Нижегородский государственный технический университет

им. Р.Е. Алексеева

Институт радиоэлектроники и информационных технологий

Лабораторная работа №2

Тема: «База данных успеваемости учеников»

Выполнила: студентка 1-го курса  
кафедры «Прикладная математика»  
Петрухина М.С

Преподаватель: к. ф-м. н. Чернов А.Г.

1. Цель………………………………………………………………2
2. Документация…...…………………………………………………
   1. Файл main.cpp………………………………………………..3
   2. Файл list.cpp………………………………………………….5
   3. Файл pupil.cpp………………………………………………..6
   4. Файл list.h…………………………………………………….7
   5. Файл pupil.h…………………………………………………..8
   6. Класс List……………………………………………………..9
   7. Класс Pupil……………………………………………………15
3. Скриншот результата программы…………….………………...19
4. Вывод………………………………………………………...…...21

Цель

Создать программу на языке программирования C++, помогающую следить за успеваемостью учеников. Она должна создавать базу данных с информацией об успеваемости учеников, состоящую из фамилии, имени и четырёх оценок для каждого ученика, в которой можно:

* Добавлять, удалять и редактировать информацию о учениках;
* Осуществлять загрузку и выгрузку данных из бинарного файла;
* Выводить всю имеющуюся информацию на экран;
* Сортировать информацию о студентах по фамилии (в алфавитном порядке) и по среднему баллу.

Все функции должны быть реализованы строго как методы классов.

Документация

## Файл DBLA1/main.cpp

Исходный файл с реализацией меню

#include <QCoreApplication>

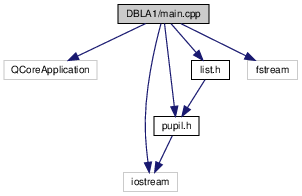
#include <iostream>

#include "pupil.h"

#include "list.h"

#include <fstream>

Граф включаемых заголовочных файлов для main.cpp:



### Функции

* int **main** ()

*Функция* ***main()***

### Подробное описание

Данный файл содержит в себе реализацию простого меню, так же вызываются многие методы классов.

### Функции

#### int main ()

Функция **main()**

Функция-меню.

Код выглядит следующим образом:

cout<<"Wellcome to Data Base: 'Lerning Achivment'!"<<endl;

int Slct;

List list;

while (1)

{

cout<<endl<<"Please, select what you want to do and enter appropriate number:"<<endl<<"0 - Exit;"<<endl<<\

"1 - Load from the file;"<<endl<<"2 - Add an element or elements;"<<endl<<"3 - Delete an element;"<<endl<<\

"4 - Change an element;"<<endl<<"5 - Sort by name;"<<endl<<"6 - Sort by average score;"<<endl<<"7 - Print elements;"<<\

endl<<"8 - Save into the file."<<endl<<"9 - Clean the Base"<< endl << "> ";

cin >> Slct;

cout << endl;

switch (Slct){

case 0:

cout<<"Thank you for using Data Base: 'Lerning Achivment'!"<<endl;

return 0;

case 1:

list.Upload();

break;

case 2:

list.Add();

break;

case 3:

list.Delete();

break;

case 4:

list.Change();

break;

case 5:

list.SortByName();

break;

case 6:

list.SortByAS();

break;

case 7:

list.Show();

break;

case 8:

list.Download();

break;

case 9:

delete &list;

break;

default:

cout<<"Something went wrong. No command with this number. Please, try again."<<endl;

break;

##### Возвращает:

0 - При удачном завершении программы.

## Файл DBLA1/list.cpp

Исходный файл с реализацией функций

#include "list.h"

#include "pupil.h"

#include <cstring>

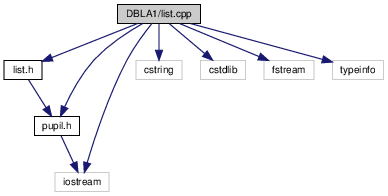
#include <cstdlib>

#include <iostream>

#include <fstream>

#include <typeinfo>

Граф включаемых заголовочных файлов для list.cpp:



### Подробное описание

Исходный файл с реализацией функций

Данный файл содержит в себе реализации основных методов классa **List**, используемых в программе.

## Файл DBLA1/pupil.cpp

Исходный файл с реализацией функций

#include "pupil.h"

#include <cstring>

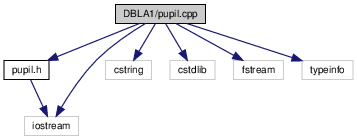
#include <cstdlib>

#include <iostream>

#include <fstream>

#include <typeinfo>

Граф включаемых заголовочных файлов для pupil.cpp:



### Подробное описание

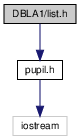
Данный файл содержит в себе реализации основных методов классов: Pupil и List - используемых в программе.

**Файл DBLA1/list.h**

Заголовочный файл с описанием классов.

#include "pupil.h"

Граф включаемых заголовочных файлов для list.h:



### Классы

* class **List**

*Класс представляющий из себя односвязный список.*

### Подробное описание

Заголовочный файл с описанием классов.

Данный файл содержит в себе определение классa **List**, используемого в программе.

## Файл DBLA1/pupil.h

Заголовочный файл с описанием классов

#include <iostream>

Граф включаемых заголовочных файлов для pupil.h:



### Классы

* class **Pupil**

*Класс для структуризации данных.*

### Подробное описание

Данный файл содержит в себе определение класса **Pupil**, используемого в программе.

## Класс List

Класс представляющий из себя односвязный список.

#include <list.h>

### Открытые члены

* **~List** ()

*Деструктор списка.*

* void **Show** ()

*Метод класса* ***List****, выводящий на экран содержимое списка.*

* void **Change** ()

*Метод класса* ***List****, предоставляющий пользователю возможность изменить поля элемента.*

* void **SortByName** ()

*Метод класса* ***List****, сортирующий список по содержимому полей Surname и Name класса* ***Pupil****.*

* void **SortByAS** ()

*Метод класса* ***List****, сортирующий список по среднему значению массива Mark.*

* void **Delete** ()

*Метод класса* ***List****, удаляющий выбранный элемент списка.*

* void **Upload** ()

*Метод класса* ***List****, загружающий список из файла.*

* void **Download** ()

*Метод класса* ***List****, загружающий список в файл.*

* void **Add** ()

*Метод класса* ***List****, предоставляющий пользователю возможность добавить один или несколько элементов в конец списка.*

### Подробное описание

Данный класс представляет из себя односвязный список и содержит все методы, а также указатель на голову списка и его длину. Дружественен классу **Pupil**. Код выглядит следующим образом:

class List: public Pupil

{

public:

List (){lenght=0; Head=nullptr;}

~List (); //Деструктор, удаление списка

void Show (); //Показать весь список

void Change (); //Изменить элемент

void SortByName (); //Сортировка по имени

void SortByAS (); //Сортировка по среднему баллу

void Delete (); //Удаление элемента

void Upload (); //Загрузка элемента из файла

void Download (); //Загрузка элемента в файл

void Add(); //Добавление элемента в список

private:

Pupil \*Head;

int lenght;

Pupil\* GetNth (int N); //Найти Н-ный элемент

};

### Методы

#### void List::Add ()

Метод класса **List**, предоставляющий пользователю возможность добавить один или несколько элементов в конец списка.

Метод запрашивает количество новых элементов и информацию для каждого нового элемента.

Код выглядит следующим образом:

{

int num;

cout<<"You have chosen adding of elements."<<endl<<"Please, enter number of new pupil:"<<endl<<"> ";

cin>>num;

if (num < 0)

{

cout<<"Something went wrong."<<endl;

return;

}

for (int i=0; i<num; i++)

{

Pupil \*Elm = new Pupil;

Elm->SetSurname();

Elm->SetName();

Elm->SetMarks();

lenght++;

Elm->Next = Head;

Head = Elm;

}

}

#### void List::Change ()

Метод класса **List**, предоставляющий пользователю возможность изменить поля элемента.

Метод запрашивает номер элемента, который нужно изменить, и что именно нужно изменить в нем, а потом просит ввести новую информацию.

Код выглядит следующим образом:

{

int Slct;

cout<<"You have chosen changing of element."<<endl<<"Please, look at least of pupils:"<<endl;

Show();

int num;

cout<<"And enter number of pupil you want to change: "<<endl<<"> ";

cin >> num;

Pupil \*Elm = GetNth(num);

if (Elm == nullptr) {return;}

cout<<"Please, select what you want to change and enter appropriate number: "<<endl<<"0 - Exit;"<<\

endl<<"1 - Surname;"<<endl<<"2 - Name;"<<endl<<"3 - Marks;"<<endl<<"4 - all informarion;"<<endl<<"> ";

cin >> Slct;

cout << endl;

switch (Slct){

case 0:

return;

case 1:

Elm->SetSurname();

break;

case 2:

Elm->SetName();

break;

case 3:

Elm->SetMarks();

break;

case 4:

Elm->SetSurname();

Elm->SetName();

Elm->SetMarks();

break;

default:

cout<<"Something went wrong. No command with this number. Please, try again."<<endl;

return;

}

cout<<endl;

}

#### void List::Delete ()

Метод класса **List**, удаляющий выбранный элемент списка.

Метод запрашивает номер элемента, который нужно удаляет, и удаляет его.

Код выглядит следующим образом:

{

cout<<"You have chosen deleting of element. "<<endl<<"Please, look at least of pupils."<<endl;

Show();

int num;

cout<<"And enter number of pupil you want to delete: "<<endl<<"> ";

cin >> num;

Pupil \*Elm = GetNth(num);

if (Elm == nullptr) {return;}

if (num==1)

{

Head = Elm->Next;

delete Elm;

}

else

{

Pupil \*Prev = GetNth(num-1);

Prev->Next = Elm->Next;

delete Elm;

}

cout<<"Pupil number "<<num<<" have deleted from the list."<<endl;

}

#### void List::Download ()

Метод класса **List**, загружающий список в файл.

Метод открывает файл на запись в формате бинарного и загружает каждый элемент файл.

Код выглядит следующим образом:

{

cout<<"You have chosen downloading of list into the file."<<endl;

ofstream fout;

string FName = "C:\\Prog\\build-DBLA1-Desktop\_Qt\_5\_12\_3\_MinGW\_64\_bit-Debug\\Pupils.bin";

fout.open(FName, ios\_base::binary | ios\_base::out);

if (!fout.is\_open())

{

cout<<"The file cannot be open"<<endl;

return;

}

Pupil \*Elm=Head;

while (Elm)

{

Elm->write(fout);

Elm = Elm->Next;

}

fout.close();

cout<<"List have downloaded into the file."<<endl;

}

#### void List::Show ()

Метод класса **List**, выводящий на экран содержимое списка.

Код выглядит следующим образом:

{

Pupil \*Elm=Head;

int num=1;

cout<<"Form of list: Surname, Name, two math marks and two informatic marks."<<endl<<"List of pupils: "<<endl;

while (Elm != nullptr)

{

cout<<num<<". "<<Elm->GetSurname()<<" "<<Elm->GetName()<<", "<<Elm->GetMarks()[0]<<", "<<Elm->GetMarks()[1]<<\

", "<<Elm->GetMarks()[2]<<", "<<Elm->GetMarks()[3]<<";"<<endl;

Elm = Elm->Next;

num++;

}

}

#### void List::SortByAS ()

Метод класса **List**, сортирующий список по среднему значению массива Mark.

Находит среднее значение текущего элемента и соседнего с ним элемента, сравнивает их и, если они идут в неправильном порядке, то меняет их местами.

Код выглядит следующим образом:

{

cout<<"You have chosen sorting of list by average score. Pupil with the highest value will in the begining. "<<endl;

Pupil\* ptr = Head, \*tmp = nullptr, \*prev = nullptr;

bool flag = false;

float AS1 = 0.0, AS2 = 0.0;

if(Head)

{

do

{

flag = false;

ptr = Head;

while(ptr->Next)

{

AS1=(ptr->GetMarks()[0]+ptr->GetMarks()[1]+ptr->GetMarks()[2]+ptr->GetMarks()[3])/4;

AS2=(ptr->Next->GetMarks()[0]+ptr->Next->GetMarks()[1]+ptr->Next->GetMarks()[2]+ptr->Next->GetMarks()[3])/4;

if(AS1<AS2)

{

if(ptr == Head)

{

tmp = ptr;

ptr = tmp->Next;

tmp->Next = ptr->Next;

ptr->Next = tmp;

Head = ptr;

flag = true;

}

else

{

tmp = ptr;

ptr = tmp->Next;

tmp->Next = ptr->Next;

ptr->Next = tmp;

prev->Next = ptr;

flag = true;

}

}

prev = ptr;

ptr = ptr->Next;

}

}

while(flag);

cout<<"List was sorted. "<<endl;

}

else

cout<<"List is empty. List cannot be sorted."<<endl;

cout<<endl;

}

#### void List::SortByName ()

Метод класса **List**, сортирующий список по содержимому полей Surname и Name класса **Pupil**.

Склеивает Surname и Name текущего элемента и соседнего с ним элемента в одну строку, сравнивает их и, если они идут в неправильном порядке, то меняет их местами.

Код выглядит следующим образом:

{

cout<<"You have chosen sorting of list."<<endl;

Pupil\* ptr = Head, \*tmp = nullptr, \*prev = nullptr;

bool flag = false;

if(Head)

{

do

{

flag = false;

ptr = Head;

while(ptr->Next)

{

char \*str1 = new char[41], \*str2 = new char[41];

strcat (str1, ptr->GetSurname());

strcat (str1, " ");

strcat (str1, ptr->GetName());

strcat (str2, ptr->Next->GetSurname());

strcat (str2, " ");

strcat (str2, ptr->Next->GetName());

if(stricmp(str1, str2) > 0)

{

if(ptr == Head)

{

tmp = ptr;

ptr = tmp->Next;

tmp->Next = ptr->Next;

ptr->Next = tmp;

Head = ptr;

flag = true;

}

else

{

tmp = ptr;

ptr = tmp->Next;

tmp->Next = ptr->Next;

ptr->Next = tmp;

prev->Next = ptr;

flag = true;

}

}

prev = ptr;

ptr = ptr->Next;

delete [] str1;

delete [] str2;

}

}

while(flag);

cout<<"List was sorted. "<<endl;

}

else

cout<<"List is empty. List cannot be sorted."<<endl;

cout<<endl;

}

#### void List::Upload ()

Метод класса **List**, загружающий список из файла.

Метод открывает файл на чтение в формате бинарного и создает новые элементы в списке.

Код выглядит следующим образом:

{

cout<<"You have chosen uploading of list from the file."<<std::endl;

ifstream fin;

fin.open("C:\\Prog\\build-DBLA1-Desktop\_Qt\_5\_12\_3\_MinGW\_64\_bit-Debug\\Pupils.bin", ios\_base::binary | ios\_base::in);

if (!fin.is\_open())

{

cout<<"The file cannot be open"<<endl;

exit (-1);

}

while (!fin.eof())

{

Pupil \*Elm = new Pupil;

Elm->read(fin);

if(fin.eof())

{

delete Elm;

break;

}

lenght++;

Elm->Next = Head;

Head = Elm;

}

cout<<"List have uploaded from the file."<<endl;

Show();

fin.close();

}

#### Объявления и описания членов классов находятся в файлах:

* DBLA1/**list.h**
* DBLA1/**list.cpp**

## Класс Pupil

Класс для структуризации данных.

#include <pupil.h>

Граф связей класса Pupil:



### Открытые члены

* **Pupil** ()

*Конструктор класса* ***Pupil****.*

* void **SetPupil** (char \*Surname, char \*Name, int \*Narks)

*Метод класса* ***Pupil****, изменяющий поля элемента.*

* void **SetSurname** ()

*Метод класса* ***Pupil****, изменяющий поле Surname.*

* void **SetName** ()

*Метод класса* ***Pupil****, изменяющий поле Name.*

* void **SetMarks** ()

*Метод класса* ***Pupil****, изменяющий поле Mark.*

* char \* **GetSurname** ()

*Метод класса* ***Pupil****, возвращающий содержимое поля Surname.*

* char \* **GetName** ()

*Метод класса* ***Pupil****, возвращающий содержимое поля Name.*

* int \* **GetMarks** ()

*Метод класса* ***Pupil****, возвращающий содержимое поля Mark.*

* bool **write** (std::ostream &os)

*Метод класса* ***Pupil****, упрощающий запись в файл.*

* bool **read** (std::istream &is)

*Метод класса* ***Pupil****, упрощающий чтение из файла.*

### Открытые атрибуты

* **Pupil** \* **Next**

### Подробное описание

Класс для структуризации данных.

В классе представлена информация о студенте, такая как имя, фамилия и 4 оценки, а так же функции для работы с этими данными. Дружественен классу **List**. Код выглядит следующим образом:

class Pupil

{

private:

char Surname[20], Name[20];

int Mark[4];

Pupil \*Next;

Pupil ();

void SetPupil(char\* Surname, char\* Name, int\* Narks);

void SetSurname ();

void SetName ();

void SetMarks ();

char\* GetSurname ();

char\* GetName ();

int\* GetMarks ();

bool write(std::ostream& os);

bool read (std::istream& is);

friend class List;

};

### Методы

#### int \* Pupil::GetMarks ()

Метод класса **Pupil**, возвращающий содержимое поля Mark.

Код выглядит следующим образом:

{return this->Mark;}

##### Возвращает:

содержимое информационного поля "Mark".

#### char \* Pupil::GetName ()

Метод класса **Pupil**, возвращающий содержимое поля Name.

Код выглядит следующим образом:

{return this->Name;}

##### Возвращает:

содержимое информационного поля "Name".

#### char \* Pupil::GetSurname ()

Метод класса **Pupil**, возвращающий содержимое поля Surname.

Код выглядит следующим образом:

{return this->Surname;}

##### Возвращает:

содержимое информационного поля "Surname".

#### bool Pupil::read (std::istream & is)

Метод класса **Pupil**, упрощающий чтение из файла.

Код выглядит следующим образом:

{

is.read((reinterpret\_cast<char \*>(this)), sizeof(Pupil));

return is.good();

}

##### Возвращает:

“False” если произошла какая-либо ошибка и “true” в ином случае.

#### void Pupil::SetMarks ()

Метод класса **Pupil**, изменяющий поле Mark.

Метод просит ввести 4 числа и сохраняет их в поле Mark.

Код выглядит следующим образом:

{

cout<<"Please, enter four the pupil's marks (two math marks and two informatic marks): "<<endl<<"> ";

cin>>this->Mark[0]>>this->Mark[1]>>this->Mark[2]>>this->Mark[3];

}

#### void Pupil::SetName ()

Метод класса **Pupil**, изменяющий поле Name.

Метод просит на ввод строку, проверяет её на соответствию формату и, если соответствует, то копирует в поле Name.

Код выглядит следующим образом:

{

char str[20];

cout<<"Enter pupil's name:"<<endl<<"> ";

cin>>str;

if (strlen(str)<20) {strcpy (this->Name, str);}

else {

cout<<"Something went wrong. We cannot keep Name longer that 20 chars. Sorry."<<endl;

return;

}

}

#### void Pupil::SetPupil (char \* Surname, char \* Name, int \* Narks)

Метод класса **Pupil**, изменяющий поля элемента.

Код выглядит следующим образом:

{

if (strlen(Surname)<20) {strcpy (this->Surname, Surname);}

else {

cout<<"Something went wrong. We cannot keep Surname longer that 20 chars. Sorry."<<endl;

return;

}

if (strlen(Name)<20) {strcpy (this->Name, Name);}

else {

cout<<"Something went wrong. We cannot keep Name longer that 20 chars. Sorry."<<endl;

return;

}

for (int i=0;i<4;i++)

{

this->Mark[i]=Marks[i];

}

}

##### Аргументы:

|  |  |
| --- | --- |
| *Surname* | строка, содержащая фамилию. |
| *Name* | строка, содержащая имя. |
| *Marks[4]* | массив целых чисел, содержащий оценки. |

#### void Pupil::SetSurname ()

Метод класса **Pupil**, изменяющий поле Surname.

Метод просит на ввод строку, проверяет её на соответствию формату и, если соответствует, то копирует в поле Surname.

Код выглядит следующим образом:

{

char str[20];

cout<<"Enter pupil's surname:"<<endl<<"> ";

cin>>str;

if (strlen(str)<20) {strcpy (this->Surname, str);}

else {

cout<<"Something went wrong. We cannot keep Surname longer that 20 chars. Sorry."<<endl;

return;

}

}

#### bool Pupil::write (std::ostream & os)

Метод класса **Pupil**, упрощающий запись в файл.

Код выглядит следующим образом:

{

os.write((reinterpret\_cast<char \*>(this)), sizeof(Pupil));

return os.good();

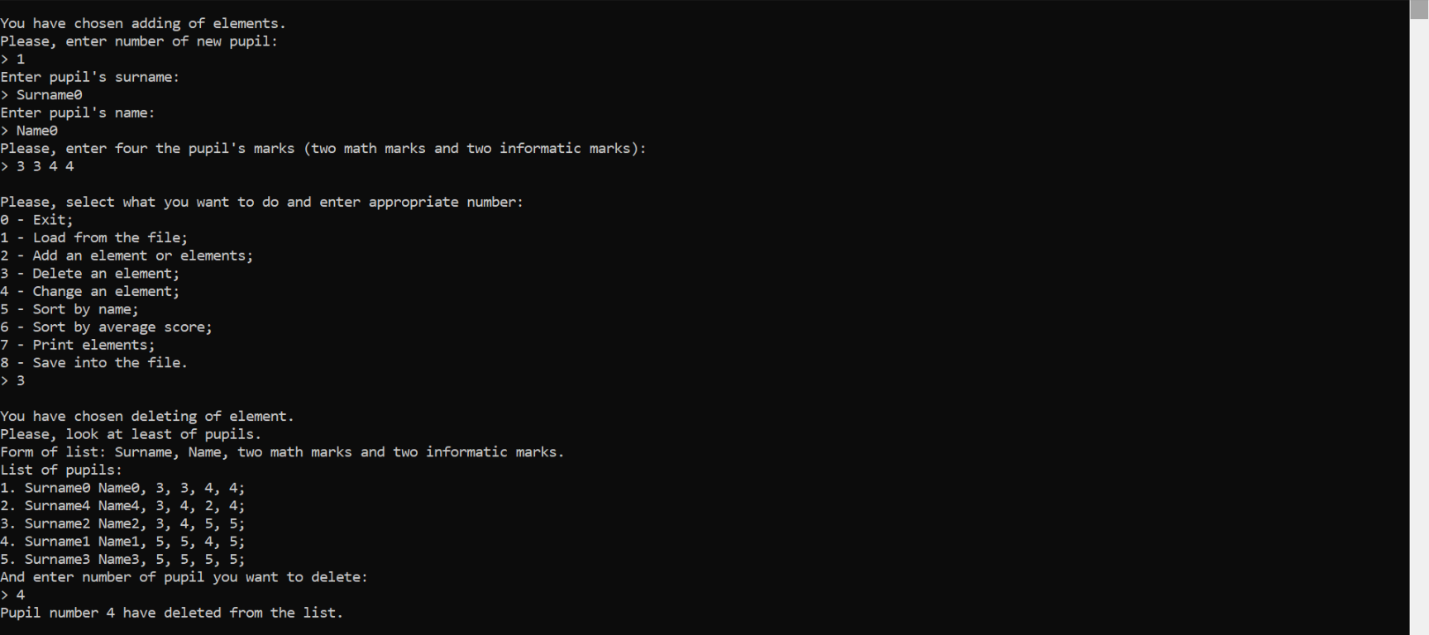
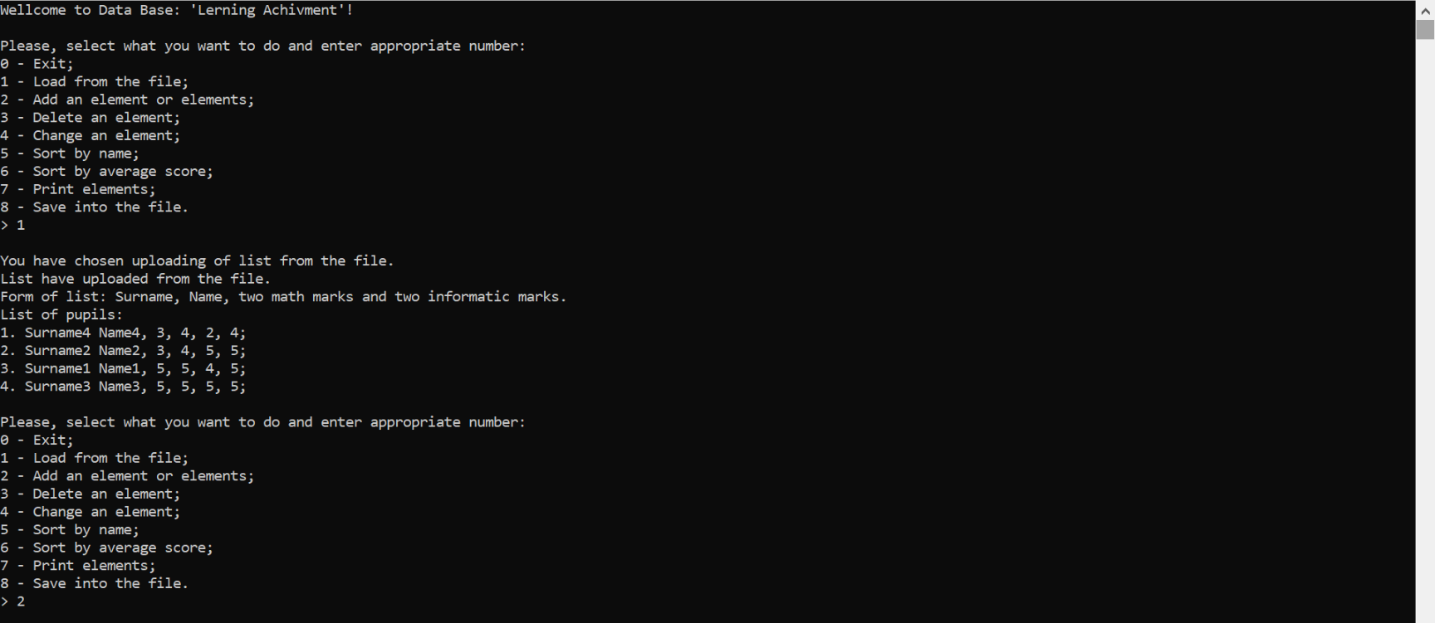
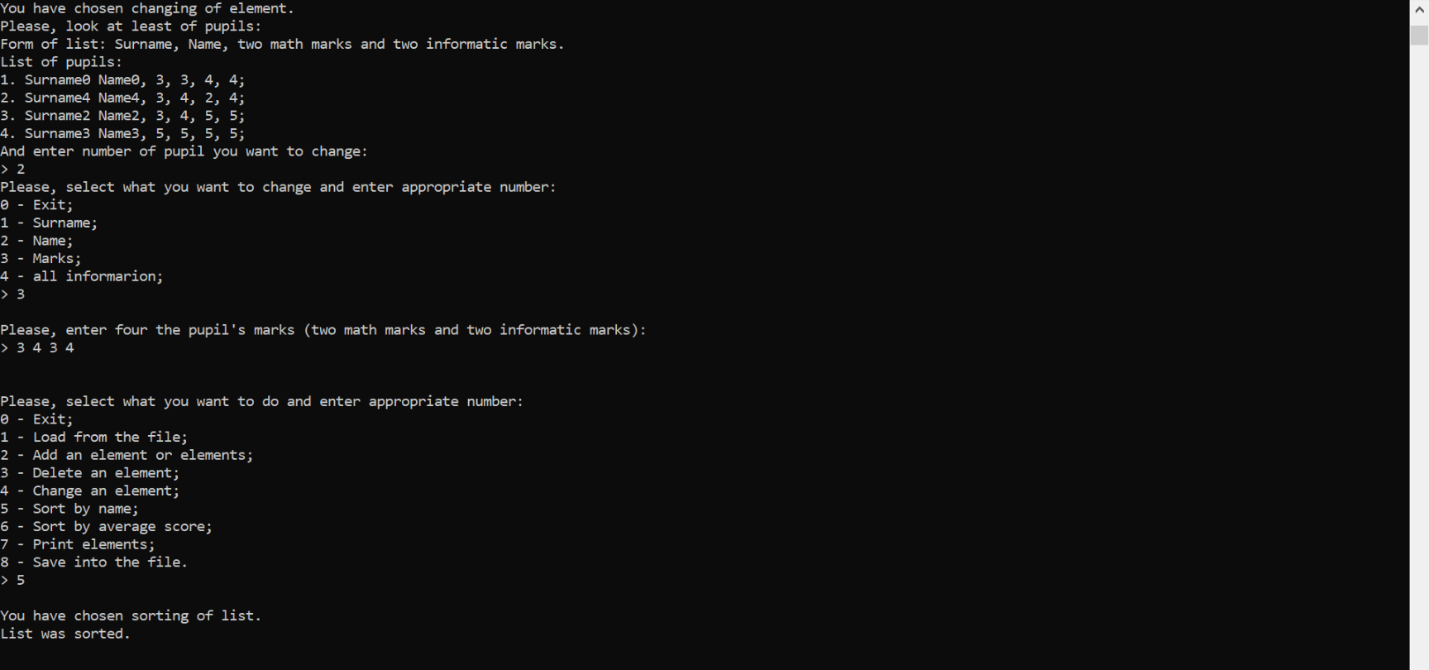
}

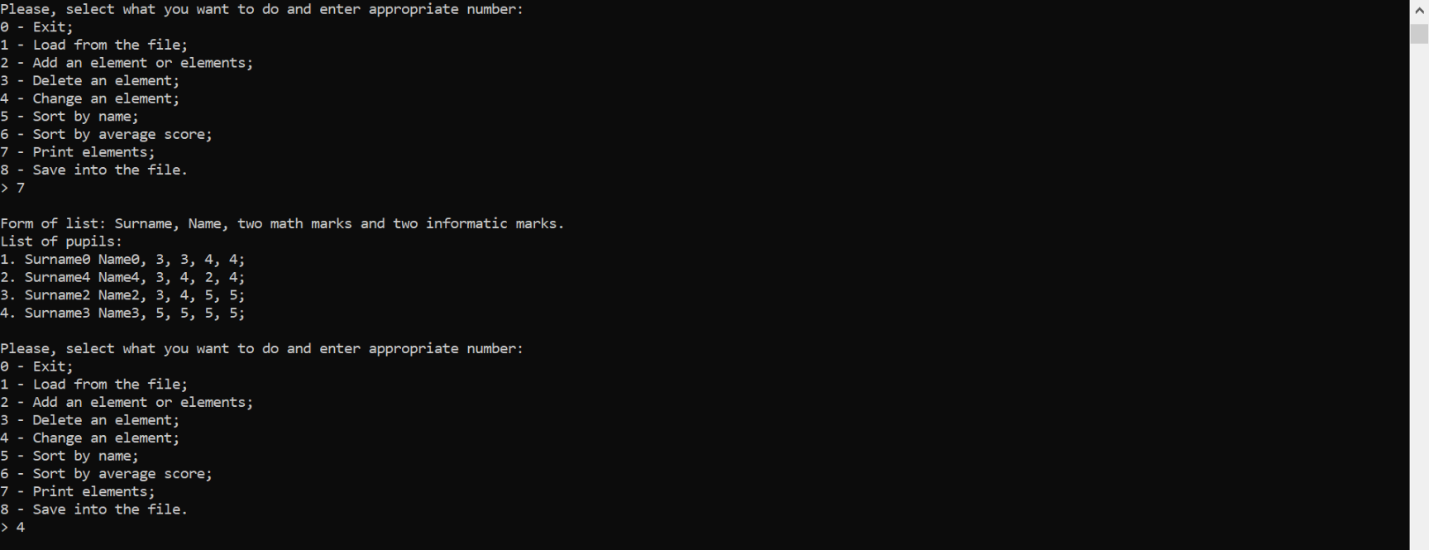
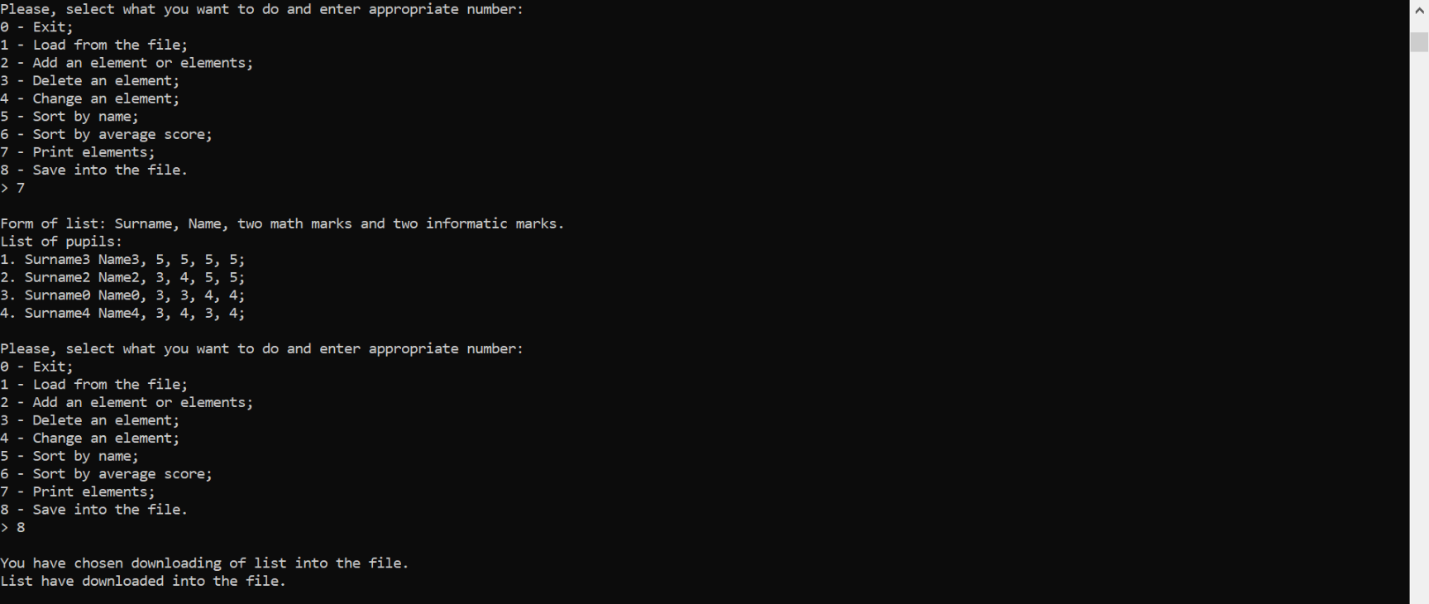
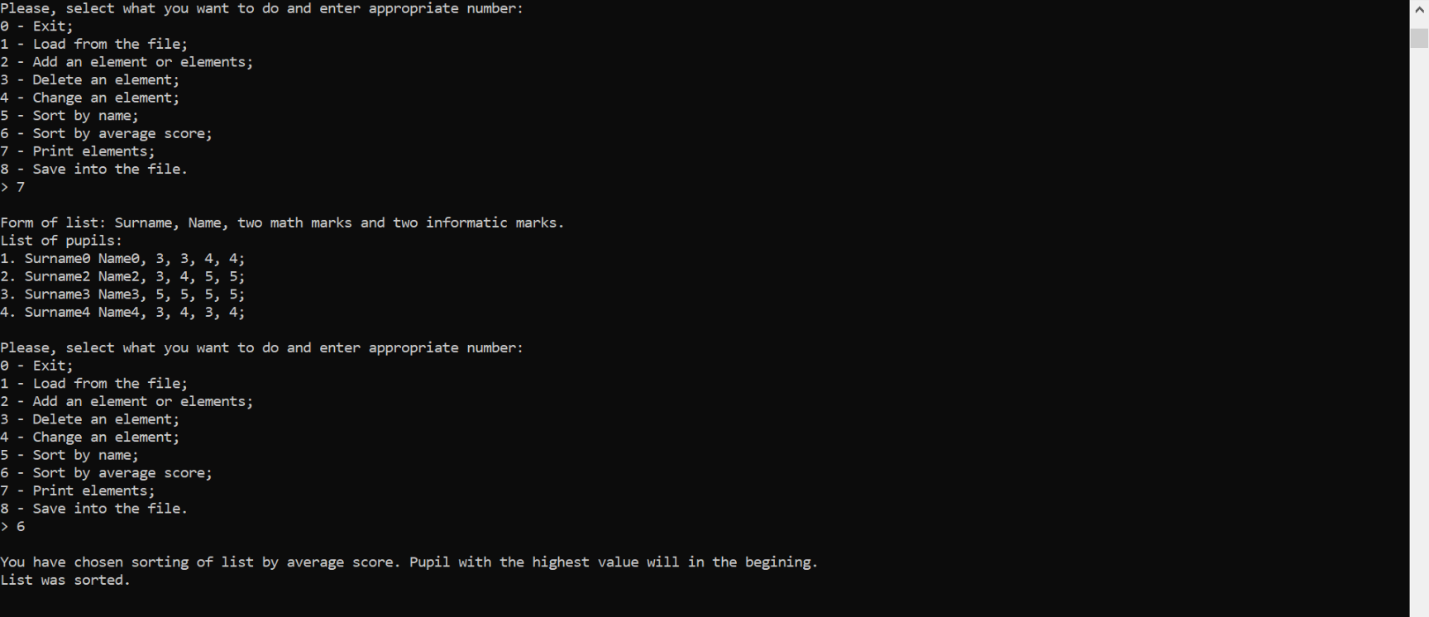
##### Возвращает:

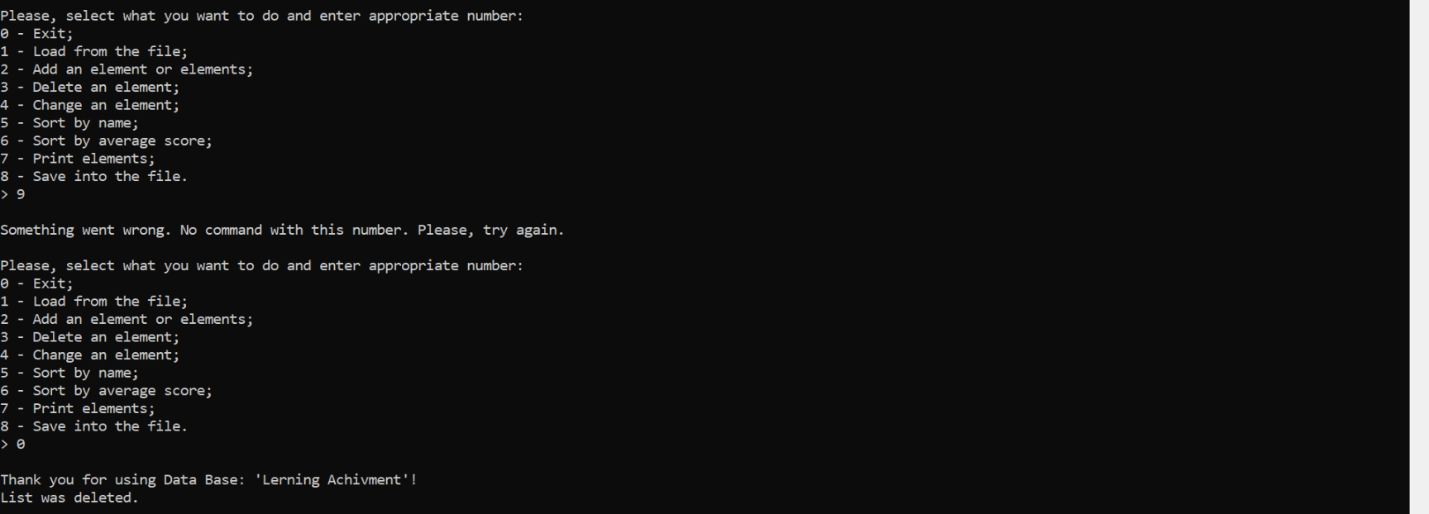
false если произошла какая-либо ошибка и true в ином случае.

#### Объявления и описания членов классов находятся в файлах:

* DBLA1/**pupil.h**
* DBLA1/**pupil.cpp**

Скриншоты результата работы программы





Вывод:

В результате выполнения лабораторной работы была получена программа, работающая с базой данных. Она выполняет:

* Добавление, удаление и редактирование информации об учениках;
* Осуществляет загрузку и выгрузку данных из бинарного файла;
* Выводит всю имеющуюся информацию на экран;
* Сортирует информацию о студентах по фамилии (в алфавитном порядке) и по среднему баллу.

В случае отсутствия возможности выполнения некоторых функций, программа предупредит об этом. Единственное, что совершается вне методов классов – вывод на экран меню и вызов методов. Документация создана при помощи Doxygen.