Laboratory lesson №1
(The spring session)

Theoretical information

The early versions of programming languages contained only simple built-in data types - integer, real, logical, etc. (the so-called simple or atomic types). This data could be organized into arrays. It was enough for computational tasks, however, as they developed, computers began to be used for text information and graphic images processing, for database creating and maintenance, etc.

Thus, the variety of information processed has led to the need for the programmer to create his own composite data types - structures. Structures allow you to combine different data types - numerical data, arrays, strings, the structures themselves, etc. Arrays can be formed from structures.

A structure is a composite data type built using other types. The structure consists of fields. **Fields** (structure elements) - variables or arrays of the standard type (int, char, etc.) or other structures previously described. The structure is declared using the keyword ***struct,*** followed by its name and then a list of elements enclosed in braces. For example, a structure describing ***date*** can be written as follows:

```
struct date {
 int day;
 int month;
 int year;
};
```

The syntax for declaring structure variables is the same as for variables of other types. For instance,

```
date days;
```

To access the fields of the structure, we use an operator .(point). For example, to write the date of November 20, 2011 to the days variable, you should use this approach:

```
days.day = 20;
days.month = 11;
days.year = 2019;
```

or it is possible to set the initial values when declaring a variable:

```
date days = {20, 11, 2019};
```

The creation of structures' array has the same syntax as an array of atomic type elements:

```
date days_array[20]; // Масив типу date з 20 елементів
date* days_dyn_array = new date[20]; // Динамічний масив
```

The work with a pointer to a structure has some features: formally, in order to access the field of a structure through a pointer, the pointer must be dereferenced using the * operation, and then use the operator. (point). For instance:

```
date days;
date* pdays = &days; // Покажчик на структуру
(*pdays).day = 6;
```

Accessing the structure field by using the dereference operator (*) and the point can be simplified by using the -> operator (arrow, it consists of a minus sign and more). The arrow operator includes dereferencing operations and a point. Thus, the last example can be converted:

```
date days;
date* pdays = &days; // Покажчик на структуру
pdays->day = 6;
```

The functions of ifstream.h library for working with binary files

The fstream.h library contains two functions to work with binary files: read and write:

```
ostream& write( const char * s, streamsize n );
istream& read( char * s, streamsize n );
```

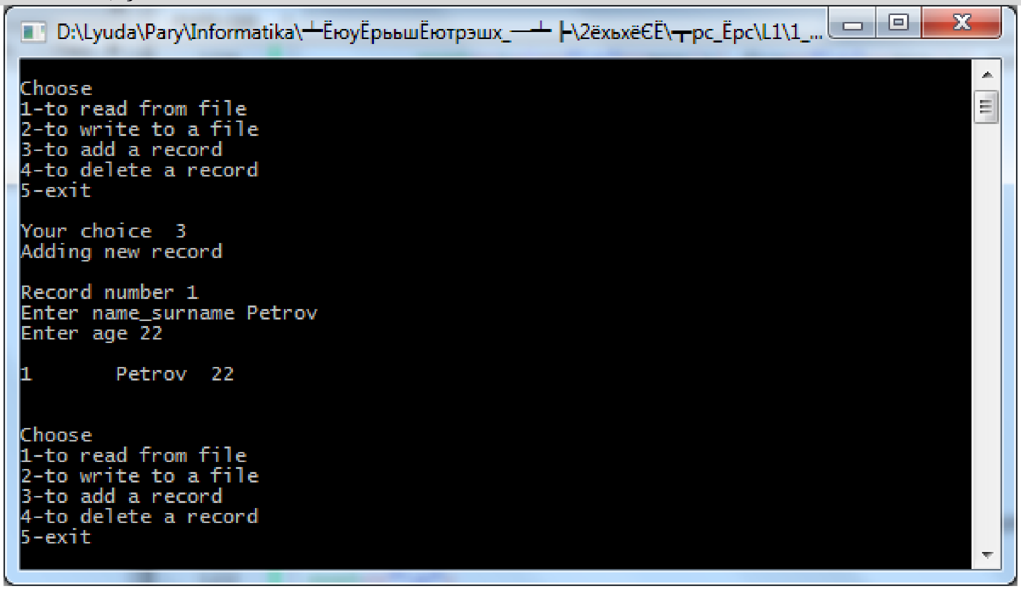
The first parameter is ***char*s*** is a pointer to the data array, which needs to be written/read from a file. Another parameter ***streamsize n*** is the size of the array in bytes, which must be read/written.

It should also be noted that for the correct operation of these functions, you should open the file in **ios::binary mode**.

Example, write a program (fig. 1) in which:
1. The user has the ability to read and write information from a binary file:
1. Name of employee
2. year of birth
2. The user has the ability to add, delete and modify information.

```
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

const int maxlen=255;
#pragma pack(push,1) // compiler directive to align the size of structure fields
struct sworker
{
    char fio[maxlen];
    int age;
};
#pragma pack (pop)  //compiler directive to align the size of structure fields

sworker arr[maxlen];
int worker_index=0;

int menu();
void readFromFile(const char *fileName);
void saveToFile (const char* fileName);
void addNew();
void del ();

int main()
{

     while  (1)// creating an endless cycle for  the selection menu
    {
        switch(menu())
        {
            case 1:readFromFile("file.dat"); break;
            case 2:saveToFile("file.dat"); break;
            case 3:addNew();break;
            case 4: del(); break;
            case 5: return 0;
            default: cout<<"Incorrect choice"<<endl;
        }
    }
}
int menu()//function shows menu items
{
    cout<<"\n";
    int ans;
    cout<<"Choose\n";
    cout<<"1-to read from file\n";
    cout<<"2-to write to a file\n";
    cout<<"3-to add a record\n";
    cout<<"4-to delete a record\n";
    cout<<"5-exit\n";
    cout<<"\n";
    cout<<"Your choice  ";
    cin>>ans;
    return ans;
}
void saveToFile (const char* fileName)// function to write data to a file
{
    ofstream f;
    f.open(fileName, ios::binary);
    f.write((char*)arr, sizeof (sworker)*worker_index);
    f.close();
    cout<<"The data was saved in a file\n";
}
void readFromFile(const char *fileName) //function to read data from file
{
  ifstream f;
  f.open(fileName, ios::binary);
  if (!f)
  {
      cout<<"The file does not exists";
  }
        else{
  sworker worker;
  worker_index=0;
  while (1)
  {
      f.read((char*)&worker, sizeof(worker));
      if (f.eof())//until end of file
          break;
      arr[worker_index]=worker;
      worker_index++;
  }
  f.close();
  cout<<"The data was read from file\n";
  for (int i=0; i<worker_index; i++)
    {
       cout<<i+1<<"\t"<<arr[i].fio<<"\t"<<arr[i].age<<endl;
    }
}}
void addNew()//function to add new record to structure
{
cout<<"Adding new record\n\n";
cout<<"Record number "<<worker_index+1<<"\n";
cin.ignore();
cout<<"Enter name_surname ";
cin.getline(arr[worker_index].fio,maxlen);
cout<<"Enter age ";
cin>>arr[worker_index].age;
worker_index++;
cout<<"\n";
for (int i=0; i<worker_index; i++)
    {
        cout<<i+1<<"\t"<<arr[i].fio<<"\t"<<arr[i].age<<endl;
    }
cout<<"\n";
```

```
}
void del ()  //function to delete record
{int d;
    cout<<"Choose record number to delete ";
    cin>>d;
    for (int i=d-1;i<worker_index;i++)
    {arr[i]=arr[i+1];}
    worker_index--;
    cout<<"\n";
for (int i=0; i<worker_index; i++)
    {
        cout<<i+1<<"\t"<<arr[i].fio<<"\t"<<arr[i].age<<endl;
    }
cout<<"\n";}
```


D:\Lyuda\Pary\Informatika\┴ЁюуЁрььшЁютрэшх_—┴ ├\2ёхьхёЄЁ\┬pc_Ёрс\L1\1_...

```
Choose
1-to read from file
2-to write to a file
3-to add a record
4-to delete a record
5-exit

Your choice  3
Adding new record

Record number 1
Enter name_surname Petrov
Enter age 22

1       Petrov  22


Choose
1-to read from file
2-to write to a file
3-to add a record
4-to delete a record
5-exit
```

Fig.1

Laboratory work №1

Var. №1.
Write a program in which
1. The user can write and read from the binary file employee information: a) name, b) last name, c) salary, d) changes.
2. The user has the opportunity to add, delete, modify and sort information by item 1a).
3. For all the actions described above, specially created functions are used.
4. After the operation is completed, information about it is logged (both on screen and in file) so that the user can see the entire history of operations

Var. №2.
Write a program in which
1. The user can write and read from the binary file train information: a) number, b) departure time, c) arrival time, d) destination.
2. The user has the opportunity to add, delete, modify and sort information by item 1a).
3. For all the actions described above, specially created functions are used.
4. After the operation is completed, information about it is logged (both on screen and in file) so that the user can see the entire history of operations

Var. №3.
Write a program in which
1. The user can write and read from the binary file product information: a) name, b) price, c)quantity, d) delivery period.
2. The user has the opportunity to add, delete, modify and sort information by item 1a).
3. For all the actions described above, specially created functions are used.
4. After the operation is completed, information about it is logged (both on screen and in file) so that the user can see the entire history of operations