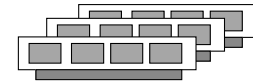


Datoteke

Memorija računala

- 1) privremena (unutarnja)
RAM (Random Access Memory)



- 2) stalna (vanjska)
 - a) sa slijednim pristupom podacima, npr.

magnetske trake



streamer trake

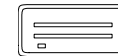


- b) s direktnim pristupom podacima, npr.

diskete

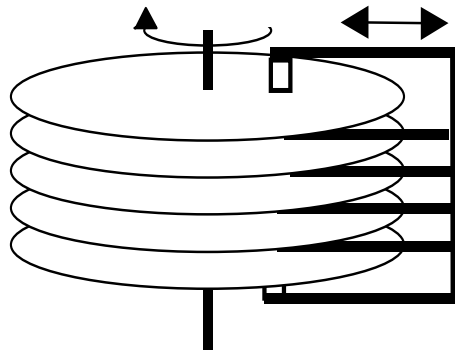


magnetski diskovi



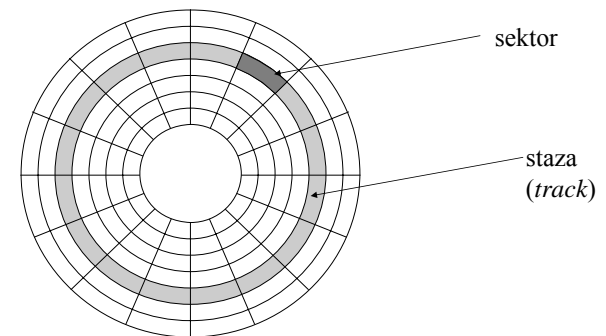
2

Shematski prikaz magnetskog diska



3

Fizička organizacija magnetskog diska



4

Logička organizacija magnetskog diska

Direktoriji
(imenici, kazala) Datoteke

c:\					
0395	atapi_cd.sys	24144	10.28.94	20:14:58	
original	cdplay.exe	34494	05.18.94	2:11:00	
update	cr520.sys	10871	02.26.95	17:30:10	
69!	eject.com	7987	01.17.91	13:23:06	
69!.ins	install.exe	18971	09.12.94	1:00:00	
access	load.exe	10700	06.28.94	11:16:16	
aw	lock.com	7987	01.17.91	13:23:28	
bin	mscdex.exe	25377	02.26.95	17:30:10	
bo	mtmcdac.sys	17351	04.18.94	1:16:00	
brink	playcd.exe	17790	01.16.92	1:30:00	
cdrom	readme.txt	6196	11.04.94	10:59:50	
corel50	unlock.com	7459	01.17.91	13:23:50	

5

Direktoriji i datoteke

- **Datoteka:** imenovani skup podataka na mediju za pohranu, obično sačinjene od zapisa.
- **Zapis:** skup susjednih podataka unutar datoteke koji se obrađuje kao cjelina.
- **Direktorij (imenik, kazalo):** datoteka koja sadrži popis i karakteristike drugih datoteka tvoreći hijerarhijsku strukturu nalik na stablo
- **Operacijski sustav računala:** program koji povezuje sklopovlje računala s programskom opremom. Između ostalog, vodi evidenciju o fizičkom smještaju direktorija i datoteka na magnetskom disku.

6

Podjela datoteka

- Po načinu pristupa
 - slijedne
 - direktne
- Po načinu upisa
 - formatirane
 - neformatirane

7

Slijedne i direktne datoteke

- Slijedne datoteke



- Direktne datoteke

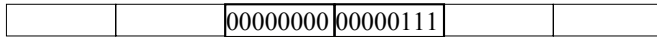


8

Neformatirane datoteke

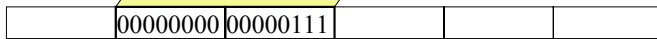
Središnja memorija

```
short int i=7;
```



```
fwrite(&i, sizeof(i), 1, d);
```

Datoteka



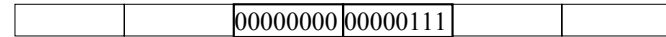
trenutna pozicija

9

Formatirane datoteke

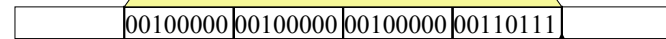
Središnja memorija

```
short int i=7;
```



```
fprintf(d, "%4d", i);
```

Datoteka



trenutna pozicija

10

Funkcija fopen

Ostvaruje vezu programa s datotekom.

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

Vraća "pokazivač na datoteku".

filename ime datoteke na disku

mode način rada

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

"a" pisanje (ako datoteka ne postoji, stvara se;
ako postoji, podatci se dodaju na kraj;
nije dozvoljeno čitanje)

11

Funkcija fopen

"r" čitanje (ako datoteka ne postoji, vraća NULL pokazivač;
nije dozvoljeno pisanje)

"r+" čitanje i pisanje (ako datoteka ne postoji,
vraća NULL pokazivač)

"w+" čitanje i pisanje (ako datoteka ne postoji, stvara se)

"a+" čitanje i pisanje (ako datoteka ne postoji, stvara se;
podatci se dodaju na kraj)

Na DOS-u za čitanje i pisanje neformatiranih datoteka treba dodati **b** npr.

"rb"

12

Funkcija `fclose`

Prekida vezu programa s datotekom.

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

vraća 0 ako je uspješno, `EOF` ako je pogreška

13

Funkcije za čitanje iz formatirane datoteke

```
int fgetc(FILE *stream);
```

vraća pročitani znak, `EOF` ako je pogreška ili kraj datoteke

```
int fscanf (FILE *stream, const char *format  
            [, address, ...]);
```

vraća broj pročitanih polja, `EOF` ako je pogreška ili kraj datoteke

```
char *fgets(char *s, int n, FILE *stream);
```

`s` područje u memoriji gdje će biti smješteni podaci

`n` maksimalni broj znakova koji se može smjestiti u `s`

vraća pokazivač na učitani niz, `NULL` ako je pogreška ili kraj datoteke

14

Primjer: Prijepis datoteke `prog.c` na zaslom a) s pomoću funkcije `fgetc`

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    FILE *d; int c;
    d = fopen ("prog.c", "r");
    if (d == NULL) {
        printf ("Datoteka se ne moze otvoriti\n");
        exit (1);
    }
    while ((c = fgetc (d)) != EOF) {
        putchar (c);
    }
    fclose (d);

    return 0;
}
```

15

Primjer: Prijepis datoteke `prog.c` na zaslom b) s pomoću funkcije `fscanf`

može se preskočiti

```
...
char linija[512+1];
...
while(fscanf(d, "%512[^\n]*%c", linija) >= 0) {
    printf ("%s\n", linija);
    ...
}
```

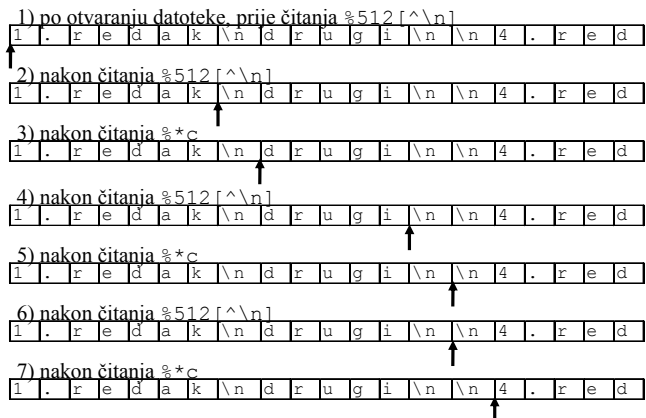
Ako je sadržaj ulazne formatirane datoteke:

```
1. redak
drugi
...
4. redak
...
```

16

Pomak pozicije u datoteci

može se preskočiti



17

Primjer: Prijepis datoteke prog.c na zaslon c) s pomoću funkcije fgets

može se preskočiti

```
#include <string.h>

...

while (fgets(linija, 512, d)) {
    /* fgets ostavlja \n, kojega treba ukloniti
    jer puts dodaje \n */
    linija[strlen(linija)-1] = '\\0';
    puts (linija);
}
```

18

Funkcije za pisanje u formatiranu datoteku

```
int fputc(int c, FILE *stream);
```

vraća ispisani znak, EOF ako je pogreška

```
int fprintf (FILE *stream,
    const char *format, [, address, ...]);
```

vraća broj ispisanih znakova (byte)

```
int fputs(char *s, FILE *stream);
```

s područje u memoriji gdje su smješteni podaci

vraća posljednji ispisani znak, EOF ako je pogreška

19

Primjer: Prijepis datoteke stara u datoteku nova a) s pomoću fgetc i fputc

```
#include <stdio.h>

int main () {
    FILE *du, *di;
    int c;

    du = fopen ("stara", "r");
    di = fopen ("nova", "w");
    while ((c = fgetc (du)) != EOF)
        fputc (c, di);
    fclose (du);  fclose (di);

    return 0;
}
```

20

Primjer: Prijepis datoteke stara u datoteku nova
b) `fscanf` i `fprintf`; c) `fgets` i `fputs`

```
b) s pomoću fscanf i fprintf
...
while (fscanf(du, "%512[^\n]*c",
             linija) >= 0) {
    fprintf(di, "%s\n", linija);
}
}
}

c) s pomoću fgets i fputs
...
while (fgets(linija, MAXLIN, du)) {
    fputs(linija, di); /* ne dodaje \n */
}
}
```

može se preskočiti

21

stdin, stdout, stderr i redirekcija

Pokretanjem programa već su otvoreni tokovi podataka ("datoteke") `stdin` (tipkovnica), `stdout` (zaslon) i `stderr` (zaslon, ali bez redirekcije), pa se prijepis iz datoteke na zaslon može načiniti kao u datoteku `stdout`.

Primjer: ispis.c

```
...
fprintf(stdout, "Na stdout\n");
fprintf(stderr, "Na stderr\n");
...
```

```
C:\>ISPIS
Na stdout
Na stderr
```

```
C:\>ISPIS > TEST.DAT
Na stderr
```

```
Datoteka test.dat
Na stdout
```

22

Čitanje neformatirane datoteke: funkcija `fread`

```
#include <stdio.h>
size_t fread(void *ptr, size_t size,
             size_t n, FILE *stream);
```

`ptr` adresa u memoriji na koju će se smjestiti učitani podaci
`size` veličina jednog objekta koji će se učitati
`n` broj objekata koji će se učitati pozivom funkcije
Vraća broj učitanih objekata.
Ako je kraj datoteke ili pogreška, rezultat je < n

23

Primjeri korištenja funkcije `fread`

```
long v; FILE *du; int n;
du = fopen("datoteka", "rb");
n = fread(&v, sizeof(v), 1, du);
...
```

```
#define MAXPOLJE 10
int p[MAXPOLJE]; FILE *du; int n;
du = fopen("datoteka", "r+b");
n = fread(p, sizeof(int), MAXPOLJE, du);
...
```

24

Pisanje u neformatiranu datoteku: funkcija `fwrite`

```
size_t fwrite(void *ptr, size_t size,
              size_t n, FILE *stream);
```

`ptr` adresa u memoriji s koje zapisuju podaci
`size` veličina jednog objekta koji će se zapisati
`n` broj objekata koji će se zapisati pozivom funkcije

Vraća broj zapisanih objekata.
Ako je bila pogreška, rezultat je $< n$

25

Primjeri korištenja funkcije `fwrite`

```
double mjer; FILE *di; size_t n;
...
di = fopen ("datoteka", "wb");
n = fwrite (&mjer, sizeof (mjer), 1, di);
....
```

```
#define MAXLEN 80
char p[MAXLEN]; FILE *di;
di = fopen ("datoteka", "w+b");
...
n = fwrite (p, sizeof (char), MAXLEN, di)
```

26

Zapisi (strukture)

- Strukture podataka čiji se elementi razlikuju po tipu.

```
struct naziv_strukture {
    tip_elementa_1 ime_elementa_1;
    tip_elementa_2 ime_elementa_2;
    ...
    tip_elementa_n ime_elementa_n;
};
```

```
struct osoba {
    char jmbg[13+1];
    char prezime[40+1];
    char ime[40+1];
    int visina;
    float tezina;
};
```

- Ovime nije definiran konkretan zapis, već je samo opisana struktura zapisa (deklaracija).

27

Definicija konkretnih zapisa

```
struct naziv_strukture zapis1, zapis2, ... , zapisN;
npr.
struct osoba o1, o2;
```

- Moguće je opisati strukturu zapisa i definirati konkretan zapis zajedno:

```
struct tocka {
    int x;
    int y;
} t1, t2, t3;
```

28

Deklaracija zapisa bez naziva

- Naziv strukture može se izostaviti ako se takva struktura drugdje ne koristi:

```
struct {  
    int dan;  
    int mjesec;  
    int godina;  
} datum;
```

29

Zapisi i typedef

- Deklaracija strukture često se koristi s `typedef`

```
typedef struct {  
    int x;  
    int y;  
} tocka;  
tocka t1, t2;
```

30

Vrijednosti elemenata zapisa

```
zapis.element = vrijednost;  
vrijednost = zapis.element;  
npr.  
strcpy (o1.jmbg, "0101970330513");  
scanf ("%s %s %d", o1.prezime, o1.ime, &o1.visina);  
o1.tezina = 75.5;  
t1.x = 7; t1.y = 2;  
t2.x = 5; t2.y = 3;  
udaljenost = sqrt(pow((double) (t1.x - t2.x), 2.) +  
                pow((double) (t1.y - t2.y), 2.))  
printf ("Datum = %02d.%02d.%d\n", datum.dan,  
        datum.mjesec, datum.godina)
```

31

Složeni zapisi

- Moguće je definiranje statičke podatkovne strukture proizvoljne složenosti jer pojedini element može također biti `struct`:

```
struct student {  
    int maticni_broj;  
    struct osoba osobni_podaci;  
    struct adresa adresa_roditelja;  
    struct adresa adresa_u_Zagrebu;  
    struct osoba otac;  
    struct osoba majka;  
};
```

32

Složeni zapisi

- Alternativno, korištenjem naredbe `typedef`:

```
typedef struct {
    char jmbg[13+1];
    char prezime[40+1];
    char ime[40+1];
    int visina;
    float tezina;
} osoba;
typedef struct {
    int maticni_broj;
    osoba osobni_podaci;
    adresa adresa_roditelja;
    adresa adresa_u_Zagrebu;
    osoba otac;
    osoba majka;
} student;
student pero;
pero.majka.tezina = 92.5;
```

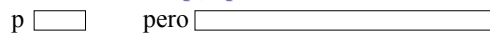
33

Pokazivač na strukturu

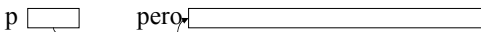
može se preskočiti

- Definicija pokazivača

```
struct osoba *p, pero;
```



```
p = &pero;
```



*p - zapis o osobi

p - adresa zapisa o osobi

- Referenciranje na element strukture preko pokazivača

```
p->prezime ili (*p).prezime
```

PrimjerZaStruct

34

Čitanje i pisanje struktura funkcijama `fread` i `fwrite`

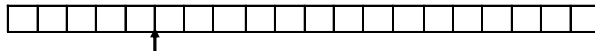
Neformatirane datoteke obično se sastoje od zapisa definiranih strukturaom.

```
n = fwrite (&o1, sizeof (o1), 1, d);
```

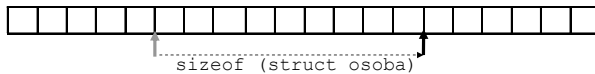
```
n = fread (&o2, sizeof (struct osoba), 1, d);
```

Čitanje i pisanje uvijek se obavlja od trenutne pozicije u datoteci.

Prije čitanja ili pisanja



Nakon čitanja ili pisanja



35

Promjena pozicije u datoteci - funkcija `fseek`

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

```
int fseek(FILE *stream, long offset,
          int whence);
```

offset pomak u byte

whence SEEK_SET - od početka

SEEK_CUR - od trenutne pozicije

SEEK_END - od kraja datoteke

Vraća 0 ako je uspjelo, <> 0 ako je pogreška.

36

Promjena pozicije u datoteci - funkcija `fseek`

Pozicioniranje na početak datoteke:

```
fseek (d, 0L, SEEK_SET);
```

Pozicioniranje na kraj datoteke:

```
fseek (d, 0L, SEEK_END);
```

Pozicioniranje na n -ti byte:

```
fseek (d, (long) n, SEEK_SET);
```

Pomak unatrag za n byte:

```
fseek (d, - (long) n, SEEK_CUR);
```

37

Trenutna pozicija u datoteci - funkcija `ftell`

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

```
long ftell(FILE *stream);
```

Vraća poziciju u datoteci ili -1 u slučaju pogreške.

Primjer: Ispisati trenutnu veličinu datoteke.

```
...
```

```
fseek (d, 0L, SEEK_END);
```

```
printf ("Velicina datoteke: %ld byte\n", ftell (d));
```

```
...
```

38

Slijedne i direktne datoteke

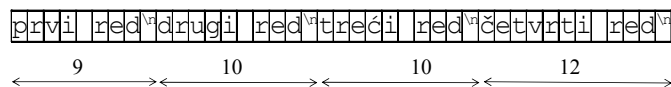
Zahvaljujući funkciji `fseek`, za razliku od nekih drugih programskih jezika svaka datoteka, ovisno o organizaciji podataka, može biti direktna.

Tekstualna datoteka načinjena editorom:

```
prvi red
drugi red
treći red
četvrti red
peti red
```

na magnetskom disku izgleda ovako:

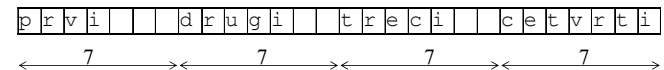
Do n -tog reda (zapisa) može se doći samo slijedno, iako postoji mehanizam pozicioniranja bilo gdje unutar datoteke.



39

Slijedne i direktne datoteke

Ako su zapisi jednake duljine, npr.



do n -tog zapisa može se doći pozicioniranjem na byte

```
(n-1)*duljina_zapisa
```

Ako je zapis u datoteci definiran strukturom naziva `zapis`, pozicioniranje na n -ti zapis načinit će se pozivom funkcije

```
fseek (d, (long) (n-1)*sizeof(struct zapis), SEEK_SET);
```

40

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

Iz slijedne formatirane datoteke "upis" treba formirati direktnu neformatiranu datoteku "tezine". U svakom zapisu datoteke "upis" nalazi se matični broj (4 znamenke), prezime i ime (40 znakova), godina rođenja (4 znamenke) i tjelesna težina (realni broj s 3 cijela mjesta i jednom decimalom). Zapis datoteke "tezine" sastoji se od matičnog broja, prezimena i imena, godine starosti i težine, pri čemu redni broj zapisa odgovara matičnom broju.

Izračunati prosječnu tjelesnu težinu.

Nakon što su svi podaci upisani treba učitavati s tipkovnice matični broj i za njega iz datoteke "tezine" godine starosti i težinu. Na zaslon ispisivati podatke za zadani matični broj uz posebnu napomenu ako osoba ima tjelesnu težinu manju od prosjeka.

Program treba završiti kad se zada neispravan ili nepostojeći matični broj.

41

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
➤SlijednaUDirektnu
#include <stdio.h>
#include <stdlib.h>
void fatal (char *poruka) {
    fputs (poruka, stderr); fputs ("\n", stderr);
    exit (1);
}
int main (void) {
    FILE *du, *di;
    struct zapis_osobe {
        short mat_br;
        char prez_ime[40+1];
        short starost;
        float tezina;
    } zapis;
```

42

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
int n, godina, god_rod, mat_br;
float prosjek;

if ((du = fopen ("upis.txt", "r")) ==NULL)
    fatal ("Ne mogu otvoriti datoteku \"upis\"");

if ((di = fopen ("tezine", "w+b")) ==NULL)
    fatal ("Ne mogu otvoriti datoteku \"tezine\"");

godina = uzmi_godinu ();
prosjek = 0; n = 0;
```

43

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
/* Ulazni podaci (datoteka "upis.txt") su oblika:
      1           2           3           4           5
12345678901234567890123456789012345678901234567890123
0001Peric Pero                               1947 76.8
0002Ivic Ivo                                  1952 86.3
0003Anic Ana                                  1968 56.7
0006Markovic Marko                            1940101.5
0012Petrovic Petar                            1972 67.8
*/
```

44

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
while (fscanf (du, "%4d%40[^\n]%4d%5f",
    &zapis.mat_br, zapis.prez_ime, &god_rod,
    &zapis.tezina) == 4) {
    zapis.starost = godina - god_rod;
    if (fseek (di, (long) (zapis.mat_br-1) *
        sizeof (zapis), SEEK_SET) != 0)
        fatal("Nije uspjelo pozicioniranje u \"tezine\");
    if (fwrite (&zapis, sizeof (zapis), 1, di) != 1)
        fatal("Nije uspjelo zapisivanje u \"tezine\");
    prosjek += zapis.tezina;
    ++n;
}
fclose (du);
```

45

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

sizeof (zapis) = 49 byte (2+41+2+4)
datoteka tezine ima velicinu 588 byte (12*49=588):

C:\NASTAVA>dir tezine

TEZINE 588 05-20-95 6:19p

0	0001	Peric Pero	48	76.8
49	0002	Ivic Ivo	43	86.3
98	0003	Anic Ana	27	56.7
147	?	?	?	?
196	?	?	?	?
245	0006	Markovic Marko	55	101.5
294	?	?	?	?
343	?	?	?	?
392	?	?	?	?
441	?	?	?	?
490	?	?	?	?
539	0012	Petrovic Petar	23	67.8

46

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
if (n > 0) {
    prosjek /= n;
    printf ("Prosjecna tezina: %5.2f\n", prosjek);
} else {
    fprintf (stderr, "Datoteka \"upis\" je prazna!\n");
    exit (1);
}
```

47

Primjer: Prijepis slijedne formatirane u direktnu neformatiranu datoteku

```
while (1) {
    printf ("\nUnesite maticni broj:");
    scanf ("%d", &mat_br);
    if (fseek (di, (long) (mat_br-1) * sizeof (zapis),
        SEEK_SET) != 0)
        fatal("Nije uspjelo pozicioniranje u \"tezine\");
    if (fread (&zapis, sizeof (zapis), 1, di) != 1 ||
        zapis.mat_br != mat_br) break;
    printf ("%04d %s %4d %5.1f\n", zapis.mat_br,
        zapis.prez_ime, zapis.starost, zapis.tezina);
    if (zapis.tezina < prosjek)
        printf ("Osoba ima tezinu manju od prosjecne!\n");
}
fclose (di);
return 0;
}
```

48

Rezultat izvođenja programa

Prosječna težina: 77.82

Unesite maticni broj:1

0001 Peric Pero 48 76.8

Osoba ima težinu manju od prosjecne!

Unesite maticni broj:2

0002 Ivic Ivo 43 86.3

Unesite maticni broj:6

0006 Markovic Marko 55 101.5

Unesite maticni broj:11

49

Dohvat tekuće godine iz sistemskog datuma a) prenosivo rješenje (DOS+Unix)

može se preskočiti

```
#include <time.h>
gdje je definiran tip podatka
typedef long time_t;

int uzmi_godinu (void) {
    time_t vrijeme;
    /* funkcija time vraća broj sekundi od 00:00:00 01. siječnja 1970 */
    vrijeme = time(NULL);
    /*
    funkcija ctime pretvara broj sekundi od 00:00:00 01. siječnja 1970 u oblik
    Fri May 19 19:05:27 1995 */
    return atoi ( ctime(&vrijeme) + 20 );
}
```

50

Rezervacija i oslobađanje memorije: funkcije malloc i free

može se preskočiti

Definicijom polja uvijek se u memoriji rezervira prostor za najveći očekivani broj članova polja npr. `int polje [1000]`; iako ćemo najčešće raditi s mnogo manjim brojem članova polja. Bolji je pristup od računala zatražiti točnu potrebnu količinu memorije.

```
#include <malloc.h>
```

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

Rezervira blok veličine `size` bajtova u memoriji računala i vraća pokazivač na taj blok. Ako blok tražene veličine nije mogao biti rezerviran, vraća `NULL` pokazivač.

```
void free (void *block);
```

Oslobađa blok memorije na koji pokazuje pokazivač `block`.

Pokazivač `block` smije biti samo jedan od pokazivača nastalih prethodnim pozivima funkcije `malloc`.

51

Kraj semestra

Predavači, ovdje završite predavanja, a preostalo slobodno vrijeme iskoristite za uvježbavanje gradiva i pripremu za završni ispit!

Studentima želimo uspješan završetak semestra, a gradivo u nastavku prezentacije može dobro poslužiti kao priprema za slijedeći semestar.

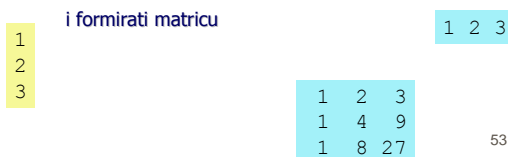
52

Primjer s rezervacijom memorije

Na disku se nalazi slijedna formatirana datoteka `polje` u kojoj se u svakom zapisu nalazi jedan cijeli broj. Učitati sadržaj datoteke u memoriju računala kao jednodimenzionalno polje. U memoriji računala formirati kvadratnu matricu kojoj su elementi prvog retka jednaki elementima učitano jednodimenzionalnog polja, a elementi ostalih redaka potencije elemenata prvog retka (drugi redak=druga potencija, treći treća itd.).

Formiranu matricu upisati na disk u slijednu neformatiranu datoteku `npolje` tako da na početku bude zapisan broj redaka odnosno stupaca matrice (`int`), a zatim elementi matrice po retcima.

Npr. datoteku `polje` treba učitati u jednodimenzionalno polje



53

Primjer s rezervacijom memorije

```
➤ Malloc
#include <stdlib.h>
#include <stdio.h>
#include <malloc.h>
#define r(i,j) r[(i)*n+(j)]
void fatal (char *poruka) {
    fputs (poruka, stderr); fputs ("\n", stderr);
    exit (1);
}
int main(void) {
    int *p, n, pom, i, j;
    long *r;
    FILE *d;
    d = fopen ("polje", "r");
    for (n = 0; fscanf(d, "%d", &pom) == 1; n++);
    fseek (d, 0L, SEEK_SET);
```

54

Primjer s rezervacijom memorije

```
p = (int *) malloc (n * sizeof (int));
if (p == NULL)
    fatal ("Nema dovoljno memorije za učitati polje");
for (n = 0; fscanf(d, "%d", &p[n]) == 1; n++);
fclose (d);
if ((r = (long *) malloc (n*n*sizeof(long))) == NULL)
    fatal ("Nema dovoljno memorije za rezultat");
for (j = 0; j < n; j++) {
    r(0,j) = p[j];
    for (i = 1; i < n; i++) {
        r(i,j) = r(i-1,j) * r(0,j);
    }
}
free (p);
```

55

Primjer s rezervacijom memorije

```
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        printf ("%10ld", r(i,j));
    }
    printf ("\n");
}
d = fopen ("npolje", "w");
fwrite (&n, sizeof (int), 1, d);
fwrite (r, sizeof (long), n*n, d);
fclose (d);
free (r);

return 0;
}
```

56

Primjer s rezervacijom memorije: rezultat izvođenja

Ulazni podaci:

3
2
4
8
5

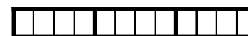
Ispis na zaslonu:

3	2	4	8	5
9	4	16	64	25
27	8	64	512	125
81	16	256	4096	625
243	32	1024	32768	3125

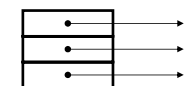
57

Polja pokazivača

Deklaracija `char *p[3];` rezervira u memoriji



Što možemo promatrati kao



Primjer: Načiniti program koji će sadržaj slijedne formatirane datoteke prepisati u drugu datoteku od posljednjeg retka prema prvom. Imena datoteka treba zadati iz komandne linije.

58

Argumenti iz komandne linije

Komandna linija npr. `prijepis ulaz izlaz` spremljena je u memoriji računala.

Funkcija `main` može imati argumente:

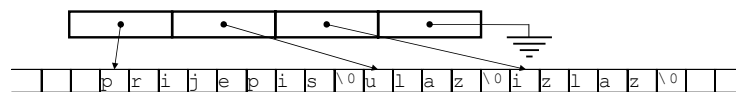
`void main (int argc, char *argv[])`

`argc` - broj argumenata u komandnoj liniji

`argv` - polje pokazivača na pojedine argumente

Za naš primjer `argc = 3`, a `argv`

`argv[0] argv[1] argv[2] argv[3]`



Na DOS-u `argv[0]` pokazuje na puno ime: `C:\prog\prijepis.exe`

59

Primjer s rezervacijom memorije

```

MallocPrijepis
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
void fatal (char *poruka) {
    fputs (poruka, stderr); fputs ("\n", stderr);
    exit (1);
}
#define MAXREDAKA 1000
#define MAXBUF 512
void main (int argc, char *argv[]) {
    FILE *d;
    char *redak[MAXREDAKA];
    char buf[MAXBUF+1];
    int i, n;
    
```

60

Primjer s rezervacijom memorije

```
if (argc != 3)
    fatal ("Poziv programa: prijepis stara nova");
if ((d = fopen (argv[1], "r")) == NULL)
    fatal ("Ne moze se otvoriti ulazna datoteka");

n = 0;
while (fgets (buf, MAXBUF, d) != NULL &&
        n < MAXREDAKA) {
    redak [n] = (char *) malloc (strlen (buf) + 1);
    if (redak[n] == NULL)
        fatal ("Nedovoljno memorije!");
    strcpy (redak[n], buf);
    n++;
}
fclose (d);
```

61

Primjer s rezervacijom memorije

```
if ((d = fopen (argv[2], "w")) == NULL)
    fatal ("Ne moze se stvoriti izlazna datoteka");

for (i = n-1; i >= 0; i--) {
    fputs (redak[i], d); free (redak[i]);
}
fclose (d);
}
```

62

Promjena rezervacije memorije: funkcija **realloc**

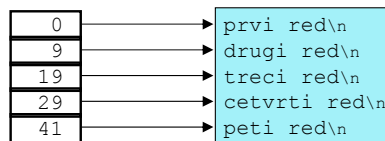
Funkcija `realloc` mijenja veličinu rezerviranog bloka u memoriji.

```
#include <malloc.h>
```

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

Ako se prije rezervirani blok može proširiti na veličinu `size`, proširuje ga, u suprotnom kopira sadržaj starog bloka na novu lokaciju na kojoj ima mjesta za `size` byte. Ako nigdje u memoriji nema `size` byte slobodnog mjesta, vraća `NULL`. Ako je `block` `NULL` pokazivač, funkcija radi kao `malloc`.

Realizacija prethodnog zadatka uz pomoć polja adresa zapisa:



63

Primjer s promjenom rezervacije memorije

```
Realloc
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <stdlib.h>
#define MAXBUF 512
int main (int argc, char *argv[]) {
    FILE *du, *di;
    long *polje;
    char buf[MAXBUF+1];
    int i, n;
    if (argc != 3)
        fatal ("Poziv programa: prijepis stara nova");
    if ((du = fopen (argv[1], "r")) == NULL)
        fatal ("Ne moze se otvoriti ulazna datoteka");
```

64

Primjer s promjenom rezervacije memorije

```
n = 0; polje = NULL;
do {
    polje = (long *) realloc (polje, (n+1)*sizeof (long));
    if (polje == NULL) fatal ("Nedovoljno memorije");
    polje[n++] = ftell (du);
} while (fgets (buf, MAXBUF, du) != NULL);
if ((di = fopen (argv[2], "w")) == NULL)
    fatal ("Ne moze se stvoriti izlazna datoteka");
for (i = n-2; i >= 0; i--) {
    fseek (du, (long) polje[i], SEEK_SET);
    fgets (buf, MAXBUF, du);
    fputs (buf, di);
}
free (polje); fclose (du); fclose (di);
return 0;
}
```

65