

## Лабораторна робота №10

**Тема:** Робота з файлами.

**Мета:** навчитись зчитувати і записувати данні у текстові і двійкові файли за допомогою файлових потоків.

Завдання 1: Написати програму, яка записуватиме ваші дані у текстовий файл «myfile.txt» (П.І.П., вік, номер телефону, дата народження, місце навчання). Дані для запису можна або зчитувати з консолі при виконанні програми або прописати у кодї програми.

Завдання 2: Написати програму, яка здійснить читання файла «myfile.txt» у відповідні локальні змінні і виведе їх значення на екран

Код програми :

```
#include <iostream>
#include<fstream>
#include <string>
class Data
{
private:
    struct Inizials
    {
        std::string m_name;
        std::string m_familyname;
        std::string m_fathername;
    };
    struct DateOfBirth
    {
        int m_day;
        int m_mounth;
        int m_year;
```

```

};
Inizials m_person;
std::string m_phoneNumber;
DateOfBirth m_birth;
std::string m_placeOfBirth;

public:
    Data ();

    Data (std::string name, std::string familyName, std::string fatherName, int
day, int mounth, int year, int phoneNumber, std::string dateOfBirth);

    void setPerson ();

    void setPerson(std::string name, std::string familyName, std::string
fatherName);

    void setPhoneNum ();

    void setPhoneNum (int phone);

    void setBirth ();

    void setBirth (int day, int mounth, int year);

    void setPlaceOfBirth(std::string placeOfBirth);

    void setPlaceOfBirth ();

    friend std::istream &operator>> (std::istream &in, Data &data);

    void getPerson ();

    std::string getPhoneNumber ();

    void getBirth ();

    void Print ();

    std::string getPlaceOfBirth ();

```

```
};
```

```
Data::Data ()
```

```
{
```

```
}
```

```
Data::Data (std::string name, std::string familyName, std::string fatherName, int  
day, int mounth, int year, int phoneNumber, std::string placeOfBirth)
```

```
{
```

```
    setPerson (name, familyName, fatherName);
```

```
    setBirth (day, mounth, year);
```

```
    setPhoneNum (phoneNumber);
```

```
    setPlaceOfBirth(placeOfBirth);
```

```
}
```

```
void Data::setPerson ()
```

```
{
```

```
    std::cout << "Enter name: ";
```

```
    std::cin >> m_person.m_name;
```

```
    std::cout << "Enter familyname: ";
```

```
    std::cin >> m_person.m_familyname;
```

```
    std::cout << "Enter fathername: ";
```

```
    std::cin >> m_person.m_fathername;
```

```
}
```

```
void Data::setPerson (std::string name, std::string familyName, std::string
fatherName)
```

```
{
    m_person.m_name = name;
    m_person.m_familyname = familyName;
    m_person.m_fathername = fatherName;
}
```

```
void Data::setPhoneNum ()
```

```
{
    std::cout << "Enter phone: ";
    std::cin >> m_phoneNumber;
}
```

```
void Data::setPhoneNum (int phone)
```

```
{
    m_phoneNumber = phone;
}
```

```
void Data::setBirth ()
```

```
{
    std::cout << "Enter day of birth: ";
    std::cin >> m_birth.m_day;
    std::cout << "Enter mounth of birth: ";
    std::cin >> m_birth.m_mounth;
    std::cout << "Enter year of birth: ";
    std::cin >> m_birth.m_year;
}
```

```
void Data::setBirth (int day, int mounth, int year)
```

```
{  
    m_birth.m_day = day;  
    m_birth.m_mounth = mounth;  
    m_birth.m_year = year;  
}
```

```
void Data::setPlaceOfBirth (std::string placeOfBirth)
```

```
{  
    m_placeOfBirth = placeOfBirth;  
}
```

```
void Data::setPlaceOfBirth ()
```

```
{  
    std::cout << "Enter place of birth: ";  
    std::cin >> m_placeOfBirth;  
}
```

```
void Data::getPerson ()
```

```
{  
    std::cout << "Name: " << m_person.m_name << std::endl;  
    std::cout << "FamilyName: " << m_person.m_familyname << std::endl;  
    std::cout << "FathersName: " << m_person.m_fathername << std::endl;  
}
```

```
std::string Data::getPhoneNumber ()
```

```
{
```

```

        return m_phoneNumber;
    }

void Data::getBirth ()
{
    std::cout << "Birthday: " << m_birth.m_day << '.' << m_birth.m_mounth <<
    '.' << m_birth.m_year;
}

std::string Data::getPlaceOfBirth ()
{
    return m_placeOfBirth;
}

void Data::Print ()
{
    getPerson ();
    std::cout << "Phone number: " << getPhoneNumber ();
    getBirth ();
    std::cout << "Place of birth " << getPlaceOfBirth ();
}

std::istream &operator>>(std::istream &in, Data &data)
{
    std::cout << "Enter name: ";
    in >> data.m_person.m_name;
    std::cout << "Enter familyname: ";
    in >> data.m_person.m_familyname;
    std::cout << "Enter fathersname: ";

```

```

in >> data.m_person.m_fathername;
std::cout << "Enter phone: ";
in >> data.m_phoneNumber;
std::cout << "Enter day of birth: ";
in >> data.m_birth.m_day;
std::cout << "Enter mounth of birth: ";
in >> data.m_birth.m_mounth;
std::cout << "Enter year of birth: ";
in >> data.m_birth.m_year;
std::cout << "enter place of birth: ";
in >> data.m_placeOfBirth;
return in;
}

```

```

int main()
{

}

```

Результат виконання програми:

```

Data d;
std::fstream fout;
std::fstream readf;
bool addf = true;
char n;
std::string path = "myfile.txt";
fout.open (path, std::ios::out | std::ios::binary);
if( !fout ) {
    std::cout << "Can't open file" << std::endl;
}

```

```

    return 1;

}

else {
    do {
        std::cin >> d;
        fout.write ((char *)&d, sizeof (Data));
        do {
            std::cout << "Do you want to enter next one? y/n";
            std::cin >> n;
        } while( n != 'y' && n != 'n' );

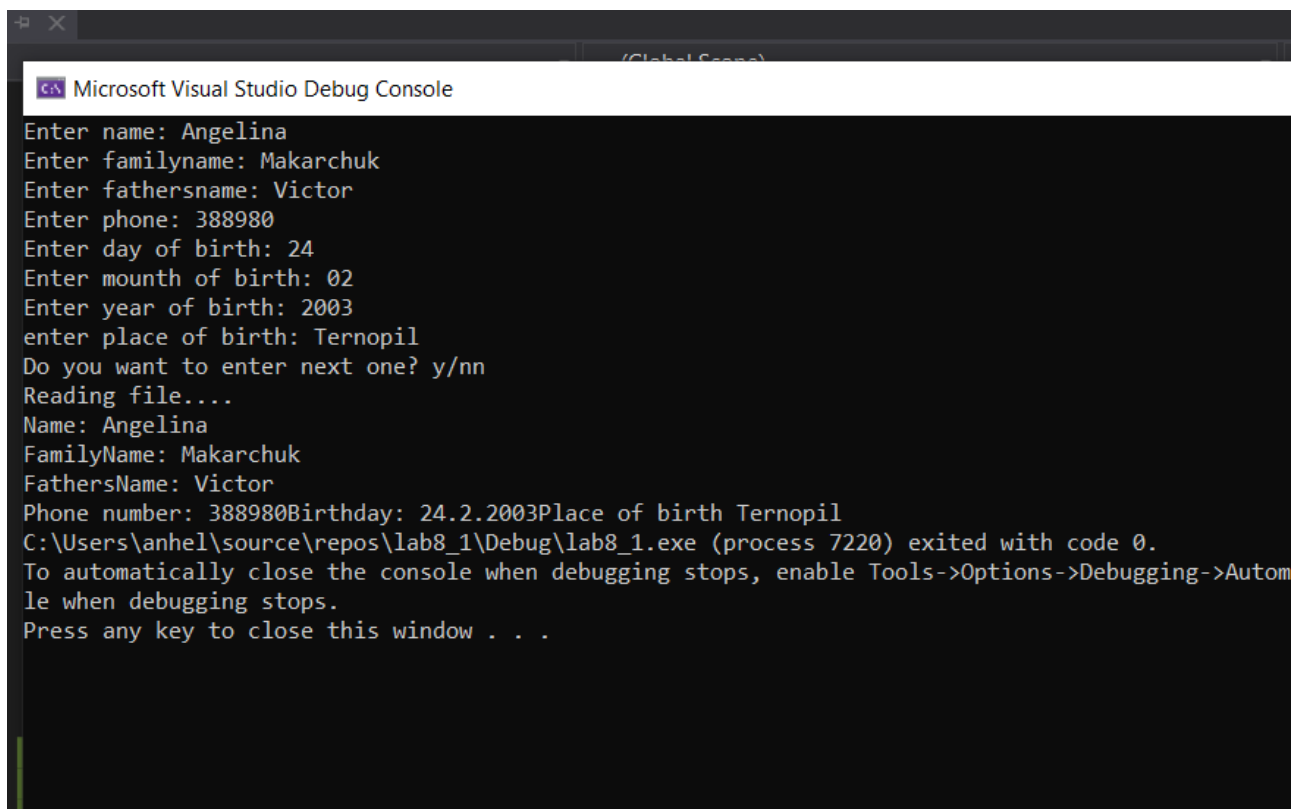
        if( n == 'n' ) addf = false;
    } while( addf );
}

fout.close ();

std::cout << "Reading file.... " << std::endl;
readf.open (path, std::fstream::in | std::fstream::out);
while( readf.read ((char *)&d, sizeof (Data)) ) {
    d.Print ();
}

```





```
Microsoft Visual Studio Debug Console
Enter name: Angelina
Enter familyname: Makarchuk
Enter fathersname: Victor
Enter phone: 388980
Enter day of birth: 24
Enter mounth of birth: 02
Enter year of birth: 2003
enter place of birth: Ternopil
Do you want to enter next one? y/nn
Reading file....
Name: Angelina
FamilyName: Makarchuk
FathersName: Victor
Phone number: 388980Birthday: 24.2.2003Place of birth Ternopil
C:\Users\anhel\source\repos\lab8_1\Debug\lab8_1.exe (process 7220) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Autom
le when debugging stops.
Press any key to close this window . . .
```

Завдання 3: Створити клас Person, який міститиме атрибутами ті дані, що записувались у завданні №2. Дані повинні зчитуватись із клавіатури при створенні об'єкта. Створити об'єкт ME і записати його у файл. (Перевизначити оператор запису у потік, який запише дані у файл).

Завдання 4: Використовуючи клас Person зчитайте дані, що були записані у попередньому завданні. (Потрібно перевизначити оператор зчитування з потоку, який зчитуватиме дані із файлу).

Код програми:

```
#include<iostream>

#include<fstream>

template <typename T>

class Person
{
private:
```

```

template<typename T>
class Node // шаблонний клас нод однозв'язний список
{
public:
    Node *pNext;
    T info;
    Node (T info = T (), Node *pNext = nullptr)
    {
        this->info = info;
        this->pNext = pNext;
    }
};
Node<T> *m_model;
int m_power;

public:
    Person ();
    Person (T *m_model, int m_power);
    Person (const Person &obj);
    void setModel (const T *m_model, int lenght); //додає масив в кінець
    T *getModel ();
    void setPower (int); //присвоює потужність
    int getPower (); //виводить потужність
    void print (); //виводить всю інформацію
    void input (); // ввід об'єкта
    void pop_front (); // видаляє голову
    void pushBack (T); // додає елемент в кінець

```

```
T getModel (int index); // виводить одну модель
```

```
friend std::ostream operator <<(std::ostream &out, Person<T> &p);
```

```
~Person ();
```

```
};
```

```
template<typename T>
```

```
Person<T>::Person () :m_power (0), m_model (nullptr) {}
```

```
template<typename T>
```

```
Person<T>::Person (T *model, int power)
```

```
{
```

```
    setModel (model, power);
```

```
}
```

```
template<typename T>
```

```
Person<T>::Person (const Person &obj) { m_model = obj.m_model; m_power  
= obj.m_power; }
```

```
template<typename T>
```

```
void Person<T>::setModel (const T *model, int lenght)
```

```
{
```

```
    for( int i = 0; i < lenght; i++ ) {
```

```
        pushBack (model[i]);
```

```
    }
```

```
}
```

```

template<typename T>
T *Person<T>::getModel ()
{
    return *m_model.info;
}

template<typename T>
void Person<T>::setPower (int a)
{
    m_power = a;
}

```

```

template<typename T>
int Person<T>::getPower ()
{
    return m_power;
}

```

```

template<typename T>
void Person<T>::print ()
{

```

```

    Node<T> *curr = this->m_model;
    std::cout << "\nAll elements:";
    while( curr != nullptr ) {
        std::cout << curr->info << "\t";
        curr = curr->pNext;
    }
    std::cout << std::endl;

```

```

        std::cout << "Power is" << m_power << std::endl;
    }
template<typename T>
void Person<T>::input ()
{
    while( m_power )
        pop_front ();
    int power;
    std::cout << "Enter number of models:";
    std::cin >> power;
    int counter = 1;
    T curr;
    while( counter <= power ) {
        std::cout << "Enter element";
        std::cin >> curr;
        pushBack (curr);
        counter++;
    }
}

template<typename T>
void Person<T>::pop_front ()
{
    Node<T> *temp = m_model;
    m_model = m_model->pNext;
    delete temp;
    m_power--;
}

```

```

template<typename T>
void Person<T>::pushBack (T element)
{
    if( m_model == nullptr ) {
        m_model = new Node<T> (element);
    }
    else {
        Node<T> *curr = this->m_model;
        while( curr->pNext != nullptr ) {
            curr = curr->pNext;
        }
        curr->pNext = new Node<T> (element);
    }
    m_power++;
}

template<typename T>
T Person<T>::getModel (int index)
{
    int counter = 0;
    Node<T> *curr = this->m_model;
    while( curr != nullptr ) {
        if( counter == index ) {
            return curr->info;
        }
        curr = curr->pNext;
        counter++;
    }
}

```

```

}
template<typename T>
Person<T>::~~Person ()
{
    while( m_model != nullptr ) {
        pop_front ();
    }
}

int main ()
{
    Person<char> d;
    std::fstream fout;
    std::fstream readf;
    bool addf = true;
    char n;
    std::string path = "myfile.txt";
    fout.open (path, std::ios::out | std::ios::binary);
    if( !fout ) {
        std::cout << "Can't open file" << std::endl;
        return 1;
    }
    else {
        do {
            d.input();
            fout.write ((char *)&d, sizeof (Person<char>));
            do{

```

```

        std::cout << "Do you want to enter next one? y/n";

        std::cin >> n;

    } while( n != 'y' && n != 'n' );

    if( n == 'n' ) addf = false;

} while( addf );

}

fout.close ();

std::cout << "Reading file.... " << std::endl;

readf.open (path, std::fstream::in | std::fstream::out);

while( readf.read ((char *)&d, sizeof (Person<char>)) ) {

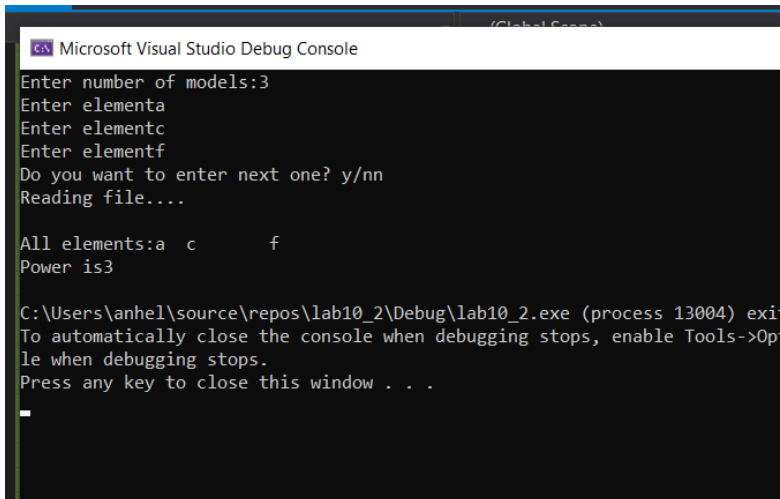
    d.print ();

}

}

```

Результат виконання програми:



```

Microsoft Visual Studio Debug Console
(Global Scope)

Enter number of models:3
Enter elementa
Enter elementc
Enter elementf
Do you want to enter next one? y/n
Reading file....

All elements:a c f
Power is3

C:\Users\anhel\source\repos\lab10_2\Debug\lab10_2.exe (process 13004) exited
To automatically close the console when debugging stops, enable Tools->Options->
Press any key to close this window . . .

```

**Висновок:** Я навчилась зчитувати і записувати данні у текстові і двійкові файли за допомогою файлових потоків