Универзитет “Св. Кирил и Методиј” во Скопје
Факултет за електротехника и информациски технологии

ПРОГРАМИРАЊЕ И АЛГОРИТМИ

ДАТОТЕКИ

- Програмски јазик C –

ДАТОТЕКИ

- Сместувањето на податоци во **променливите (кои се чуваат во работната меморија)** е само **привремено**.

- За **трајно чување** на големи количества податоци се користат **датотеки**.
  - Датотеките се основниот ентитет за чување на податоци на перманентните медиуми (дискови, CD/DVD-ROM).
  - Датотеките претставуваат едноставна секвенца/ низа од бајти која завршува со **ознака за крај на датотеката (*end-of-file marker*)**.

ШТО Е ДАТОТЕКА?

- ***Датотека*** е колекција на порвзани податоци кои компјутерот ги „гледа“ (третира) како една целина.
- Кога компјутерот **чита од** датотека, тој ја **копира** датотеката **од** трајниот медиум во работната меморија.
- Кога компјутерот **запишува во** датотека, тој ја **пренесува/снима** датотеката од работната меморија **во/кон** трајниот медиум.

ТЕКОВИ

- Медиумите за трајно складирање податоци вообичаено се третираат како надворешни влезно-излезни уреди.
  - за транспарентна работата со ваквите уреди за програмерот
  - што понезависна од природата на уредот
  - апстракција помеѓу самиот уред и програмата.
  - Оваа апстракцијата се вика **тек (stream)**.

- Во програмскиот јазик C, преку апстракцијата **тек**, програмите **на ист начин** се обраќаат и кон датотеките на диск, и кон тастатурата, екранот, влезно излезни порти, мрежни ресурси, печатачи, ...

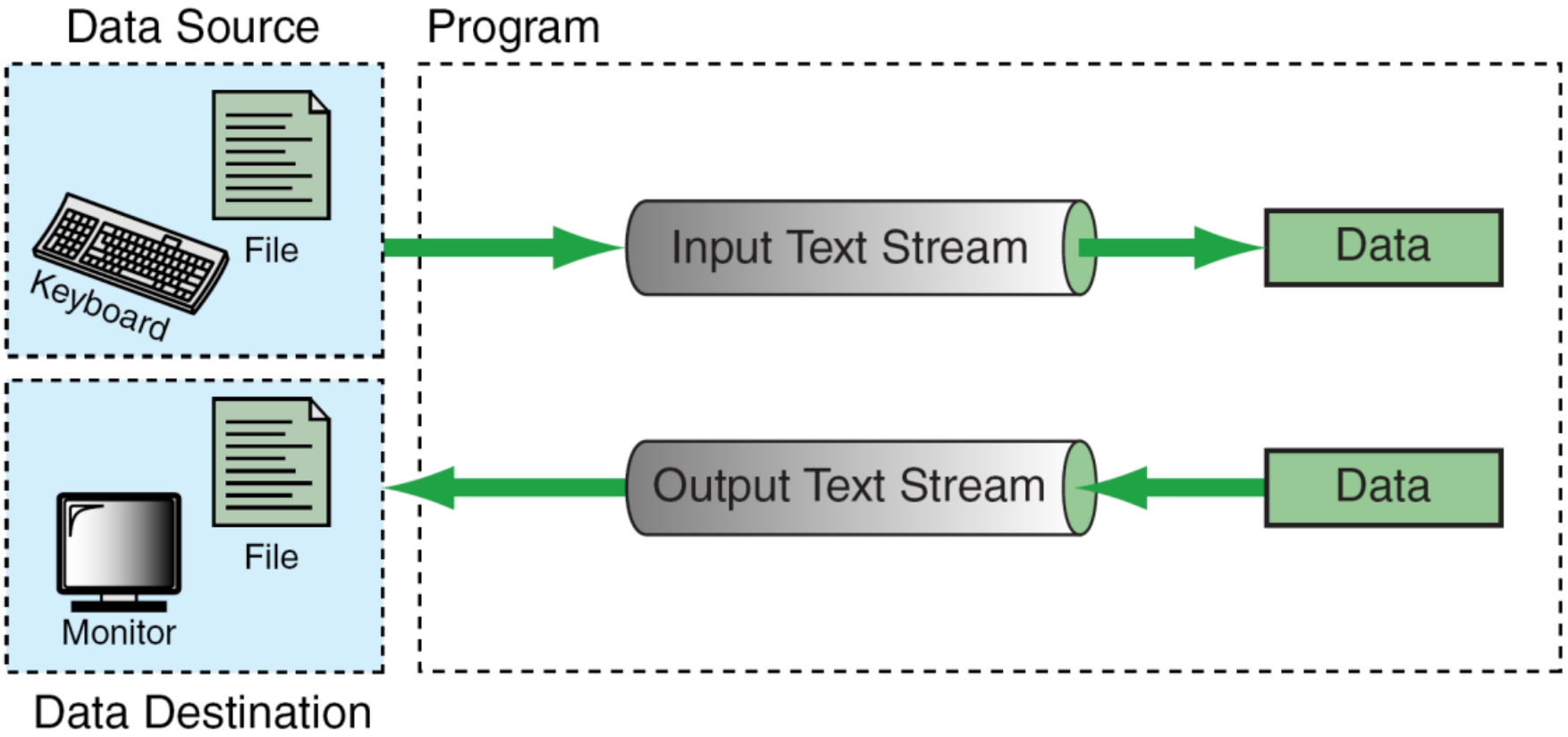
Датотеката е физички ентитет кој прима и/или праќа податоци.

ТЕКОВИ (2)

- Во секоја C програма автоматски се креираат 3 тека:
  - **stdin** – стандардниот влез асоциран кон тастатурата;
  - **stdout** - стандардниот излез асоциран кон екранот; и
  - **stderr** – стандардниот излез за грешки, повторно асоциран кон екранот.

ТЕКОВИ (3)

- во C, влез(input)/излез(output) на податоци се врши преку текови.
- Тек може да се асоцира со уред (на пр. екранот) или со датотека.



ТЕКСТУАЛНИ ТЕКОВИ & БИНАРНИ ТЕКОВИ

- С поддржува два вида датотеки
  - Text Stream Files (текстуални тек датотеки)
  - Binary Stream Files (бинарни тек датотеки)
- ***Текстуалните тековии (датотеки)*** се состојат од низа на последователни (секвенцијални) знаци поделени во редови.
  - Секој ред завршува со знакот за нов ред (`\n`).
- ***Бинарните тековии (датотеки)*** се состојат од податоци (како цели броеви, реални броеви или комплексни податочни типови-структури), со користење на „нивната мемориска претстава“.

ДАТОТЕКИ & ТЕКОВИ

- **Датотека** е **“независен ентитет”** со име доделено и зачувано во **оперативниот систем**.
- **Текот** се креира од **програмата**.
- За да се работи со датотека, **треба да се креира тек** во програмата и тој да се **асоцира со името на датотеката** снимено во ОС.
- Со **отворање на датотека** се креира тек кој се асоцира кон неа.
- Целата комуникација со датотеката понатаму се одвива преку овој тек.

ЧЕКОРИ ЗА РАБОТА СО ДАТОТЕКА

1. Се креира тек преку променлива покажувач кон структурата `FILE`:
   ```
   FILE* spData;
   ```
2. Се отвора датотеката, при што се асоцира името на текот со името на датотеката
3. Се читаат или запишуваат податоци
4. Се затвора датотеката

FILE СТРУКТУРА

- Во библиотеката на функции stdio.h постои дефинирана структура за опис на датотеки која се вика **FILE**.
  - Во неа се чуваат клучните податоци потребни за работа со датотеката како нејзиното име, почетната локација, бројот на знаци во баферот, начинот на работа и др.
  - Пред да се отпочне каква било работа со датотеката таа мора да се **отвори**.

ОТВОРАЊЕ НА ДАТОТЕКА

- Отворање на датотека се извршува со помош на функцијата **fopen( )** која ги извршува следните активности:
  - Ја поврзува физичката датотека со текот;
  - на структурата **FILE** и ги доделува **потребните информации за комуникација на програмите со датотеката**;
  - **враќа покажувач** кон локацијата на која се наоѓаат податоците за структурата.

- Сите понатамошни операции со датотеката се одвиваат преку овој **покажувач**, односно тој е **задолжителен аргумент на сите функции за работа со датотеки**.

- Покажувачот обезбедува информација со која од отворените датотеки се работи.

ОТВОРАЊЕ НА ДАТОТЕКА (2)

- За отворање на датотека потребни се информации за:
  - името на датотеката и
  - видот на датотеката (текстуална/бинарна)
  - начинот на кој ќе и се пристапува (пишување/читање/додавање...)

- **Декларирањето покажувачи** за датотеките се врши со:

```
FILE *pokf1, *pokf2, ... , *pokfn;
```

- а **нејзино отворање** со функцијата:

```
FILE *fopen(char *ime, char *nacin_na_rabota);
```

ОТВОРАЊЕ НА ДАТОТЕКА (3)

- За отворање на датотека со функцијата:

```
FILE *fopen(char *ime, char *nacin_na_rabota);
```

- Функцијата враќа покажувач кон структура од типот FILE во која ќе се чуваат податоците за отворената датотека.
- Прима два аргумента: име на датотеката која треба да се отвори и начин на (режим во) кој треба да се отвори.
- Доколку отворањето на датотеката не е успешно функцијата враќа NULL покажувач.

Пример: `pokf1 = fopen("myFile.dat", "rb");`

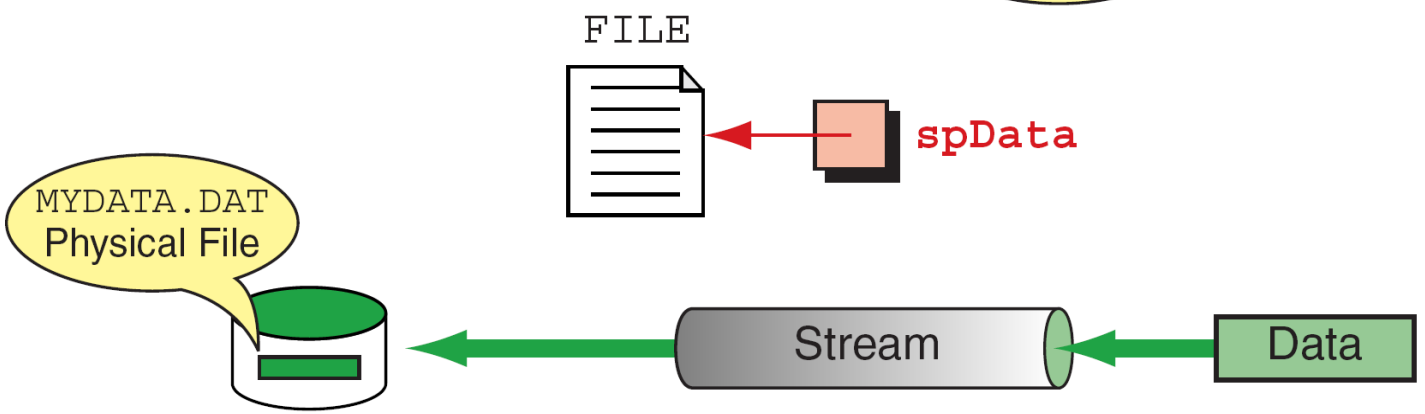
ОТВОРАЊЕ НА ДАТОТЕКА (4)

Пример: **spData = fopen("MYDATA.DAT", "w");**

```
#include <stdio.h>
…
{
int main (void)
   FILE* spData;
   …
   spData = fopen("MYDATA.DAT", "w");
   …
} // main
```

Internal File Variable

External File Name



ОТВОРАЊЕ НА ДАТОТЕКА (5)

- Отворање на датотека со функцијата:

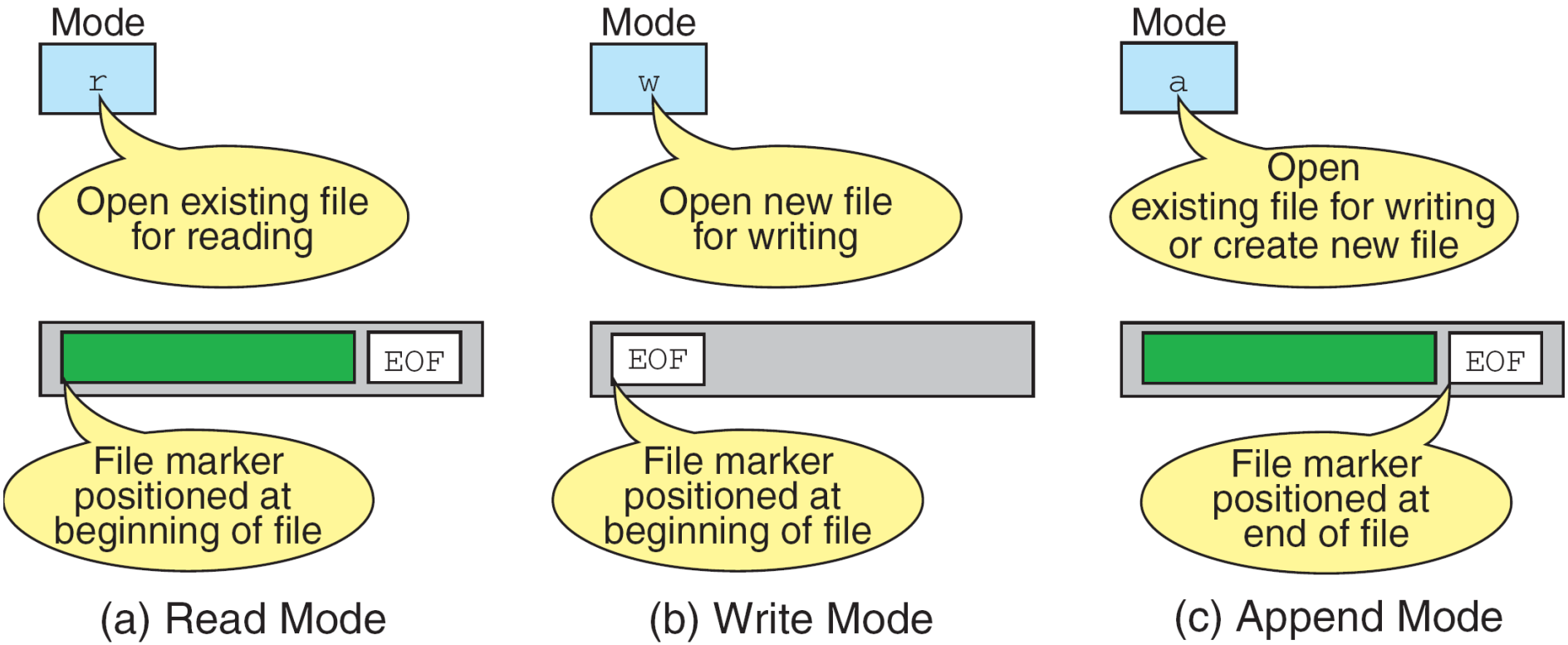
```
FILE *fopen(char *ime, char *nacin_na_rabota);
```

- Начинот на работа во најопшт случај може да биде:
  - r - читање од датотека
  - w - пишување во датотека
  - a - додавање на крајот на датотеката
- како и комбинации од некој од овие знаци и знаците b, + и t, при што:
  - b - значи работа со бинарна датотека;
  - + - значи отворање на датотеката и за читање и за пишување.

РЕЖИМИ НА ОТВОРАЊЕ ДАТОТЕКИ

Mode	Meaning
r	Open text file in read mode • If file exists, the marker is positioned at beginning. • If file doesn't exist, error returned.
w	Open text file in write mode • If file exists, it is erased. • If file doesn't exist, it is created.
a	Open text file in append mode • If file exists, the marker is positioned at end. • If file doesn't exist, it is created.

РЕЖИМИ НА ОТВОРАЊЕ ДАТОТЕКИ



ОТВОРАЊЕ НА ДАТОТЕКА

- Начините на отворање за датотека според ANSI стандардот за C се следниве:

  - "r“ отворање текстуална датотека само за читање
  - "w“ креирање текстуална датотека за запишување
  - "a“ додавање текст на крајот на датотеката
  - "rb“ отворање бинарна датотека само за читање
  - "wb“ креирање бинарна датотека за запишување
  - "ab“ додавање податоци на бинарна датотека
  - "r+“ отворање текстуална датотека за читање и запишување
  - "w+“ креирање текстуална датотека за читање и запишување
  - "a+“ отворање текстуална датотека за читање и запишување
  - "rb+“ отворање бинарна датотека за читање и запишување
  - "wb+“ креирање бинарна датотека за читање и запишување
  - "ab+" отворање бинарна датотека за читање и запишување

ПРИМЕР

```
FILE *pd;
if((pd = fopen("test","w")) == NULL)
{
    puts("Ne moze da se otvori datotekata");
    exit(-1);
}
...
```

со кој пред да се обидеме да пишуваме во, или читаме од датотеката, се проверува дали истата е успешно отворена.

Доколку датотеката се отвора за пишување, ако не постои се креира, а ако постои се пребришува.

ЧИТАЊЕ/ЗАПИШУВАЊЕ ВО ДАТОТЕКА

- За читање и пишување на единечни знаци во датотеката
  - се користат специјални влезно-излезни функции
    - fgetc( ) и fputc( )
  - стандардот ANSI ги прифаќа и алтернативните имиња
    - getc() и putc().

```
int fgetc(FILE  * fptr);
int fputc(int promenliva, FILE * fptr);
```

ЧИТАЊЕ/ЗАПИШУВАЊЕ ВО ДАТОТЕКА

- Во C се дефинирани само еден вид на датотеки – **секвенцијални**, односно **сите запишувања и читања** од датотеките се одвиваат во **секвенцијален редослед**.

- **int fgetc(FILE *pointer)**
  се чита еден знак-бајт од местото каде што покажува покажувачот **pointer**. Функцијата го враќа знакот што е прочитан. По читањето на знакот, покажувачот на датотеката се поместува за едно место надесно, на следниот знак во датотеката. Ако покажувачот е на крајот на датотеката или ако се појави грешка, тогаш се враќа EOF.

- **int fputc(int char, FILE *pointer)**
  се запишува еден знак-бајт на местото каде што покажува покажувачот **pointer**. По запишувањето на знакот, покажувачот на датотеката се поместува за едно место надесно. При успешно запишување функцијата го враќа знакот што е запишан, додека при грешка се враќа EOF.

ЗАТВОРАЊЕ ДАТОТЕКА

- По завршената работа датотеката се затвора со функцијата **fclose( )**

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

- int **fclose(**FILE ***pokazuvac_na_datoteka**) враќа:
  - **0** ако затворањето е **успешно** ;и
  - **-1** ако е **неуспешно**.

```c
#include <stdio.h>
#include <stdlib.h>
void main(){
   char niza[80] = "Test za rabota so datoteki";
   FILE *fp; char *p=niza; int i;
   /* otvaranje na datoteka za zapisuvanje */
   if((fp = fopen("datoteka", "w"))==NULL){
      printf("Ne moze da se otvori datotekata\n");
      exit(1);
   }
   while(*p){  /* zapisuvanje niza na disk */
      if(fputc(*p, fp)==EOF){
          printf("Greska pri zapisuvanje\n");
          exit(1);
      }
      p++;
   }
   fclose(fp);
}
```

Програма која отвора датотека за запишување, запишува во неа, ја затвора, ја отвора за читање, чита од неа и испишува на екран.

```c
/* otvaranje na datoteka za citanje */
if((fp = fopen("datoteka", "r"))==NULL){
  printf("Ne moze da se otvori datotekata\n");
  exit(1);
}


for(;;){ /* citanje od datotekata */
  if((i = fgetc(fp)) == EOF)
     break;
  printf(i);
}
fclose(fp);
```

Програма која отвора датотека за запишување, запишува во неа, ја затвора, ја отвора за читање, чита од неа и испишува на екран.

ФУНКЦИИ ЗА РАБОТА СО ДАТОТЕКИ

- **`fgets(pok_na_niza, max_dolz, pok_na_dat);`**
- **`fputs(pok_na_niza, pok_na_dat);`**
- **`fprintf(pok_na_dat,"kontrolna_niza", lista na promenlivi);`**
- **`fscanf(pok_na_dat,"kontrolna_niza", lista na pokazuvaci na promenlivi);`**

- fgets() чита знаци од датотеката и ги сместува во меморијата кон која покажува покажувачот.
  - Знаците се читаат од датотеката сè додека не се наиде на '\n' или EOF (не заврши датотеката) или додека не се прочитаат максимално **`max_dolz`**-1 знаци.
  - Знакот '\n' доколку е прочитан влегува во низата и зад него се додава NULL терминатор.
  - Како резултат функцијата враќа покажувач кон прочитаната низа, а доколку датотеката заврши или дојде до грешка при читањето, враќа NULL.

- За утврдување дали неуспешното читање е резултат на грешка или EOF код, се користат функциите feof( ) и ferror( ).
- `int feof(FILE *file)` - дали настапил крај на датотеката
- `int ferror(FILE *file)` - дали дошло до грешка

Пример:

Запишување координати во датотека

```
#include <stdio.h>
void vpisi(FILE *pdat)
{
  float x, y;
  char xnaslov[31], ynaslov[31];
  int i=0;
  printf("Vnesi naslov za X kolona:
");
  scanf("%30s",xnaslov);
  printf("Vnesi naslov za Y kolona:
");
  scanf("%30s",ynaslov);
  fprintf(pdat, "%30s%30s\n", xnaslov,
ynaslov);
  while(1){
    printf("Vnesi x%d, y%d : ", i, i);
    if(scanf("%f%f", &x, &y)!=2)
      break;
    fprintf(pdat,"%30.8f%30.8f\n",x,y)
;
    i++;
  }
```

Работа со аргументи на командна линија

```
int main(int argc, char *argv[ ] )
{
  FILE *pd;
  if(argc == 1)
    printf("Upotreba: %s
ime_na_datoteka\n", argv[0]);
  else
    if(argc > 2)
      printf("Premnogu argumenti");
    else
      if((pd = fopen(argv[1], "w")) ==
NULL)
     {
      printf("Ne se otvori
datotekata %s", argv[1]);
       exit(1);
     }
     else
     {
       vpisi(pd);
       fclose(pd);
     }
  return(0);
}
```

ЧИТАЊЕ/СНИМАЊЕ БИНАРНИ ПОДАТОЦИ

- Податоците што се запишуваат во датотека со fprintf се запишуваат во текстуален формат.
- Снимањето на податоци во бинарна форма заштедува простор на дискот и го забрзува преносот на податоците.
- Функции што овозможуваат читање и запишување на податоците во бинарна форма се **fread( )** и **fwrite( )**.

```
int fwrite(int *bafer, int golemina, int n,
    FILE *pokazuvac_na_datoteka)
int fread(int *bafer, int golemina, int n,
    FILE *pokazuvac_na_datoteka)
```

- каде *bafer* е покажувач на почетокот на меморискиот блок што ќе се сними во датотеката, односно во кој ќе се сместат прочитаните податоци;
- *n* е бројот на елементите во баферот;
- *golemina* е должината на секој од елементите изразена во бајти.

- И двете функции како вредност го враќаат бројот на елементи што се запишани или прочитани.
- Овој број се разликува од вредноста на n само ако настанала грешка или е достигнат крајот на датотеката.

ДИРЕКТЕН ПРИСТАП ДО ПОДАТОЦИ ВО ДАТОТЕКА

- Функции за поместување низ датотеката:

  `rewind(FILE *f)`
  - поместување на почетокот на датотеката

`int fseek(FILE *f, long offset, int pocetok)`
- релативно поместување за offset бајти во зависност од вредноста pocetok.

- Ако pocetok е:

SEEK_SET — се врши позиционирање во однос на почетокот на датотеката

SEEK_CUR — се врши позиционирање во однос на моменталната позиција во датотеката

SEEK_END — се врши позиционирање во однос на крајот на датотеката

- Следната наредба за читање/пишување ќе се изврши на новопоставената позиција.

- fseek враќа 0 ако позицијата била успешно поставена.

ДИРЕКТЕН ПРИСТАП ДО ПОДАТОЦИ ВО ДАТОТЕКА (2)

- Со функцијата

```
long ftell(FILE *pokazuvac_na_datoteka)
```

- може да се испита тековната позиција во датотеката која е претставена како број на бајти од нејзиниот почеток.

- Тековната позиција во датотеката може да се промени во почетната и со функцијата `rewind(FILE *fp)` која е идентична на `fseek(fp, 0L, SEEK_SET)`

Пример:

Да се напише програма која за дадена текстуална датотека ќе изброи колку зборови **подолги од 3 букви почнуваат и завршуваат** на **иста буква**.

- Да **не се прави разлика** меѓу **голема и мала буква**.
- Зборовите да се составени од произволен број букви, а меѓусебно да се одделени со најмалку еден специјален знак, цифра или празно место.
- Името на влезната датотека да се задава од командната линија, а доколку не е зададено се чита од стандардниот влез.

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
main(int argc, char *argv[])
{
   char prva=0,posledna=0,c;
  //brojac na zborovi, oznaka deka se naogame vo zbor, dolzina na zborot
   int b=0, iw=0, len=0;
   FILE *fin;
   if(argc>2){   /*ako ime poveke od eden parametar vo komandnata linija
ispecati kako se upotrebuva programata */
        printf("Upotreba: %s ime_na_datoteka\n",argv[0]);
        exit(-1);
   }
   if(argc==1) /* ako ne e zadaden parametar vo komandnata linija */
        fin=stdin;       /* citaj od standardniot vlez */
   /* inaku od datotekata zadadena vo komandnata linija */
   else if((fin=fopen(argv[1],"r"))==NULL) {
     fprintf(stderr,"Ne mozam da ja pronajdam datotekata %s\n", argv[1]);
      exit(0);
   }
```

```
while((c=fgetc(fin))!=EOF) { /* citaj znak po znak se do krajot */
    if(isalpha(c)) { /* ako e procitana bukva */
        if(!iw) { /* ako ne se naogas vo zbor */
            iw=1;      /* obelezi deka si vlegol vo zbor */
            prva=c;    /* ova e i prvata bukva od zborot */
        }
        len++;    /* izbroj ja stotuku procitanata bukva */
        posledna=c;   /* zapomni ja posledn procit. bukva */
    }
    else { /* ako e procitanoto ne e bukva */
        if(iw) { /* ako si se naogal vo zbor */
            iw=0;      /* obelezi deka si izlegol od zbor */
            if(len>3 && toupper(prva)==toupper(posledna))
                b++;//dokolku zborot gi zadovoluva uslovite izbroj go
            len=0;//resetiraj ja dolzinata za merenje na sledniot zbor*/
        }
    }
  } // za while ()
  printf("%d zborovi\n",b);
} // za main()
```

Domasna: da go ispecati zborot

КРАЈ