

PROGRAMIRANJE I PROGRAMSKO INŽENJERSTVO

2) Uvod u programiranje - pojam algoritma

Algoritam

- pravilo (ili skup pravila) kojim se opisuje kako riješiti neki problem, i koje posjeduje sljedeća svojstva:
 - preciznost
 - jednoznačnost
 - obuhvaća končan broj koraka, a svaki korak je opisan instrukcijom
 - definirani su početni objekti, koji pripadaju nekoj klasi objekata, nad kojima se obavljaju operacije
 - istodobno obavljanja algoritma je skup završnih objekata (rezultat), tj. algoritam je djelotvoren
- procedura ima ta ista svojstva, ali ne mora završiti u končnom broju koraka (operacijski sustav računala, redatelj teksta)

Programiranje u užem smislu

pogreške:

prevedilac - syntax errors

kolektor - linker errors

izvršavanje - run-time errors \Rightarrow pogreške koje se otvaraju prilikom izvršavanja (npr. dijeljenje s nulom)

rezultati - logičke pogreške \Rightarrow program "radi" (ne dojavljuje pogreške), ali daje pogrešne rezultate

tablica/straga \Rightarrow otkrivanje sadržaja memorijских varijabli tijekom izvršavanja programa (testiranje na papiru)

struktogram \Rightarrow pseudokod

kod u nekom od programskih jezika

osnovni tipovi podataka: int - cijelobrojni tip \Rightarrow unsigned / signed short \leq int \leq long

float - realni tip

double - realni tip u dvostrukoj preciznosti float \leq double \leq long double

char - znakovni tip (ili mali cijeli broj)

Primjer svih sadržaja u registru od 3 bita (ako je prvi bit predznak)

- općenito vrijedi: $[-2^{m-1}, 2^{m-1} - 1]$

Globnije konstante u jeziku C

- konstante pišane u dekadskoj notaciji:

$$7, 20, 64, -110, 8092, 34567821, -176987941, 7u, 20u, 64u$$

- konstante pišane u oktalnoj notaciji:

$$07, 024, 0100, 0177777, 07u, 024u$$

- konstante pišane u heksadekadskoj notaciji:

$$0x7, 0x6E, 0xFFFFFFFF, 0x7u, 0x14u$$

Decimlani brojevi u binarnom sustavu

- smisao "binarnog točku"

- točka se miče množenjem s potencijama baze 2

$$111 * 2^2 = 11100 \quad (\text{množenje})$$

$$1.11 * 2^{-2} = 0.0111 \quad (\text{dijeljenje})$$

Prikaz realnih brojeva u binarnom obliku

- p je predznak [31], p=1: negativan broj, p=0: pozitivan broj

- karakteristika = binarni eksponent + 127, raspon k ∈ [0, 255], raspon BE ∈ [-126, 127] ⇒ bitovi [30-23]

- Mantisa [22-0] ⇒ bez skivenog bita (vodeća jedinica) i decimalne točke

Primjer prikazane realnih brojeva

$$-2 = -10_2 \cdot 2^0 = -1 \cdot 2^0 \Rightarrow p=1, k=1+127=128 = 10000000_2, M=0000\ 0000\ 0000\ 0000\ 0000_2$$

$$= 1100\ 0000\ ...0000_2 = C000\ 0000\ 16$$

$$3.75 = 101.11_2 = 1.0111 \cdot 2^2 \Rightarrow p=0, k=2+127=129 = 10000001_2, M=0.111_2$$

$$= 0|1000\ 0001|0111\ 0000\ 0000\ 0000\ 0000_2 = 40B8\ 0000\ 16$$

- preciznost: broj znamenaka koji opisuje mjeru veličinu

- točnost: bliskost stvarnoj (nepoznatoj) vrijednosti

Veličina osnovnih tipova podataka

char, unsigned char, signed char 1 oktet

$\Rightarrow \text{sizeof}(\text{char})$ vraca broj okteta : 1

short, unsigned short 2 okteta

int, unsigned int 4 okteta

long, unsigned long 4 okteta

float 4 okteta

double 8 okteta

long double 8(10,12,16) okteta

Raspom i preciznost realnih brojeva standarske preciznosti

- podrazumijeva se 7 prvih važećih znamenaka

- dekadski brojevi s više od 7 znamenaka, i koji predstavljaju zbroj potencija broja 2, mogu se pohraniti bez gubitaka (= broj je unutar dopuštenog intervala)

float f_1, f_2

$$f_1 = 212992.265625; \Rightarrow 2^{-6} + 2^{-2} + 2^{14} + 2^{16} + 2^{17}$$

$$f_2 = 212992.365625; \Rightarrow 2^{-6} + 2^{-2} + 2^{14} + 2^{16} + 2^{17} + 0.9 \cdot 10^{-10}$$

`printf ("%18.12f|m", f1);` $\Rightarrow 212992.265625000000$

`printf ("%18.12f|m", f2);` $\Rightarrow \underline{\underline{212992.359375000000}}$

7 znamenaka

3. znakovni i logički tip podataka

prikaz slova i ostalih znakova - ASCII vrijednosti (0-127) American Standard Code for Information Interchange

- 26 velikih slova engleske abecede A-Z (65-90)

- 26 malih slova engleske abecede a-z (97-122)

- 10 znamenaka 0-9 (48-57)

- operatori, interpunkcije i upravljalici znakovi

znakovne konstante

c = 'A'; ekvivalentno je razm c = 65;

- specijalni znakovi unutar navodnika moraju imati ispred sebe znak \

a = '\n'; isto što i a = 49;

- ukoliko se želi dobiti brojčana vrijednost znamenke: broj = a - 48; odnosno broj = a - '0';

niz znakova (string) - unutar dvostrukih navodnika

char ime - miza [duljina - miza + 1];

- kao polje znakova, paziti da se rezervira mjesto za NULL znak ('\0')

- niz znakova uvek završava sa '\0'

zamjena za logički tip podataka

Matematička logika

- osnovni pojam: logički sud

- može biti istinit = 0, ili lažan = 0

- osnovni operatori: negacija \neg , konjunkcija \wedge , disjunkcija \vee , ekskluzivna disjunkcija \otimes

logički operatori

! logičko NE, && logičko I, || logičko IU

<, <=, >, >= Relacijski operatori

④ Kontrolne masebne selekcije

- kontrolna maseba if - jednostrana selekcija \Rightarrow if... ako je niz masebni, onda if h... ?
- kontrolna maseba if - dvostrana selekcija \Rightarrow if... else ...
- višestrana selekcija pomoći masebni if-else if-else
- else pripada "majblžem" if-u koji "mema svoj else"
- ukoliko želimo else dio približiti nekoj drugoj if masebi, moramo poziti na vitičaste zagrade izu if-ova

Ostali operatori

Umarni operatori

sizeof začeće memorije

(tip) pretvorba tipa (cast)

! logičko ne

+ umarni plus

- umarni minus

++ uvećavanje za 1

-- umanjenje za 1

~ inverzija bitova (NOT)

* indirekcijski

& adresni operator

Operatori poređanija i smanjenja za 1

- prefiksni oblik (operator ispred varijable): +a; -a;

\Rightarrow u izrazima se vrijednost varijable prvo uveća/umani, a tek onda "iskoristi" mijena/ vrijednost

- postfiksni oblik (operator izu varijable): a++; a--;

\Rightarrow u izrazima se vrijednost varijable prvo "iskoristi", a nakon toga se varijabla uveća / umani

Bimarni operatori

$*$ / %	množenje, dijeljenje, modulo	
$+$ -	zbrajanje, oduzimanje	
$<<$	pomak bitova u lijevo - umnožak	broj pomaka određen je parametrom sa desne strane operatora
$>>$	pomak bitova u desno - dijeljenje	
$< > <= > =$	relacijski operatori	
$= !=$	operatori jednakosti	
$\&$	logički I po bitovima	
\wedge	isključivi Ili po bitovima	
$ $	veličući Ili po bitovima	
$\&& $	logički I i Ili	

Termini operatori

- operator za uvjetno podesivanje (?:)

uvjetni izraz ? izraz1 : izraz2 ; umjesto if-else maredbi

Kontrolne maredbe - programske petlje

- služe za obavljanje određenog programskog odsječka (tijelo petlje) više puta

- programska petlja s ispitivanjem uvjeta ponavljanja mal početku (\Rightarrow while (...) { ... })

- programska petlja s ispitivanjem uvjeta ponavljanja mal kraj (\Rightarrow do { ... } while (...) ;)

- programska petlja s poznatim brojem ponavljanja (\Rightarrow for (i = poc; i <= kraj; i = i + k) { ... })

- ako je u for petlji potrebno napisati više maredbi u izrazu, one se odvajaju zarezom

- bilo koji od izraza u zagradama se može izostaviti

- tada se petlja izvodi kao da je logička vrijednost tog izraza istinita (true)

Beskonačne petlje

- tijelo petlje izvodi se beskonačno mnogo puta ako ne sadrži:

naredbu za izlazak iz petlje (break)

naredbu za povratak iz funkcije (return)

poziv funkcije za završetak programa (exit)

i li goto naredbu

goto oznaka-naredbe-1;

...

oznaka-naredbe-1:

programski odjeljak

Naredbe break i continue - kontrola toku petlje

- break \Rightarrow prekida izvođenje najbliže vanjske programске petlje

- continue \Rightarrow unutar while i do petlje prebacuje izvođenje programa na ispitivanje uvjeta, a unutar for petlje prebacuje izvođenje programa na korak for petlje, i potom na ispitivanje uvjeta (također se odnosi na najbližu vanjsku petlju)

Naredba switch-srednica

- može se upotrijebiti umjesto višestruke if selekcije

switch (jelobrojni-izraz) {

case const-izraz1 : naredbe1;

[case const-izraz2 : naredbe2;]

...

[default : naredbeni];

}

- aže se unutar case bloka ne mavede klijućom riječ break; nastavak programa je slijedeći case blok u listi!

- "propadanje" po case labelama može se iskoristiti onda kad se "ispod" nekoliko labela

nalazi isti programski kod

① Polje - sizeof vrati ukupnu veličinu polja u oktetima (bajtovima)

- podatkovna struktura ili složeni tip podatka koji obuhvaća više članova
- svakom pojedincu članu polja može se pristupati pomoću indeksa

- potrebno je definisati:

ime varijable

tip podatka za članove polja \Rightarrow svi članovi jednog polja moraju biti istog tipa

broj članova polja \Rightarrow u uglađivim zagradama $[0, \text{BrojElementataPolja} - 1]$

- vrijednosti članova polja su međefinisane ("smjeće"), ukoliko se ne navede unutar $\{ \dots \}$

- članovi za koje nije navedena početna vrijednost postavljaju se na vrijednost 0

- inicijalizacija članova polja na 0 \Rightarrow int polje[2000] = $\{ 0 \}$;

- ako se polju podeljuju početne vrijednosti, u vitičastim zagradama se mora malaziti barem

jedna vrijednost, i broj početnih vrijednosti mora smjeći biti veći od deklarisanog broja članova polja

- broj članova polja može biti definisan brojem navedenih konstanti

float x[] = $\{ 0, 0.25, 0, -0.5 \}$; isto kao: float x[4] = $\{ 0, 0.25, 0, -0.5 \}$;

Polje znakova kao niz znakova (string)

- konstanta unutar dvostrukih navodnika

- ne postoji tip podatka string, već se konisti jednodimenzionalno polje znakova

- na kraju niza znakova je obavezno \0 \Rightarrow označava kraj niza znakova

- treba osigurati prostor za barem jedan znak više, \0 će biti dodan

char ime[4+1] = $\{ 'I', 'V', 'a', 'm', '\0' \}$; isto kao: char ime[4+1] = $\{ 'I', 'V', 'a', 'm' \}$;

Višedimenzionalna polja - sve dimenzije moraju biti zadane!

- jednodimenzionalno polje (vektor) \Rightarrow smještaj u memoriji računala: član za članom

- dvodimenzionalno polje \Rightarrow redak za retkom int polje[MAXRED][MAXSTUP];

- višedimenzionalno polje \Rightarrow sloj za slojem

Primjer transponiranja matrice

- član $\text{mat}[i][j]$ originalne matrice zamjenjuje se s članom $\text{mat}[j][i]$

- treba pozititi da to kad imaju više stupaca, mego redaka i obrnuto:

indeks redka: $i = 0; i < \max(m, n) - 1; i++$

indeks stupca: $j = i + 1; j < \max(m, n); j++$

⑥ Pokazivači (pointers)

- za adresu također se koristi pojam pokazivač, zato jer onaj "pokazuje" na neki objekt u memoriji

- ako je x varijabla, tada je $\&x$ mjesto adresa u memoriji

- adresu nije moguće pohraniti u varijablu tipa int, short, ili long

- za pohranu adrese variable koja je tipa short, mora se koristiti varijabla posebnog tipa i pokazivač na short
int a = 15;

int *p;

$p = \&a;$

printf("%d", *p); \Rightarrow ispisuje 15, vrijednost na koju pokazuje p

printf("%p", p); \Rightarrow ispisuje adresu objekta na koji pokazuje pokazivač $*p$, adresu varijable a

Pola i pokazivači

int X[4] = {1, 2, 3, 4};

int *p = &X[0]; // 54282

printf("%d %d %d %d", *p, *(p+1), *(p+2), *(p+3)); \Rightarrow 54282, 54286, 54290, 54294

Konštenje varijable tipa polje umjesto pokazivača na prvi element polja

- jednodimenzionalno polje: $p = \&X[0]$; isto kao: $p = x$;

- dvodimenzionalno polje: $\&X[0][0]$, ili $X[0]$ $\Rightarrow *(*p + 0 * \text{MAXSTUP} + 0) = \&X[0][0];$

- tridimenzionalno polje: $\&X[0][0][0]$, ili $X[0][0]$

- općenito vrijedi / ako je p adresa prvog člana dvodimenzionalnog polja x : $\&X[i][j] = *(*p + i * \text{MAXSTUP} + j);$

7) Funkcije formalni argumenti

tip-funkcije ime-funkcije (tip1 arg1, tip2 arg2, ...)

tjelo funkcije : definicije varijabli i mafedbe

}

Poziv funkcije

int x, y, a, b, c, d;

x = 7;

y = 4;

a = vec(x, y * 2);

✓ stvarni argumenti \Rightarrow mogu biti i izrazzi

- rezultat funkcije se može koristiti u izrazima, naravno, ako se pospremi u neku varijablu prije
- ako se pri definiciji funkcije ne navede tip funkcije, podrazumijeva se da je tip int
- void funkcija ne vraća rezultat
- ako funkcija treba vratiti rezultat, a ne obavi se odgovarajuća return mafedba, rezultat funkcije je nedefiniran
- u tijelu funkcije smije biti više return mafedbi
- ako tip podatka u mafedbi return ne odgovara tipu funkcije, automatski se obavlja pretvorba tipa
- prenose se vrijednosti ("kopije") stvarnih argumentata
- izmjenju vrijednosti formalnih argumentata ne utječe na stvarne argumente

Stog (stack) - LIFO (Last In First Out)

- dio memorije koji služi za privremeni smještaj varijabli i povratnih adresa
- pozivajući program na stog postavlja vrijednosti argumentata i povratnu adresu

Nacini prijenosa argumentata u funkciju

- call by value \Rightarrow poziv predavanjem vrijednosti argumentata (funkcija dobiva svoje vlastite kopije argumentata)
- call by reference \Rightarrow poziv predavanjem adresi argumentata (predaja pokazivača kao argumenta funkcije)

Funkcije koje trebaju vratiti više vrijednosti

- predali funkciji dvije vrijednosti pokazivača, u suprotnom se gube vrijednosti u trenutku završetka funkcije
- => kad pozvana funkcija treba direktno izmjeniti vrijednost jedne ili više varijabli iz pozivajuće funkcije, tada kao argumente treba koristiti pokazivače

Prototip/deklaracija funkcije - polje se u funkciji može "dočekati" jedino kao pokazivač

- omogućuje prevodiocu kontrolu tipa funkcije, te broja i tipa argumentata

tip-fun ime-fun (tip1 arg1, tip2 arg2, ...);

tip-fun ime-fun (tip1, tip2, ...);

=> uputa/objava prevodiocu da postoji (megđe je definisana) varijable/funkcija s navedenim imenom i tipom

- deklaracija se može pojaviti više puta u istom programu, dok se definicija smije pojaviti samo jednom

- definicijom se ujedno i deklariše (na mjestu na kojem se nalazi definicija), a deklaracijom se ne definira

- smještajni razred i mjesto definicije varijable određuju: (auto, register, static, extern)

postojanost varijable (trajnost) => podnosi programskog kôda tijekom čijeg izvršavanja sadržaj variable ostaje sačuvan

podnosi varjenju varijable (doseg) => podnosi programskog kôda unutar kojeg se varijable može referencirati ("unutar kojeg se varijable može koristiti")

Primer: funkcija koja znak po znak uspoređuje dva niza znakova s1 i s2

```
int strcmp (char *s1, char *s2) {
```

```
    while (*s1 == *s2 && *s1) {
```

```
        s1++;
```

```
        s2++;
```

```
}
```

return *s1 - *s2; => vraća razliku ASCII vrijednosti za prvi par znakova

```
}
```

u kojima se dva niza razlikuju

=> ako su nizovi jednak, funkcija vraća 0

Macro s parametrima #define ...

- u programskom kodu pretprocesor zamjenjuje macro prije prevodenja
- macro definicija se mora nalaziti u jednom retku (ili na kraj fokal stavimo \)
- ako macro koristi operator, staviti ujeli izraz unutar zagradica
- macro parametar unutar izraza uvijek staviti unutar zagradica
- macro s parametrima ne smije imati praznju imena i zagrade kojom zapocinje "lista argumentata"

typedef deklaracija

typedef postoji - tip novi tip

- deklariira sinonim: novo ime tipa s istim značenjem

null pokazivač

- umjesto nепознате vrijednosti, funkcija bi trebala vratiti muku vrijednost koju će pozivajući program moći prepoznati
- u takvima se situacijama koristi null pokazivač ("pokazivač na ništa")
- u standardnoj biblioteci potprograma definiran je macro NULL

⑧ Ugrađene funkcije

- ugrađene matematičke funkcije #include < math.h >

int abs(int x); long labs(long x); double fabs(double x);

double sin (double x); (cos, tan, asin, acos, atan, sinh, cosh, tanh)

double exp (double x); (log, log10)

double pow (double x, double y); double sqrt (double x);

double fmod (double x, double y); X Mod y

double ceil (double x); => vraća najmanji cijeli broj koji je veći ili jednak x

double floor (double x); => vraća majeći cijeli broj koji je manji ili jednak x

Ugrađene posebne funkcije #include < stdlib.h >

void exit (int status);

- exit(x) trenutno prekida izvođenje programu i pozivajućem programu (operacijskom sustavu) vraća vrijednost x => unutar main funkcije ekvivalentno s return x;

void srand (unsigned int seed);

- za isti seed, dobit će se uvijek isti niz pseudoslučajnih brojeva

int rand (void); [0, RAND_MAX] = [0, 32767]

- uzastopnim pozivanjem ove funkcije, dobiva se niz pseudoslučajnih brojeva iz zadatog intervala

- jednoliko preslikavanje cijelih brojeva x iz intervala [a,b] u interval [c,d]

$$y = (x-a) / (b-a+1) * (d-c+1) + c$$

Razliku između konstantnog znakovnog niza i niza znakova

char ime1[] = "Ivan"; => polje čiji su elementi inicijalizirani sa 'I', 'v', 'a', 'n', '\0'

(elemente ovog polja znakova je dopušteno mijenjati)

char *ime2 = "Ama"; => pokazivač koji pokazuje na konstantni znakovni niz "Ama"

(sadržaj mu nije dopušteno mijenjati)

Ugrađene funkcije za operacije nad nizovima znakova #include < string.h >

char * strcpy(char * dest, const char * src); \Rightarrow vraća dest

- kopiranje niza znakova (src u dest), uključujući 10

char * strncpy(char * dest, const char * src, size_t maxlen); \Rightarrow vraća dest

- kopiranje maxlen znakova iz niza src u niz dest (10 se možda neće uspjeti kopisati!)

- ako je maxlen veći od duljine niza koji se kopira, u dest se dodaju 10 znakova dok se ne dospije do duljine maxlen

char * strcat(char * dest, const char * src); \Rightarrow konkatenacija (nadovezivanje) nizova znakova

size_t strlen(const char * s); \Rightarrow vraća broj znakova u nizu, ne broji 10

int strcmp(const char * s1, const char * s2); \Rightarrow leksikografski uspoređuje dva niza

int strncmp(const char * s1, const char * s2, size_t maxlen); \Rightarrow leksikografski uspoređuje najviše maxlen znakova

vraća 0 ako su nizovi jednaki

vraća ujeli broj < 0 ako je s1 < s2

vraća ujeli broj > 0 ako je s1 > s2

char * strchr(const char * s, int c); \Rightarrow traženje znaka unutar niza

(vraća pokazivač na prvi znak vrijednosti c unutar niza znakova s, ako ga ne postoji, vraća NULL)

char * strstr(const char * s1, const char * s2); \Rightarrow traženje podniza s2 unutar niza s1

funkcije nad znakom #include < ctype.h >

int toupper(int ch); int tolower(int ch);

Macro nad znakom

int isdigit(int c); zamjenika 0-9

- u <ctype.h> je definiran macro #define isdigit(c) ((c) >= '0' && (c) <= '9')

int isalpha(int c); slovo (A-z ili a-z)

int isalnum(int c); slovo (A-z ili a-z) ili zamjenika (0-9)

int isprint(int c); (iscntrl, isspace, islower, isupper)

Učitavanje i ispis podataka #include <stdio.h>

int getchar(); \Rightarrow učitava jedan znak i pretvara ga u cijeli broj (ako dođe do kraja datoteke, vraća EOF)
int putchar(int ch); \Rightarrow ispisuje jedan znak (ili vraća EOF ukoliko ispis znaka nije uspio)
char * gets(char * s); \Rightarrow učitava znakovne u mrežu s dok me učita 'lm', ili znak koji odgovara kraju datoteke
int puts(const char * s); \Rightarrow ispisuje mrežu znakova s i 'lm'
int scanf(const char * format, arg1, arg2, ...); \Rightarrow učita znakovne sa tipkovnice u skladu sa zadanim formatom
int printf(const char * format, arg1, arg2, ...); \Rightarrow ispis mreži zastava u skladu sa zadanim formatom
(ako se preciznost ne zada, to je šest znamjenki iz decimalne točke)

9. Datoteke

datoteka : imenovani skup podataka na mediju za pohranu (flaka, disk, disketa, CD, memory stick)
zapis : skup susjednih podataka unutar datoteke koji se obraćaju kao cjelina
direktorij (imenik, kazalo) : datoteka koja sadrži popis i podatke o karakteristikama drugih datoteka

Tok podataka (stream)

bilo koji izvor ulaznih podataka i/ili odredište izlaznih podataka (npr. stdin - stdout)

FILE * fopen(const char * filename, const char * mode);

\Rightarrow otvara tok podataka za čitanje i/ili pisanje u datoteku filename (ako otvaranje nije uspjelo, vraća NULL)

int fclose(FILE * stream); \Rightarrow zatvara tok podataka stream

mode: način roništenja

"w" pisanje - ako datoteka ne postoji, stvara se, ako postoji, briše se sadržaj (nije dopušteno čitanje)

"a" pisanje - ako datoteka ne postoji, stvara se, ako postoji, podaci se dodaju na kraj (nije dopušteno čitanje)

"r" čitanje - ako datoteka ne postoji, vraća NULL (nije dopušteno pisanje)

"rt" čitanje i pisanje - ako datoteka ne postoji, vraća NULL

"wt" čitanje i pisanje - ako datoteka ne postoji, stvara se

"at" čitanje i pisanje - ako datoteka ne postoji, stvara se, podaci se dodaju na kraj

za čitanje i pisanje neformatičnih datoteka, treba dodati b, npr. "rb"

Podjela datoteka prema maximum upisa

- formatirane datoteke (tekstualne)

- neformatirane datoteke (binarne)

Formatirane datoteke

int fgetc(FILE * stream); \Rightarrow čita jedan znak iz toku podataka stream (EOF ako je pročitan kraj datoteke)

int fscanf(FILE * stream, const char * format, arg1, arg2, ...); \Rightarrow čita iz stream

char * fgets(char * s, int n, FILE * stream); \Rightarrow učitava n-1 znakova u niz s iz stream

(dok se ne učita 'lm' ili dospije do kraja datoteke, vraća NULL u slučaju neuspješnog učitavanja)

int fputc(int c, FILE * stream); \Rightarrow ispisuje na stream jedan znak (EOF ako nije uspješno)

int fprintf(FILE * stream, const char * format, arg1, arg2, ...); \Rightarrow ispisuje na stream u skladu sa formatom

int fputs(char * s, FILE * stream); \Rightarrow ispisuje na stream niz znakova s bez 'lm'

Nefomatirane datoteke . bin

u bajtovima

size_t fwrite(void * ptr, size_t size, size_t n, FILE * stream); \Rightarrow u tok stream zapisuje n objekata

size_t fread(void * ptr, size_t size, size_t n, FILE * stream); \Rightarrow iz streama učitava n objekata veličine size

Struktura (zapis) - složeni tip podataka čiji se elementi razlikuju po tipu

typedef struct {

int x;

struct Varijable. element = vrijednost;

int y;

vrijednost = struct Varijable. element;

} tocka;

tocka t1, t2;

pomak u bajtovima u odnosu na whence

int fseek(FILE * stream, long offset, int whence); \Rightarrow promjena trenutne pozicije u datoteci (uspješno - vraća 0)

- whence : SEEK-SET (početak datoteke), SEEK-CUR (trenutna pozicija), SEEK-END (kraj datoteke)

long ftell(FILE * stream); \Rightarrow vraća trenutnu poziciju u datoteci izraženu u broju bajtova od početka datoteke
(u slučaju pogreške vraća -1L)