklaracije je šablon strukture koji možete da koristite kao specifikator tipa.

```
struct oznaka{
                tip imeprom;
              };
```

Šablon meseci u godini

- Evo deklaracije koja kreira šablon za strukturu korišcenu za smeštaj informacije o mesecima u godini:

```
struct mesec_st { char ime[10];
                  char skr[4];
                  short dana;
                     };
```

- Sufiksi "_st" ili "_t" su uobičajeni za konvenciju da je ime koje ga sadrži oznaka strukture.
- U standardnoj biblioteci jezika C mnogi izvedeni tipovi imaju sufiks "_t" .

Deklaracija promenljive

- Dva niza znakova u ovoj strukturi zauzimaju prostor fiksne veličine u memoriji.
- U članu ***ime*** se sadrži ime meseca plus završni nulti znak.
- U članu ***skr*** je skracenica od tri slova plus njen završni nulti znak.
- Član ***dana*** je kratak ceo broj, koji zauzima dva bajta.
- Pošto su članovi ime i skr nizovi, oni zauzimaju podjednako veliki memor- ijsko prostor bez obzira šta je u njih upisano.

```
struct mesec_st mesec;
```

Pristup elementima strukture

- Kada se deklariše strukturna promenljiva njenim članovima se mogu dodeliti ili u njih kopirati vrednosti.
- Da bi dodelili broj dana u mesecu januaru promenljivoj ***mesec***, koristi se:

```
mesec.dana = 31;
```

- Tačka se naziva operatorom člana strukture.
- Prvenstvo ovog operatora je na samom vrhu tabele prioriteta, zajedno sa funkcijom (zagrade), nizom (uglaste zagrade).
- Radi pristupa članu strukturne promenljive primenite operator tačka:

```
imeprom.clan
```

Strukture i funkcije

- Struktura može biti parametar funkcije.

```
int prestupGod(struct datum_st);
```

- Funkcija može da vrati strukturu onome ko ju je pozvao.

```
struct datum_st zadajDatum(char, char, char);
```

Pokazivači i strukture

- Izbor člana strukture pomoću pokazivača može se predstavit na dva načina:
  - standardnom notacijom, koristeci ime strukture i operator tačka;
  - korišćenjem pokazivača na strukturu ("->").

```
struct oznaka { tip m;...};

struct oznaka s, *p;//struktura i pokazivac
p = &s;//p sada pokazuje na s
```

- Bez korišcenja pokazivača, članu m pristupa se primenom operatora tačka:

```
s.m
```

- Kada se koristi prethodno inicijalizovan pokazivač, p, izraz *p je struktura (isto kao kada se s korist neposredno).

```
(*p).m
```

- Zagrade su ovde potrebne zato što operator tačka proizvodi čvršcu vezu nego operator posrednog pristupa (*).
- Izbor člana strukture preko pokazivača javlja se često u programima na jeziku C. Iz tog razloga postoji pogodniji oblik:

```
p->m
```

DATOTEKE

Datoteke

- Datoteka predstavlja sekvencu bajtova.
- Funkcija fopen() povezuje datoteku s tokom podataka i inicijalizuje objekat tipa FILE, koji sadrži sve informacije neophodne za upravljanje tokom podataka.
- U takve informacije spadaju **pokazivač na korišćeni bafer**, **indikator pozicije** u datoteci koji određuje mesto pristupa u datoteci, i indikatori (engl. flags) **greške** i **kraja datoteke**.

Datoteke

- Prenošenje tokova podataka u odnosu na bafer može se odvijati na tri načina:
  - **Po punjenju bafera** (engl. fully buffered)
    Znakovi u baferu se normalno prenose samo ako je bafer pun.
  - **Posle znaka za novi red** (engl. line buffered)
    Znaci u baferu se normalno prenose samo ako se u bafer upiše znak za novi red ili ako je pun.
  - **Izvan bafera** (engl. unbuffered.)
    Znakovi se prenose što je brže moguće.

Sfandardni tokovi podataka

Pokazivač	Naziv	Upis
stdin	Standardni ulaznitok	Posle znaka za novi red
stdout	Standardni izlaznitok	Posle znaka za novi red
stderr	Standardni izlazni tok greške	Van bafera

- Tok **stdin** obično je pridružen tastaturi, a tokovi **stdout** i **stderr** ekranu konzole.
- Te veze mogu se izmeniti takozvanim preusmeravanjem (engl. *redirection):* to može uraditi funkcija **freopen()** kada je program pozove ili okruženje u kome se program izvršava.

Otvaranje i zatvaranje datoteka

- Da biste upisali sadržaj u novu datoteku ili izmenili sadržinu postojeće, morate je prvo otvoriti.
- Prilikom otvaranja datoteke, morate navesti *režim pristupa* (engl. *access mode)* kako biste ukazali na to da li planirate da čitate njen sadržaj i/ili da upisujete u nju.
- Kada završite rad s datotekom, zatvorite je da biste oslobodili resurse.

Otvaranje datoteke

- Standardna datoteka sadrži funkciju **fopen()** za otvaranje datoteke. U posebnim slučajevima, datoteka se otvara pomoću funkcija **freopen()** i **tmpfile()**.

```
FILE *fopen( const char * restrict imedat, const char * restrict rezim );
```

- Ova funkcija otvara datoteku čije ime predstavlja znakovni niz ***imedat***. Ime datoteke možeda sadrži i ime direktorijuma.
- Drugi argument, ***rezim,*** takođe je znakovni niz i označava režim pristupa.
- Funkcija fopen() povezuje datoteku s novim tokom podataka imedat.

Otvaranje datoteke

```
FILE *freopen( const char * restrict imedat,
               const char * restrict rezim,
               FILE * restrict tok );
```

- Ova funkcija preusmerava tok podataka.
- Poput funkcije fopen(), otvara određenu datoteku u zadatom režimu.
- Funkcija freopen() povezuje datoteku s postojećim tokom podataka određenim trećim argumentom.
- Datoteka koja je prethodno bila povezana sa tim tokom, zatvara se.
- Funkcija freopen() najčešće se koristi za preusmeravanje standardnih tokova stdin, stdout i stderr. **Z26b.c**

Otvaranje datoteke

```
FILE *tmpfile( void );
```

- Funkcija tmpfile() pravi novu privremenu datoteku, imena različitog od svih postojećih datoteka, i otvara je zbog binarnog upisa i čitanja (kao kada bi se funkcija fopen() pozvala pomoću znakovnog niza režima pristupa "wb+").
- Ako se program normalno završi, datoteka se automatski briše.

Otvaranje datoteke

- Sve tri funkcije za otvaranje datoteke vraćaju pokazivač na otvoreni tok u slučaju uspeha, ili pokazivač na vrednost NULL, u suprotnom.

Režim pristupa

- Režim pristupa određen drugim argumentom funkcije fopen() ili freopen() označava koje su ulazne i izlazne operacije dozvoljene nad novim tokom podataka.
- Dozvoljene vrednosti znakovnog niza za režim pristupa jasno su definisane: prvi znak znakovnog niza koji određuje režim pristupa je uvek:
  - **r** za čitanje („read“),
  - **w** za pisanje („write“), ili
  - **a** za dopisivanje („append“);

  u najjednostavnijem slučaju, znakovni niz sadrži samo jedan znak.

Režim pristupa

- Znakovni niz režima pristupa može da sadrži znak + ili b (ili oba, u proizvoljnom redosledu: sekvence +b i b+ imaju isti efekat).
- Znak plus (+) u znakovnom nizu režima pristupa kazuje da je dozvoljeno i čitanje i upis.
- Ipak, program ne može posle jedne od njih odmah izvršiti drugu.
- Nakon operacije upisivanja, prvo morate pozvati funkciju fflush() ili funkcije za pozicioniranje fseek(), fsetpos() ili rewind() pre operacije čitanja.
- Slično tome, posle operacije čitanja, pre upisivanja morate pozvati funkciju za pozicioniranje.

Režim pristupa

- **Znak b** u znakovnom nizu režima pristupa uslovljava otvaranje datoteke u binarnom režimu — to jest, novi tok koji se povezuje sa datotekom **binaran** je.

Režim pristupa

- Ukoliko znakovni niz režima pristupa počinje znakom **r**, neophodno je da **datoteka već postoji** u sistemu datoteka.
- Ako je na početku znakovnog niza slovo **w** i datoteka ne postoji, mora se napraviti.
- Ukoliko datoteka već postoji, njen sadržaj biće **izgubljen**, pošto funkcija fopen() veličinu datoteka u režimu upisivanja svodi na nulu.
- Ako znakovni niz režima upisa počinje znakom **a** („append“), a datoteka ne postoji - onda se mora napraviti.
- Ukoliko datoteka postoji, njen sadržaj će se očuvati, jer se sve operacije upisa automatski obavljaju na kraju datoteke.

```
ifinclude <stdio.h> ifinclude <stdbool.h>
_Bool isReadWriteable( const char *filename )
{

FILE *fp = fopen( filename, "r+" );// Otvara datoteku za upis i ispis.
if ( fp != NULL )      // Da li je fopen() uspešna?
{
       fclose(fp);     // Da: zatvara datoteku; bez obrade greške.
       return true;
}
else   // Ne.
       return false;
}
```

```c
// Primer rada sa tekstualnom datotekom – čitanje
#include<stdio.h>
#define MAXL 80

int main()
{
    FILE *pf;
    char str[MAXL];

    pf=fopen("test.txt", "r");

    if(pf!=NULL){
        while(fgets(str, MAXL, pf)!=NULL)
            puts(str);

        fclose(pf);
    } else {
        printf("Nije moguce otvoriti datoteku ili ona ne postoji.");
    }

    return 0;
}
```

Rad sa tekstualnim datotekama

- Funkcijama za prenos znakova vrši se čitanje ili pisanje pojedinačnih znakova ili nizova znakova bez konverzije.
- int fgetc (FILE *dat) ;
- int getc   (FILE *dat) ;
- int getchar (void);

Rad sa tekstualnim datotekama - čitanje

- Ove funkcije čitaju jedan znak iz datoteke **dat** (fgetc i getc) odnosno preko glavnog ulaza (getchar).
- Dok je **fgetc** funkcija, dotle **getc** se ostvaruje kao makro.
- Zato, može da se desi da se u slučaju getc, izraz dat izračunava više puta, što može da predstavlja problem ako taj izraz proizvodi i bočne efekte.
- S druge strane, izvršavanje getc je efikasnije od pozivanja funkcije fgetc. U logičkom smislu, fgetc i getc su istovetne.

Rad sa tekstualnim datotekama - čitanje

- Vređnost sve tri funkcije je kod pročitanog znaka, odnosno simbolička konstanta EOF u slučaju nailaska na kraj datoteke ili u slučaju otkrivanja greške u toku čitanja.

Rad sa tekstualnim datotekama - pisanje

- **fputc (int *zn,** FILE ***dat)**;
- **int putc (int*zn,** FILE ***dat) ;**
- **int putchar (int** zn) ;
- Ove funkcije uspisuju jedan znak zn u datoteku dat (fputc i putc) odnosno ispisuju preko glavnog izlaza (putchar).
- Dok je **fputc** funkcija, dotle **putc** se ostvaruje kao makro. Zato, može da se desi da se u slučaju putc izraz dat izračunava više puta, što može da predstavlja problem ako taj izraz proizvodi i bočne efekte.
- S druge strane, izvršavanje putc je efikasnije od pozivanja funkcije fputc.
- U logičkom smislu, fputc i putc su istovetne.

Rad sa tekstualnim datotekama - ispis

- Vrednost sve tri funkcije je kod ispisanog znaka, odnosno simbolička konstanta EOF u slučaju otkrivanja greške u toku pisanja.
- Evo primera kojim se piše jedan znak u datoteku podaci:

```
greska = fputc ('A', podaci) == EOF;
```

Rad sa tekstualnim datotekama - čitanje stringa

```
char *fgets (char *tekst, int n, FILE *dat);
char * gets (char *tekst);
```

- Ove funkcije čitaju jedan red teksta u znakovni niz tekst iz datoteke dat (fgets) odnosno preko glavnog ulaza (gets).
- Funkcija fgets čitanje završava kada pročita n-1 znakova ili kad pročita znak za kraj reda \n i iza pročitanih znakova dodaje završni znak \0.
- Funkcija gets uvek čita sve znakove do znaka \n, koga ne stavlja u niz *tekst* već ga zamenjuje znakom \0.

Rad sa tekstualnim datotekama - čitanje stringa

- Vrednost obe funkcije je tekst (adresa!), odnosno NULL u slučaju nailaska na kraj datoteke ili u slučaju otkrivanja greške u toku čitanja.
- Primer kojim se čita jedan red teksta u znakovni niz tekst (tip *string*) iz datoteke podaci:

```
greska = fgets (recenica, 55, podaci) == NULL;
```

Rad sa tekstualnim datotekama - pisanje stringa

```
int fputs   (const char *tekst, FILE *dat);
int puts    (const char *tekst);
```

- Ove funkcije uspisuju znakovni niz tekst kao jedan red teksta u datoteku dat (fputs) odnosno ispisuju preko glavnog izlaza (puts).

Rad sa tekstualnim datotekama - pisanje stringa

- Vrednost obe funkcije je ne-negativan broj, odnosno simbolička konstanta EOF u slučaju otkrivanja greške u toku ispisivanja.
- Primer kojim se ispisuje jedna rečenica u datoteku podaci kao zaseban red teksta:

```
greska = fputs ("Dobar dan!", podaci) == EOF;
```

Prenos podataka sa konverzijom

- Funkcije kojima se vrši ulazna ili izlazna konverzija za sva moguća izvorišta i odredišta podataka.
- Bez obzira na izvorište ili odredište podataka, važe sva pravila i tok konverzije ranije spomenuti.

```
int fprintf(FILE*dat, const char *format, argl,
      arg2, ...);
int printf(constchar *format,argl,arg2,...) ;
int sprintf(char*niz, const char *format, argl,
      arg2, ...) ;
```

Prenos podataka sa konverzijom

- Ove funkcije vrše izlaznu konverziju podataka arg1, arg2, ... uz primenu konverzija koje su zadate znakovnim nizom format.
- Pojedini argumenti mogu da budu nuemrički izrazi, odnosno adresni izrazi za znakovne nizove.
- Odredište rezultujućeg niza znakova je datoteka dat (fprintf), glavni izlaz (printf) ili znakovni niz niz (sprintf).
- Argument niz mora da je dovoljno dugačak za prihvatanje svih znakova dobijenih u toku konverzije i za završni znak \0.
- Ovaj poslednji slučaj naziva se internom konverzijom, jer je i rezultat konverzij u operativnoj memoriji.
- Funkcija **sprintf** je korisna za konverziju binarnih oblika numeričkih podataka u nizove cifara koji mogu dalje da se obrađuju kao tekst u raznim programima za obradu teksta.

Prenos podataka sa konverzijom

- Vrednost sve tri funkcije je broj znakova u rezultujućem nizu znakova, odnosno negativna vrednost u slučaju otkrivanja greške u toku konverzije.
- Primer kojim se piše jedna rečenica uz konverziju numeričkog podatka u datoteku podaci:

```
greska = fprintf (podaci, "Broj kuglica = %d\n", kugl) < 0;
```

Prenos podataka sa konverzijom

```
int fscanf ( FILE *dat, const char *format, arg1, arg2, ...) ;
int scanf ( const char *format, arg1, arg2, ...) ;
int sscanf (const char *niz,  const char *format, arg1, arg2,
        ...) ;
```

- Ove funkcije vrše ulaznu konverziju podataka arg1, arg2, ... uz primenu konverzija koje su zadate znakovnim nizom format.
- Pojedini argumenti treba da budu adrese numeričkih podataka ili znakovnih nizova.
- Izvorište niza znakova koji se konvertuje je datoteka dat (fscanf), glavni ulaz (scanf) ili znakovni niz niz (sscanf).
- Ovaj poslednji slučaj naziva se internom konverzijom, jer je i početni niz koji se konvertuje u operativnoj memoriji.
- Funkcija sscanf je korisna, u programima za obradu teksta, za izdvajanje numeričkih podataka iz teksta i njihovo pretvaranje u binarni oblik za potrebe raznih izračunavanja.

- Vrednost sve tri funkcije je broj konvertovanih podataka (ne znakova!), odnosno simbolička konstanta EOF u slučaju nailaska na kraj datoteke ili otkrivanja greške pre konverzije prvog podatka.
- Primer kojim se čita jedan celobrojni podatak uz konverziju iz datoteke podaci:

```
greska = fscanf (podaci, "%d", Skugl) == EOF;
```

```c
// Primer rada sa tekstualnom datotekom – upis
#include <stdio.h>
#include <stdlib.h>
typedef char Tizraz30[31]; // 30 karaktera i ‘\0’
int main()
{
    FILE *IZLdat;
    Tizraz30 ime, prezime, NazivIZLdat;
    printf("Unesite Vase ime: ");
    scanf("%s", ime);
    __fpurge(stdin);
    printf("\nUnesite vase prezime: ");
    scanf("%s", prezime);
    __fpurge(stdin);
    printf("Unesite naziv datoteke u koju zelite da zapisete podatke: ");
    scanf("%s", NazivIZLdat);
    // otvaranje datoteke sa proverom prava na pisanje(w)
    if((IZLdat=fopen(NazivIZLdat,"w")) == NULL) {
        printf("Greska prilikom otvaranja datoteke %s\n",NazivIZLdat);
        // izlaz iz programa u slucaju zabrane pisanja u datoteku
        exit(EXIT_FAILURE);
    }
    fprintf(IZLdat,"%s\n%s",ime, prezime); // upis u izlaznu datoteku
    fclose(IZLdat); // zatvaranje datoteke
    printf("Podaci studenta su upisani u datoteci: %s\n", NazivIZLdat);
    return 0;
}
```