Работа с входно/изходни файлове-2
Файлов вход/изход в C

Базово програмиране
Лекция 11
Милена Карова
кат. КНТ ТУ-Варна

Функции за отваряне и затваряне на файла

- В С функциите за достъп до файлове от високо ниво са fopen, fclose, getc, putc, fgetc, fputc, ungetc, fseek, fprintf, sprintf, fscanf, sscanf, remove, fgets и fputs. Всички те се намират в stdio.h.
- Функция, чрез която се отваря файл и връща резултат указателя на файла. Също така определя допустимите операции при буфериран достъп.
- FILE *fp; fp = fopen(char *name, char *mode);Тук името на файла представлява всеки валиден за операционната система път (зададен като символен низ). Режимът (mode) е един от следните символни низове:

"r" - за четене "w" - за запис. "a" - за добавяне към съществуващ файл "r+" - за четене и запис "w+" - за създаване на файл и запис в него "a+" - за добавяне към съществуващ файл и промяна в него

fopen

- При w и w+ всичката текуща информация във съществуващ файл ще бъде изтрита!
  Променливата fp е задължително от тип FILE и трябва да бъде указател. Функцията fopen ще върне указател към файла ако резултатът е успешен или NULL (0) ако не е. Затова често функцията се използва по следния начин:
- FILE *fp; fp = fopen("file.txt", "r");
- Тук можем да използваме също if(fp==0)... или if(fp==NULL)... if (!fp) printf("Cannot open file");
- за работа с текстови файлове, към режима на работа се добавя t; например:
- rt, wt, r+t rt+ и т.н.;
- за работа с двоични файлове, към режима на работа се добавя b; например:
- rb, ab, a+b ab+ и т.н.

fclose

- fclose(FILE *fp);Функцията ще върне резултат 0 при успешно изпълнение и -1 при грешка.
- Освобождава всички заети ресурси
- Незаписаните данни в буфера се записват

getc и putc

- int getc (FILE *fp);
- Функцията връща символа от текущата позиция и увеличава указателят към текущата позиция да сочи към следващия символ от входния поток (файлът сочен от fp); при грешка при въвеждането или при достигане на край на файла, функцията връща EOF;
- пример:
- int c;
- while ((c = getc (fp))!=EOF)

Функция putc

- Прототип: int putc (int *c*, FILE *fp);
- функцията извежда символът *c* в изходния поток, т.е. във файла, сочен от fp; тя връща изведения символ или EOF при грешка;
- пример:

```
char msg[] = “Здравейте!\n”;
int i = 0;
while (msg[i])
putc (msg[i++], stdout);
```

Функциите fgetc и fputc са еквивалентни на getc и putc;

- тяхните прототипи са:
- int fgetc (FILE *fp);
- int fputc (int c, FILE *fp);

Пример за ползване на fopen,fgetc,fputc и fclose

. Съставете програма, която записва низа “Моят първи файл на С” в изходен файл “out.txt”. В програмата да се затваря файла, след това да се отваря за четене и да се извежда информацията на екрана.

Примерен вариант на решение:

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

void main()
{ char str[80]="Моят първи файл на С \n";
  FILE *fp;
  char *p, ch;
  if((fp=fopen("out.txt","w"))==NULL) // Отваря файла за запис
   {cout << endl << "Грешка при отваряне на файла";
    exit(1);}
  p = str;
  while(*p)
       {if(fputc(*p,fp) == EOF)    // Записва стринга във файла
         {cout << endl << "Грешка при запис във файла";
          exit(1);}
        p++;
       }
  fclose(fp);                          // Затваря файла
  if((fp=fopen("out.txt","r"))==NULL) // Отваря файла за четене
   {cout << endl << " Грешка при отваряне на файла";
    exit(1);}
  for( ; ; )           // Четене от файла и извеждане на екрана
   {   ch = fgetc(fp);
       if(ch == EOF ) break;
       putchar (ch);
   }
  fclose(fp);
}
```

fscanf и fprintf

- Функция fscanf
- прототип: int fscanf (FILE *fp, const char *format,);
- тази функция има еквивалентно изпълнение на функцията scanf, но чете от произволен входен поток (сочен от fp);

Още функции за работа с текстови файлове

- Функция **sscanf**;
- прототип: int sscanf (const char *ss, const char *format,);
- функцията sscanf има еквивалентно изпълнение на scanf, но чете от низ, записан в оперативната памет; използва се за преобразуване на низ от символи към числови стойности; нулевият байт в края на низа, сочен от ss се интерпретира като EOF;
- Функция **sprintf**;
- прототип: int sprintf (char *ss, const char *format,);
- функцията sprintf има еквивалентно изпълнение на printf, но записва в низ в оперативната памет; използва се за преобразуване на числови стойности към символен низ; в края на този низ автоматично се добавя нулев байт;

Още функции за работа с текстови файлове

- Функция **gets**;
- прототип: char *gets (char *str);
- тази функция се използва за прочитане на цял ред от stdin; тя преобразува символа за край на ред n в нулев байт; прочетеният ред ще се прехвърли в низа, сочен от указателя str; резултатът е указателят към прехвърления низ или NULL при грешка или прочитане на EOF;
- Функция **puts**;
- прототип: int puts (const char *str);
- тази функция извежда на stdout низа, който се сочи от str; puts преобразува символа за край на низ () в символ за нов ред (n); при успешно изпълнени puts връща неотрицателно цяло число, при грешка връща EOF;

fgets и fputs

- Функция **fgets**;
- прототип: char *fgets (char *str, int n, FILE *fp);
- Функцията чете низ от входния поток (сочен от fp) и го прехвърля в низа, сочен от указателя str; функцията спира въвеждане след прочитане на символа за нов ред, който се съхранява, или след прочитане на n-1 символа или след прочитане на EOF; и в трите случая функцията автоматично добавя нулев байт; при успешно изпълнение, функцията връща указател към прочетения низ, при достигане на EOF или при грешка по време на изпълнение, fgets връща NULL;
-
- Функция **fputs**;
- прототип: int fputs (const char *str, FILE *fp);
- Функцията fputs извежда във файл, сочен от fp, низът сочен от str; за разлика от puts, тя не добавя символ за нов ред към извеждания низ; функцията връща като резултат неотрицателно цяло число при успешно изпълнение или EOF при грешка;

Примерна програма

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 81
void main ()
{  char name[MAX], address[MAX], ch;
   int age;
   FILE *fp;
   fp = fopen (my.txt, w);
   if (fp==NULL)
   { printf (“Файлът не може да бъде отворен!\n”);
       exit (1);
   }
   printf (“Въведете данни или Ctrl+Z за край”);
   printf (“Име:” );
   while ( fgets (name, MAX-1, stdin) != NULL)
       { fputs (name, fp);

     printf (“Адрес:” );
     fgets (address, MAX-1, stdin);
     fputs (address, fp);
     printf (“Възраст:” );
     scanf (“%d”, &age);
     fflush (stdin); //ще четем низ след това
      fprintf (fp, “%d”, age);
      fputc (n, fp); fputc (n, fp);
      printf (“\nИме:” );
}
      fclose (fp);
       fp = fopen (my.txt, r);
if (fp==NULL)
{ printf (“Файлът не може да бъде отворен\!n”);
     exit (1);
}
printf (“Съдържание на файла:\ n”);
while ( fgets (name, MAX-1, fp) != NULL)
   { fputs (name, stdout);
fgets (address, MAX-1, fp);
fputs (address, stdout);
fscanf (fp,” %d”, &age);
printf (“%d”, age);
ch = fgetc (fp); putchar (ch);
ch = fgetc (fp); putchar (ch);
}
fclose (fp);
}
```

ftell и fseek

- Функция **ftell**;
- прототип: long ftell (FILE *fp);
- Функцията връща текущата позиция за достъп до данните във файла, сочен от fp; в структурата FILE има поле в което се съхранява тази текуща позиция; тя има стойност 0, когато указателят към текущата позиция сочи към началото на файла; резултатът от функцията при успешно изпълнение е текущата позиция във файла, при грешка функцията връща 1;
- Функция **fseek**;
- прототип: int fseek (FILE *fp, long offset, int whence);
- С тази функция се задава нова текуща позиция за достъп до данните във файла, сочен от fp; за целта се задава offset адресно отместване в брой байтове и whence тип на адресното отместване; той може да бъде:
  - спрямо началото на файла тип 0, може да се задава и чрез макрос SEEK_SET;
  - спрямо текущата позиция тип 1, може да се задава и чрез макрос SEEK_CUR;
  - спрямо края на файла тип 2, може да се задава и чрез макрос SEEK_END;
  - функцията връща 0 при успешно изпълнение и цяло число различно от 0 при грешка;

fflush, fread и fwrite

- Функция **fflush**;

прототип: int fflush (FILE *fp);

тази функция изчиства буферите на файла, сочен от fp; ако буферите са свързани с изходен файл или с файл за актуализация, то съдържанието им се извежда в този файл;

функция връща 0 при успешно изпълнение и EOF при грешка;

- Функция **fread** функция за въвеждане без преобразуване;

прототип: size_t fread (void *ptr, size_t size, size_t n, FILE *fp);

функцията чете от файла, сочен от fp, указаният брой блокове от данни и записва полученото в област от оперативната памет, която се сочи от указателя ptr; обикновено ptr е указател към структура; size е размер на един блок в брой байтове, n е броят на блоковете за прочитане; общият брой байтове за четене е n*size; при успешно изпълнение функцията връща броя на действително прочетените блокове (n), при грешка или достигане до края на файла връща 0 или цяло число < n;

- функция **fwrite** функция за извеждане без преобразуване;

прототип: size_t fwrite (void *ptr, size_t size, size_t n, FILE *fp);

функцията извежда във файла, сочен от fp, указаният брой блокове от данни; ptr е указател към областта от оперативната памет, където са записани данните; обикновено ptr е указател към структура; size е размер на един блок в брой байтове, n е броят на блоковете за извеждане; общият брой байтове за извеждане е n*size; при успешно изпълнение функцията връща броя на действително изведените блокове (n), при грешка или достигане до края на файла връща 0 или цяло число < n;

Примерна програма с използване на fread и fwrite

Съставете програма, която инициализира масив с реални числа, записва ги във двоичен файл, след което ги чете обратно и ги извежда на екрана на монитора.

Примерен вариант на решение:

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

double massiv[10] = {1.34, 18.67, 1009.674, 666.88, 12.8,
                     0.0, 34.86, 22.222, 1.00001, 0.7777};
void main()
  {int i;
   FILE *fp;
   if((fp = fopen("out_massiv.dat","wb")) == NULL)
    {cout << endl << "Грешка при отваряне на файла";
     exit(1);}
   for (i = 0;i < 10;i++)    // Записване във файла
    if(fwrite(&massiv[i], sizeof(double),1,fp) != 1)
     {cout << endl << "Грешка при запис" << endl;
      exit(1);}
   fclose(fp);
   if((fp = fopen("out_massiv.dat", "rb")) == NULL)
    {cout << endl << "Файлът не се отваря";
     exit(1);}
   for(i = 0; i < 10; i++)   // Нулиране на масива
    massiv[i] = 0.0;
   for(i = 0; i < 10; i++)   // Четене от файла
    if(fread(&massiv[i], sizeof(double), 1, fp)!=1)
     {cout << endl << "Грешка при четене" << endl;
      exit(1);}
   fclose(fp);
   for(i = 0; i < 10; i++)   // Извеждане на масива
    cout << endl << massiv[i];
  }
```

Функцията **fwrite()/fread()** съдържа **4** параметъра: **адреса** на променливата, който ще получи записаната/прочетената в/от файла стойност,
брой байтове, които се записват/четат в/от файла,
брой записани/прочетени „единици - обекти" и
файловия указател (**fp**).
Броят байтове се формира чрез оператора **sizeof**, който определя **размера** на променливата или на нейния тип (в **брой байтове**).

fwrite, fseek

```
for(i = 0; i < 10; i++)
  if(fwrite(&massiv[i], sizeof(double),1,fp)!=1)
   { cout << endl << "Грешка при запис" << endl;
     exit(1);}
```

```
if(fwrite(massiv, sizeof massiv,1,fp)!=1)
   { cout << endl << "Грешка при запис" << endl;
     exit(1);}
```

Възможно е данните на файла да не се четат последователно. Ако се използва **функцията fseek()**, винаги може да имате **достъп до всяка позиция във файла** (позицията във файла, в която ще се осъществи следващият достъп). Функцията оперира с **3** параметъра: файлов **указател**, **брой** байтове, които ще прескочи (стойност на **отместване**) и позицията от която ще се извърши **отместването** (вижте таблицата). Забележете, че ако търсенето се прави от SEEK_END, **отместването** трябва да бъде с **отрицателен знак**.

Начало	**Значение**
SEEK_SET	Търси от началото на файла
SEEK_CUR	Търси от текущата позиция
SEEK_END	Търси от края на файла

Функцията fseek() връща **0** при **успешно завършване** и **ненулева** стойност при **грешка**.

Текущата позиция на даден файл се определя с помощта на **функцията ftell()**. Ако се появи **грешка** се връща стойност **-1**.

Позиция във файла

Допълнителният фрагмент към програмата изглежда така:

```
// Търсене на елемент
cout <<endl << "Кой номер елемент търсите?" << endl;
cin >> pos;
// pos е променлива, дефинирана от тип long int, показваща
// номера на търсения елемент
if (fseek(fp, pos*sizeof(double), SEEK_SET))
 { cout << endl << "Грешка при търсене" << endl;
   exit(1);}
fread(&value, sizeof(double), 1, fp);
// value е променлива, дефинирана от тип double, в която се
// записва търсената стойност
cout << endl << "Търсеният елемент е: " << value;
fclose(fp);
```

Добавете в програмата **обратното** действие: търсене на стойността **22.222**. На коя **позиция във файла** е записана тя?

Вариант на добавения код изглежда така:

```
cout << endl<< "Въведи число:";
cin >> value;
for (i=0; i<10; i++)
 {if(fread(&massiv[i], sizeof(double), 1, fp) !=1 )
   {cout << endl << "Грешка при четене" << endl;
    exit(1);}
  else
   if(massiv[i] == value) break;
 }
pos=ftell(fp);
fclose(fp);
cout << endl << "Позицията е:" << pos/sizeof(double) << endl;
```

Примерна програма с използване на fread и fwrite

- Примерна програма: създава двоичен файл с компоненти структури от данни за студенти име, група, среден успех; извежда студентът с най-голям среден успех;

```
#include <stdio.h>
#include  <stdlib.h>
#include  <conio.h>
#include <ctype.h>
struct student
{ char name[60];

int group;
float av_score;
} ;
int writefile (char *);
void readfile (char *);
void main ()
{ int i; char name_file[30];
printf (“Име на файл:” );
gets (name_file);
i = writefile (name_file);
if (i!=EOF)
printf (“Файлът е създаден!\n”);
else {
printf (“Файлът не е създаден\n”);
exit (1);
}
readfile (name_file);
}
int writefile (char *name_file)
{ FILE *fp;
struct student s;
if ( (fp = fopen (name_file, wb) ) == NULL)
{ printf (Файл с име %s не може да се създаден, name_file);
exit (1);
}

do {
printf (“Име:” ); gets (s.name);
printf (“Група:” ); scanf (%d, &s.group);
printf(“ Среден успех: “ ); scanf (%f, &s.av_score);
fflush (stdin);
fwrite (&s, sizeof (s), 1, fp);
printf (“ Ще въвеждате ли още? (Y/N): “);
} while ( toupper (getche ()) == Y );
return fclose (fp);
}
void readfile (char *name_file)
{ FILE *fp;
struct student s;
int flag = 0;
long int pos;
float max_av;
if ( (fp = fopen (name_file, rb)) == NULL )
{ printf (“ Файлът не може да бъде отворен!\n “);
•       exit (1);
}
while ( fread (&s, sizeof (s), 1, fp) == 1)
{ if (!flag)

{ max_av = s.av_score; pos = 0; flag = 1; }
else if (s.av_score > max_av)
{ max_av = s.av_score;
pos = ftell (fp) sizeof (s);
}
}
fseek (fp, pos, SEEK_SET);
fread (&s, sizeof (s), 1, fp);
printf (“ %s %d %.2f\n “, s.name, s.group, s.av_score);
fclose (fp);
}
```

Информационна система

Да се състави **C/C++** програма, която:
създава двоичен файл със записи, представляващи данни за сътрудник във фирма: идентификационен номер на сътрудника, почасово плащане, брой отработени часове за една седмица, седмична заплата;
включва възможност за допълване на нови записи във файла;
включва възможност за извеждане на изчислената седмична заплата на екрана. При изчисляване на работната заплата е необходимо да се знае, че всеки извънреден час (над 40 часа седмично) се заплаща 150%, а стойността на данъците е 3.625% от общата заработка.
включва меню в главната функция main().

Примерен вариант на решение:

```
#include <fstream.h>
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

const N=30;                // Най-голям брой сътрудници
const char NAME[]="my.dat"; // Име на файл в текущата папка
struct person
 {                         char idn[6];          // Идентификационен
номер
                           float rate;           // Почасово заплащане
                           int hours;            // Отработени часове
                           double zaplata;       // Седмична заплата
 };
int n;                  // Брой сътрудници
fstream fp;             // Указател към файла

void syzdavane(struct person per[]); // Прототипи на функции
void izvejdane(struct person per[]);
void dopylvane();
void menu();

void main(void)
{menu();
}

void menu()
{struct person per[N];          // Масив от сътрудници
 int izbor;                     // За избор от менюто
 do{
                                do{                          cout <<
endl<<"********Меню********" << endl;
                                                             cout << "1.Създаване
на файл" << endl;
                                                             cout << "2.Извеждане
на данните" << endl;
                                                             cout << "3.Допълване"
<< endl;
                                                             cout << "4.Край" <<
endl;
                                                             cout << "Въведете
своя избор от 1 до 4!" << endl;
                                                             cin >> izbor;
                           }while(izbor<1||izbor>4);
                                switch(izbor){
                                                             case 1:
syzdavane(per); break;
                                                             case 2:
izvejdane(per); getch(); system("cls"); break;
                                                             case 3: dopylvane();
break;
                                                             default:
cout<<endl<<"********Край!********"<<endl;
                                                            }
                                }while(izbor!=4);
}

/* Функция за въвеждане на данните за сътрудниците и
                                записването им във файл */
void syzdavane(struct person per[])
{                               cout << endl << "Въведете броя на
сътрудниците: "<< endl;
                                cin >> n;
                                fp.open(NAME,ios::binary|ios::out); // Двоичен
файл за запис
                                if(!fp)
                                 {cout << endl << "Грешка при създаване на
файла:" << endl;
                                  exit(1);}
                                for(int i=0;i<n;i++)
                                {fflush(stdin);          // Изчистване на входния
буфер
  cout << endl << "Въведете данните за сътрудник номер "
                                << i+1 << endl;
                                 cout << endl << "Идентификационен номер:
";
                                 cin >> per[i].idn;
                                 cout << endl << "Почасовото заплащане: ";
                                 cin >> per[i].rate;
                                 cout << endl << "Отработени часове: ";
                                 cin >> per[i].hours;
                                 per[i].zaplata=per[i].rate*per[i].hours; //
Седмична заплата
                                }
 fp.write((char*)per,n*sizeof(person)); /* Запис на масива във
                                файл */
                                fp.close();
}

/* Функция за четене на данните от файл, попълване на масива и
                                извеждането им на екрана на монитора */
void izvejdane(struct person per[])
{int i=0; long pos;
 person p;              // Локална променлива за един сътрудник
cout << endl << "Пълен списък на сътрудниците: " << endl;
fp.open(NAME, ios::binary|ios::in); // Файлът се отваря за четене
if(!fp)
 {cout<< endl << "Грешка при прочитане на файла!" << endl;
  exit(1);}
fp.seekg(0l,ios::end); // Премества указателя на файла в края
pos=fp.tellg(); // Определя дължината на файла в брой байтове
fp.close();
fp.open(NAME, ios::binary|ios::in); // Указателят на файла се
                                // позиционира в началото
for(i=0; i<pos/(sizeof(person)); i++)
 {fp.read((char*)&p,sizeof(person)); // От файла се четат данните
                                // за един сътрудник
  per[i]=p;                     // Попълване на масива с данни
  cout << endl << "idn: "<< per[i].idn;
  cout << endl << "Отработено време: " << per[i].hours;
  cout << endl <<"Коефициент на заплащане :"<<per[i].rate;
  if(per[i].hours>40)
   cout<<endl<<"Отработената заплата е: "<< per[i].zaplata +
                                0.5*per[i].zaplata*(per[i].hours-40) –
(0.3625*per[i].zaplata)
                                <<endl;
  else
   cout << endl << "Отработената заплата е: " << per[i].zaplata -
                                (0.3625*per[i].zaplata)<< endl;
 }
fp.close();
}

// Функция за допълване на файла с данни за един сътрудник
void dopylvane()
{person pp; // Локална променлива с данните за един сътрудник
 long pos;
fp.open(NAME, ios::binary|ios::in); // Файлът се отваря за четене
if(!fp)
 {cout<< endl << "Грешка при прочитане на файла!" << endl;
  exit(1);}
fp.seekg(0l,ios::end); // Премества указателя на файла в края
pos=fp.tellg(); // Определя дължината на файла в брой байтове
fp.close();
fp.open(NAME,ios::binary|ios::app); // Файлът се отваря за
                                // допълване
if(!fp)
 {cout<< endl << "Грешка при отваряне на файла:" << endl;
  exit(1);}
fflush(stdin);                  // Изчистване на буфера
cout << endl<<"Въведете данните за сътрудник номер "
     << pos/(sizeof(person))+1<<endl;
cout << endl << "Идентификационен номер: ";
cin >> pp.idn;
cout << endl << "Почасово заплащане: ";
cin >> pp.rate;
cout << endl << "Отработените часове: ";
cin >> pp.hours;
pp.zaplata=pp.rate*pp.hours;
fp.write((char*)&pp,sizeof(person));
fp.close();
}
```

Информационна система

Да се състави **C/C++** програма, която:

- създава двоичен файл със записи, представляващи данни за сътрудник във фирма: идентификационен номер на сътрудника, почасово плащане, брой отработени часове за една седмица, седмична заплата;
- включва възможност за допълване на нови записи във файла;
- включва възможност за извеждане на изчислената седмична заплата на екрана. При изчисляване на работната заплата е необходимо да се знае, че всеки извънреден час (над 40 часа седмично) се заплаща 150%, а стойността на данъците е 3.625% от общата заработка.
- включва меню в главната функция main().

Примерен вариант на решение:

```
#include <fstream.h>
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

const N=30;                     // Най-голям брой сътрудници
const char NAME[]="my.dat"; // Име на файл в текущата папка
struct person
 {      char idn[6];                 // Идентификационен номер
  float rate;                   // Почасово заплащане
  int hours;                    // Отработени часове
  double zaplata;               // Седмична заплата
 };
int n;                          // Брой сътрудници
fstream fp;                     // Указател към файла

void syzdavane(struct person per[]); // Прототипи на функции
void izvejdane(struct person per[]);
void dopylvane();
void menu();

void main(void)
{menu();
}

void menu()
{struct person per[N];                  // Масив от сътрудници
 int izbor;                             // За избор от менюто
 do{
   do{ cout << endl<<"********Меню********" << endl;
       cout << "1.Създаване на файл" << endl;
       cout << "2.Извеждане на данните" << endl;
       cout << "3.Допълване" << endl;
       cout << "4.Край" << endl;
       cout << "Въведете своя избор от 1 до 4!" << endl;
       cin >> izbor;
```

Информационна система

```
 switch(izbor){
        case 1: syzdavane(per); break;
        case 2: izvejdane(per); getch(); system("cls"); break;
        case 3: dopylvane(); break;
        default: cout<<endl<<"********Край!********"<<endl;
       }
   }while(izbor!=4);
}

/*  Функция за въвеждане на данните за сътрудниците и
    записването им във файл */
void syzdavane(struct person per[])
{ cout << endl << "Въведете броя на сътрудниците: "<< endl;
  cin >> n;
  fp.open(NAME,ios::binary|ios::out); // Двоичен файл за запис
  if(!fp)
   {cout << endl << "Грешка при създаване на файла:" << endl;
    exit(1);}
  for(int i=0;i<n;i++)
  {fflush(stdin);            // Изчистване на входния буфер
   cout << endl << "Въведете данните за сътрудник номер "
           << i+1 << endl;
   cout << endl << "Идентификационен номер: ";
   cin >> per[i].idn;
   cout << endl << "Почасовото заплащане: ";
   cin >> per[i].rate;
   cout << endl << "Отработени часове: ";
   cin >> per[i].hours;
   per[i].zaplata=per[i].rate*per[i].hours; // Седмична заплата
  }
  fp.write((char*)per,n*sizeof(person)); /* Запис на масива във
                                            файл */
  fp.close();
}
```

Информационна система

```
/* Функция за четене на данните от файл, попълване на масива и
    извеждането им на екрана на монитора */
void izvejdane(struct person per[])
{int i=0; long pos;
 person p;                  // Локална променлива за един сътрудник
 cout << endl << "Пълен списък на сътрудниците: " << endl;
 fp.open(NAME, ios::binary|ios::in); // Файлът се отваря за четене
 if(!fp)
  {cout<< endl << "Грешка при прочитане на файла!" << endl;
   exit(1);}
 fp.seekg(0l,ios::end); // Премества указателя на файла в края
 pos=fp.tellg(); // Определя дължината на файла в брой байтове
 fp.close();
 fp.open(NAME, ios::binary|ios::in); // Указателят на файла се  //
позиционира в началото
 for(i=0; i<pos/(sizeof(person)); i++)
  {fp.read((char*)&p,sizeof(person)); // От файла се четат данните
                                      // за един сътрудник
   per[i]=p;                         // Попълване на масива с данни
   cout << endl << "idn: "<< per[i].idn;
   cout << endl << "Отработено време: " << per[i].hours;
   cout << endl <<"Коефициент на заплащане :"<<per[i].rate;
   if(per[i].hours>40)
    cout<<endl<<"Отработената  заплата  е:  "<<  per[i].zaplata  +
    0.5*per[i].zaplata*(per[i].hours-40) - (0.3625*per[i].zaplata)
      <<endl;
   else
    cout << endl << "Отработената заплата е: " << per[i].zaplata -
                        (0.3625*per[i].zaplata)<< endl;
  }
 fp.close();
}
```

Информационна система

```
// Функция за допълване на файла с данни за един сътрудник
void dopylvane()
{person pp; // Локална променлива с данните за един сътрудник
 long pos;
 fp.open(NAME, ios::binary|ios::in); // Файлът се отваря за четене
 if(!fp)
  {cout<< endl << "Грешка при прочитане на файла!" << endl;
   exit(1);}
 fp.seekg(0l,ios::end); // Премества указателя на файла в края
 pos=fp.tellg(); // Определя дължината на файла в брой байтове
 fp.close();
 fp.open(NAME,ios::binary|ios::app); // Файлът се отваря за     //
допълване
 if(!fp)
  {cout<< endl << "Грешка при отваряне на файла:" << endl;
   exit(1);}
 fflush(stdin);                          // Изчистване на буфера
 cout << endl<<"Въведете данните за сътрудник номер "
      << pos/(sizeof(person))+1<<endl;
 cout << endl << "Идентификационен номер: ";
 cin >> pp.idn;
 cout << endl << "Почасово заплащане: ";
 cin >> pp.rate;
 cout << endl << "Отработените часове: ";
 cin >> pp.hours;
 pp.zaplata=pp.rate*pp.hours;
 fp.write((char*)&pp,sizeof(person));
 fp.close();
}
```