МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе № 4**

**«Дружественные функции и классы. Методика выбора необходимых решений в ходе разработки ПО»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБ-121», «АВТФ» *доцент кафедры ЗИ*

*Новиков Втюрин Александр Романович Архипова А. Б.*

«21» декабря 2022г«\_\_\_» \_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2022

**Цели и задачи работы:**изучение принципов реализации дружественных функций и работы с ними.

**Задания к работе:**

В рамках задания необходимо выполнить 2 реализации работы бизнес-процесса по заданной предметной области. Первая с использованием принципа friendly. Вторая с использованием принципа GET/SET.

**Вариант 5.**

**Реализовать сбор заявок на соревнования и последующую печать дипломов участников (сокрыть персональные данные).**

Дополнительные требования:

- Наличие функционального класса.

- Терминальное или графическое общение с пользователем.

- Вывод на печать через инструменты ОС.

- Реализация уровней доступа к персональным данным.

**С++**

**Main.cpp**

#pragma once

#include "Application.h"

using namespace std;

int main()

{

Applications application;

mainmenu(application);

}

**Application.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

#include <windows.h>

#include <map>

#include <ctime>

using namespace std;

class LogIn

{

public:

int UserLogIn(const int& username, const string& userpass);

bool AdminLogIn(const int& username, const string& userpass);

protected:

pair <int, string> admin = { 1111, "admin" };

map <int, string> users;

};

class Applications : public LogIn

{

public:

Applications()

{

admin = { 1111, "admin" };

}

Applications(const int& username, const string& adminpass)

{

admin.first = username;

admin.second = adminpass;

}

friend ostream& operator<< (ostream& os, const Applications& applications)

{

os << "Member number\tMember\t\tMember result\n";

map<int, string>::const\_iterator it;

map<int, int>::const\_iterator itr;

for (it = applications.members.begin(), itr = applications.result.begin(); it != applications.members.end(), itr != applications.result.end(); ++it, itr++)

os << (\*it).first << "\t\t" << (\*it).second << "\t\t" << (\*itr).second << endl;

return os;

}

/\*friend istream& operator>>(istream& is, Applications& applications)

{

is <<

return is;

}\*/

map <int, int> result;

void PrintCertificate(const int& username);

int GenMemberNumber();

friend void AddMember(Applications& applications, pair <int, string>& member, const string& password);

void GetMembers();

friend void AddResult(Applications& applications, pair <int, int>& member);

void FindMember(const int& membernumber);

protected:

map <int, string> members;

};

void AddMember(Applications& applications, pair <int, string>& member, const string& password);

void AddResult(Applications& applications, pair <int, int>& member);

void adminmenu(Applications& application);

void usermenu(Applications& application, const int& username);

void mainmenu(Applications& application);

**Menu.cpp**

#pragma once

#include "Application.h"

using namespace std;

void adminmenu(Applications& application)

{

while (true)

{

system("cls");

cout << "What do you need?\n" << "1)Add member;\n" << "2)Add result;\n" << "3)Find member;\n" << "4)Display a table of members;\n" << "5)Printing of the certificate;\n" << "6)Exit\n";

int choice;

cin >> choice;

if (choice == 1)

{

string password;

pair <int, string> member;

cout << "Enter the full name of member:" << endl;

string name;

cin.ignore();

getline(cin, member.second);

//cin.ignore();

cout << "Come up with a password: ";

getline(cin, password);

member.first = application.GenMemberNumber();

AddMember(application, member, password);

}

else if (choice == 2)

{

pair <int, int> result;

cout << "Enter the member number: ";

cin >> result.first;

cout << "Enter the member result: ";

cin >> result.second;

AddResult(application, result);

}

else if (choice == 3)

{

int membernum;

cout << "Enter the member number: ";

cin >> membernum;

application.FindMember(membernum);

}

else if (choice == 4)

{

cout << application;

system("pause");

}

else if (choice == 5)

{

int username;

cout << "Enter the member number: ";

cin >> username;

application.PrintCertificate(username);

}

else if (choice == 6)

{

break;

}

}

}

void usermenu(Applications& application, const int& username)

{

while (true)

{

system("cls");

cout << "What do you need?\n" << "1)View the results table\n" << "2)View your result\n" << "3)Printing of the certificate\n" << "4)Exit" << endl;

int choice;

cin >> choice;

if (cin.fail())

{

cout << "Incorrect input!!!" << endl;

continue;

}

switch (choice)

{

case 1:

{

application.GetMembers();

break;

}

case 2:

{

application.FindMember(username);

break;

}

case 3:

{

application.PrintCertificate(username);

}

case 4:

{

return;

}

default:

{

system("cls");

cout << "Incorrect input!!!" << endl;

system("pause");

break;

}

}

}

}

void mainmenu(Applications& application)

{

while (true)

{

system("cls");

cout << "What do you need?\n" << "1)Submit an application\n" << "2)LogIn\n" << "3)Exit" << endl;

int choice;

cin >> choice;

if (cin.fail())

{

system("cls");

cout << "Incorrect input!!!" << endl;

cin.clear();

cin.ignore();

system("pause");

continue;

}

switch (choice)

{

case 1:

{

string password;

pair <int, string> member;

cout << "Enter the full name of member:" << endl;

string name;

cin.ignore();

getline(cin, member.second);

//cin.ignore();

cout << "Come up with a password: ";

getline(cin, password);

member.first = application.GenMemberNumber();

AddMember(application, member, password);

break;

}

case 2:

{

while (true)

{

string password;

int username;

cout << "Enter the member's number: ";

cin >> username;

if (cin.fail())

{

continue;

}

cout << "Enter the password: ";

cin.ignore();

getline(cin, password);

if (application.AdminLogIn(username, password))

{

adminmenu(application);

break;

}

if (application.UserLogIn(username, password) != 0)

{

usermenu(application, username);

break;

}

}

break;

}

case 3:

exit(1);

}

}

}

**MembersFunc.cpp**

#pragma once

#include "Application.h"

using namespace std;

int LogIn::UserLogIn(const int& username, const string& userpass)

{

if (users.empty())

{

cout << "No registered users" << endl;

system("pause");

return 0;

}

for (auto i : users)

{

if (i.first == username && i.second == userpass)

{

return i.first;

}

else continue;

}

return 0;

}

bool LogIn::AdminLogIn(const int& username, const string& userpass)

{

if (username == admin.first && userpass == admin.second)

{

return true;

}

else return false;

}

void Applications::PrintCertificate(const int& username)

{

map <int, string> ::iterator it = members.find(username);

int place = 1;

for (auto i : result)

{

if (i.second > (\*it).first)

{

place++;

}

}

ofstream fout;

fout.open("Certificate.txt");

if (fout.is\_open())

{

fout << "Diploma\n" << endl;

fout << "to" << endl;

fout << "\n" << (\*it).second << endl;

fout << "\n" << "for achiving the " << place << " place" << endl;

fout << "\n\n" << "Director\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

fout << "Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

fout.close();

}

void AddMember(Applications& applications, pair <int, string>& member, const string& password)

{

system("cls");

applications.members.insert(member);

pair <int, int> res = { member.first, 0 };

applications.result.insert(res);

pair <int, string> user = { member.first, password };

applications.users.insert(user);

cout << "Application accepted" << endl;

cout << "Your member number: " << member.first << endl;

cout << "Your password: " << password << endl;

system("pause");

}

void Applications::GetMembers()

{

system("cls");

cout << "Member number\t Result" << endl;

for (auto i : result)

{

cout << i.first << "\t\t " << i.second << endl;

}

system("pause");

}

void AddResult(Applications& applications, pair <int, int>& member)

{

system("cls");

if (applications.result.find(member.first) != applications.result.end())

{

map <int, int> ::iterator it;

it = applications.result.find(member.first);

(\*it).second = member.second;

}

else cout << "There is no such member";

system("pause");

}

int Applications::GenMemberNumber()

{

system("cls");

while (true)

{

srand(time(NULL));

int membernum = rand() % 1000 + 1;

map <int, string> ::iterator it;

if (members.find(membernum) == members.end())

{

return membernum;

}

else continue;

}

}

void Applications::FindMember(const int& membernumber)

{

system("cls");

map<int, string>::iterator it;

map <int, int>::iterator itr;

it = members.find(membernumber);

itr = result.find(membernumber);

if (it != members.end() && itr != result.end()) {

cout << "Member number\t Member\t Result" << endl;

cout << (\*it).first << "\t " << (\*it).second << "\t " << (\*itr).second << endl;

}

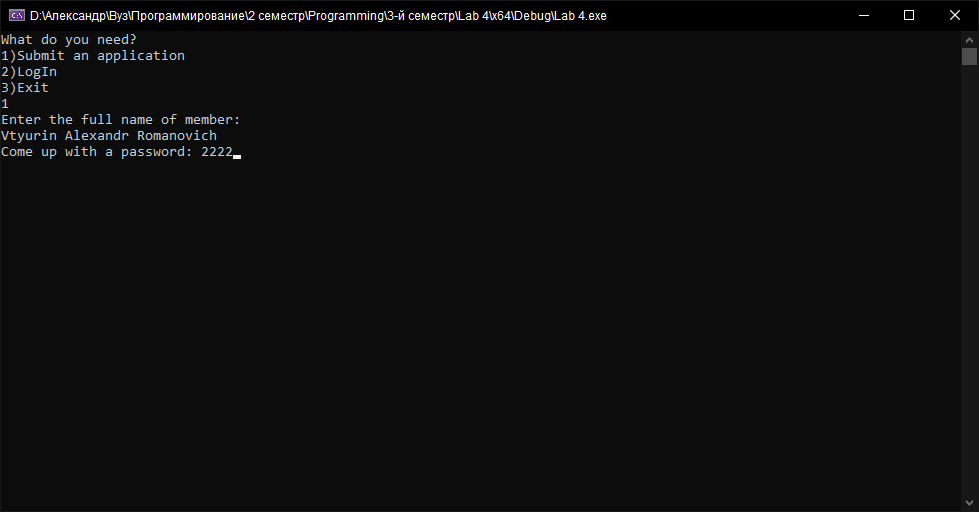
else cout << "There is no such member";

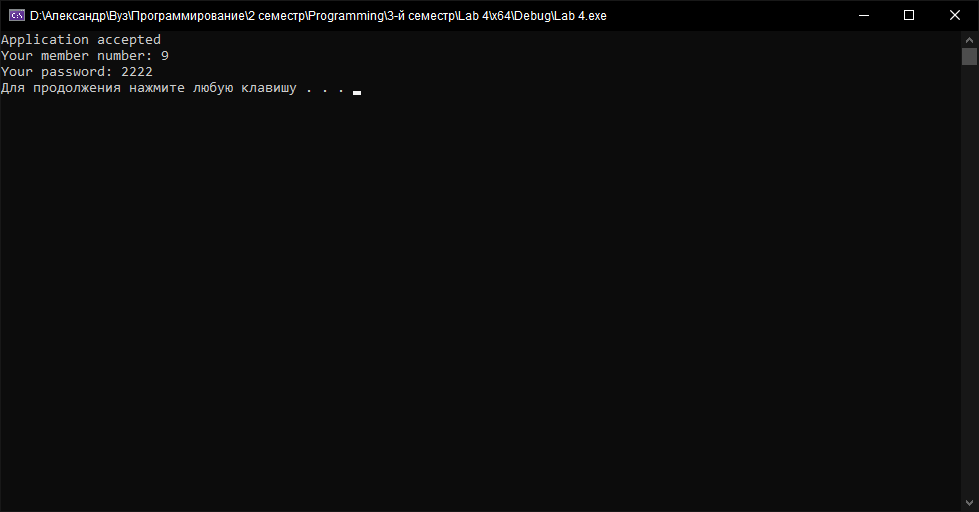
}

LogIn menu

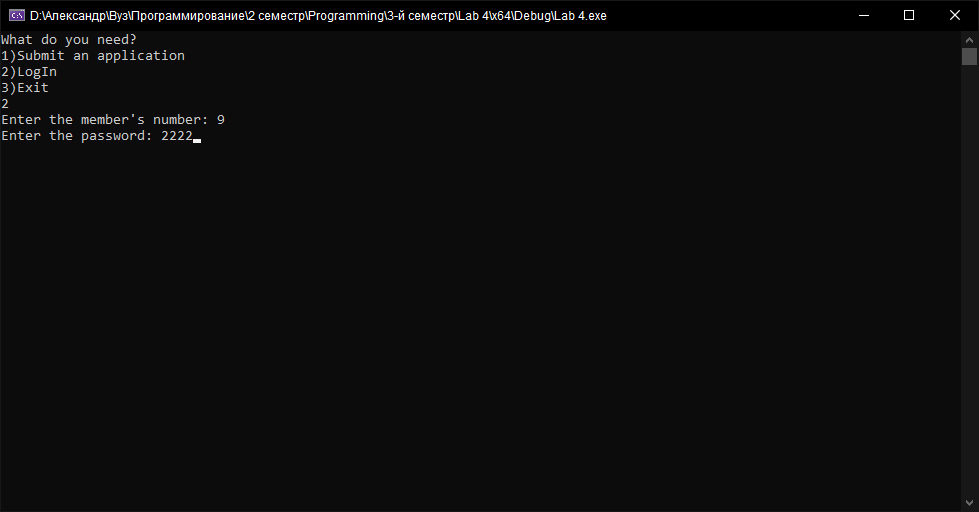


Submit an application





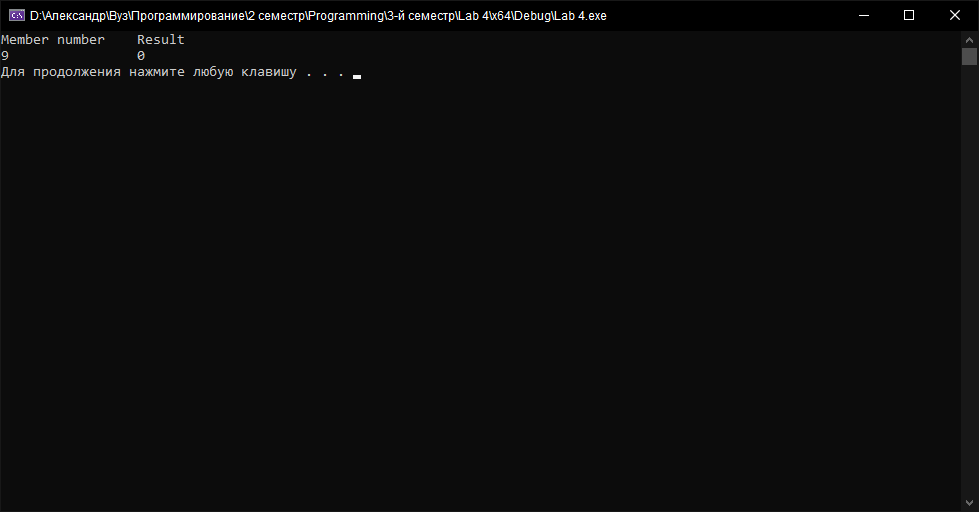
LogIn User



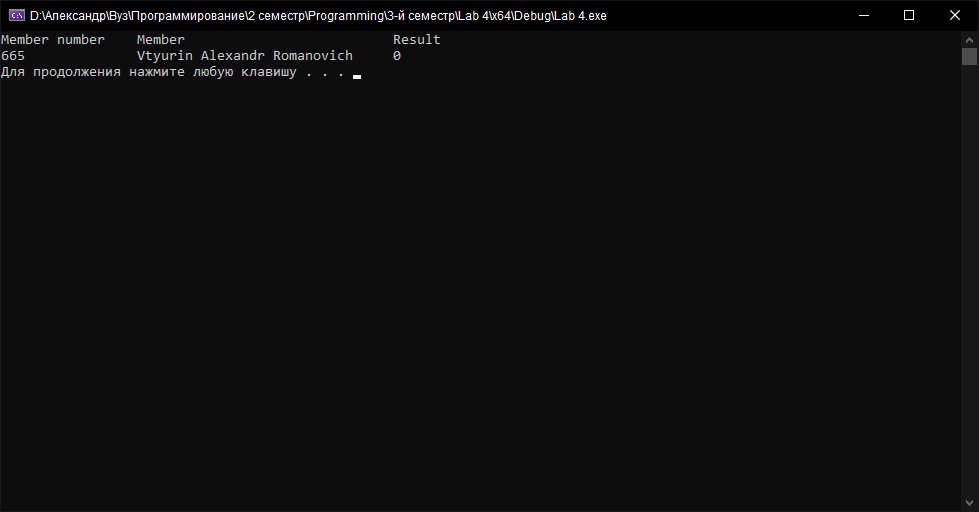
User menu



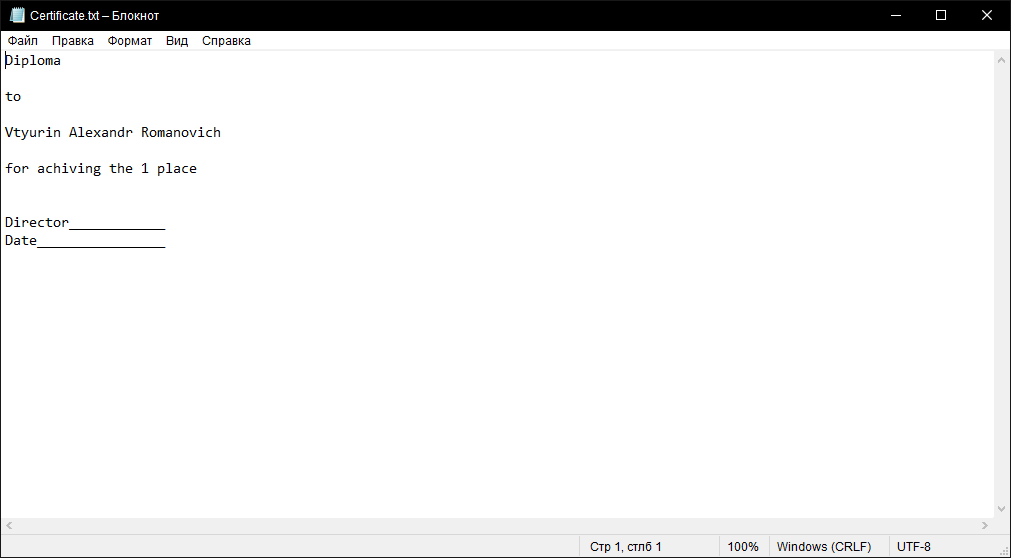
View the results table



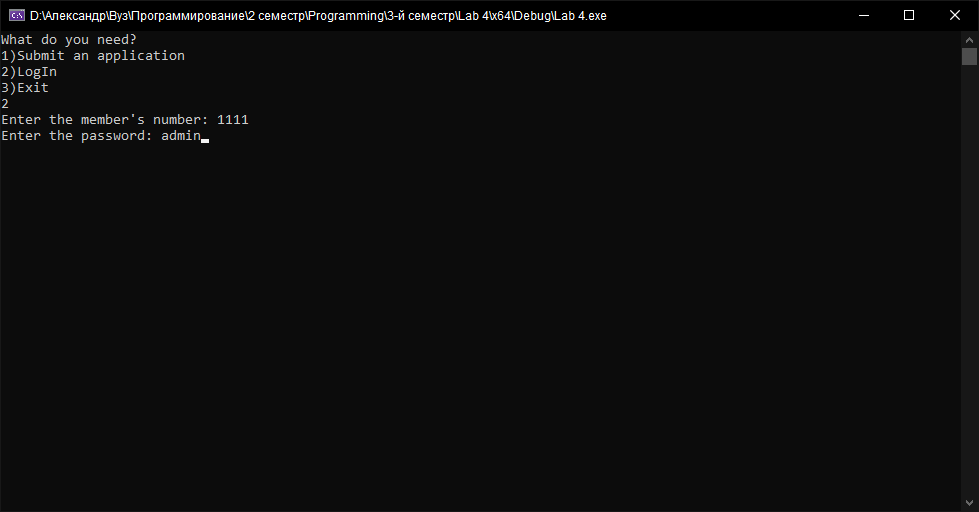
View your result



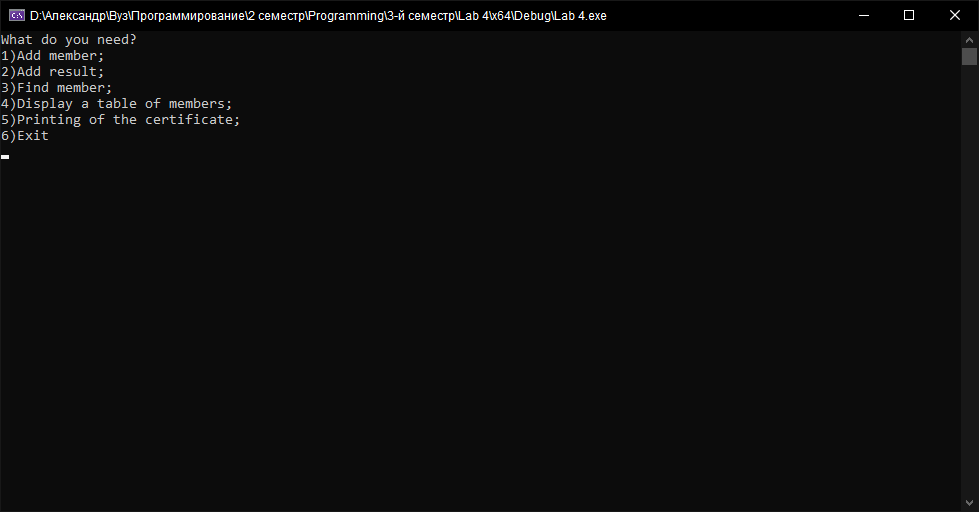
Printing of the certificate



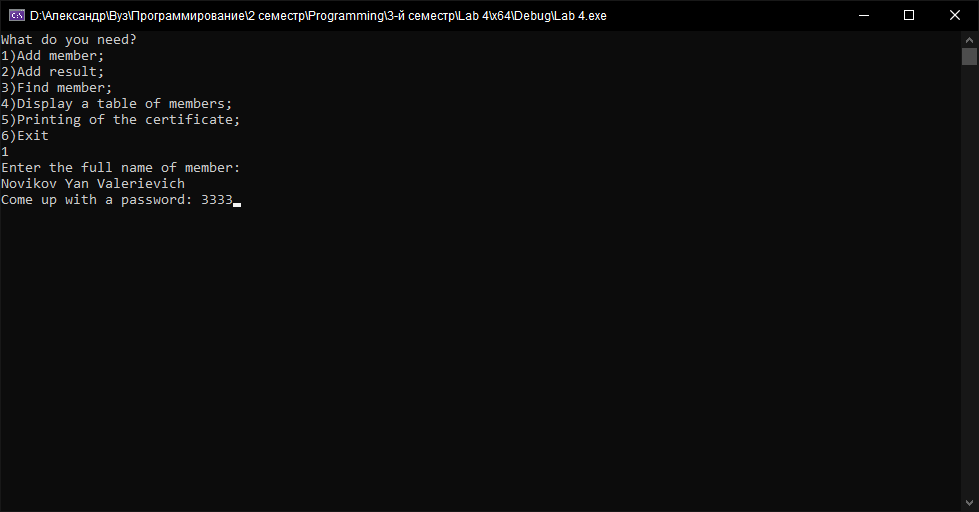
LogIn admin

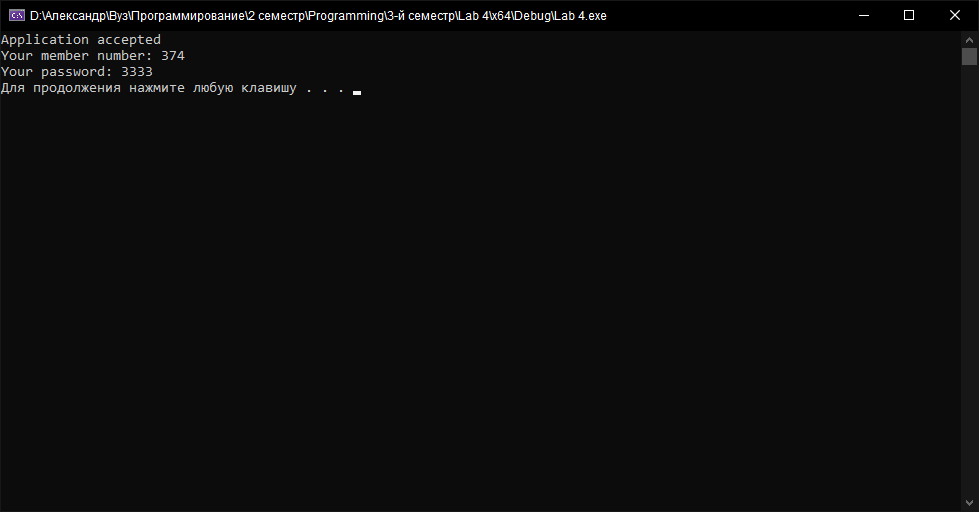


Admin menu

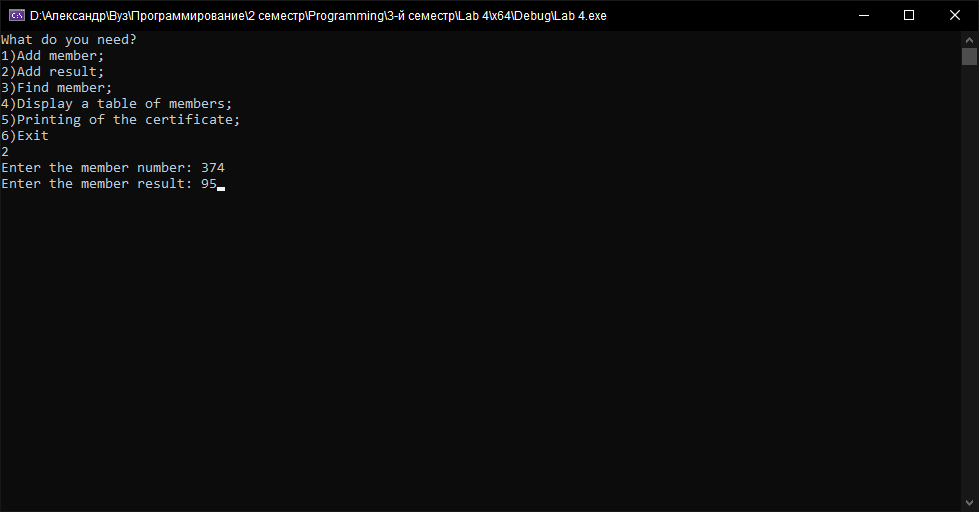


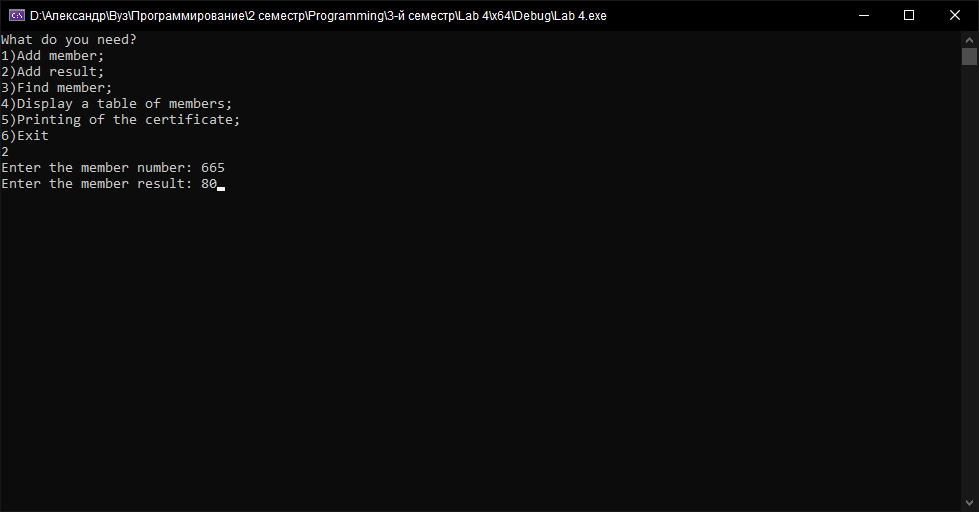
Add member



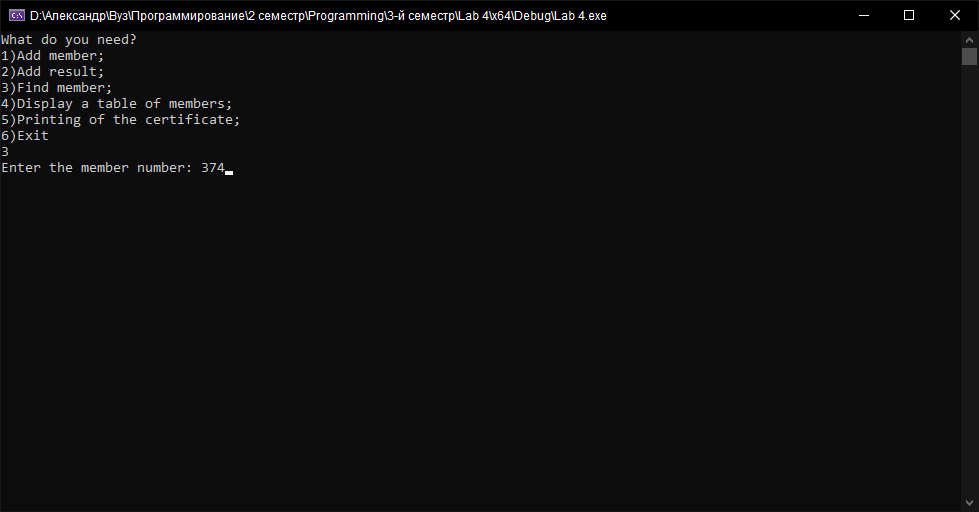


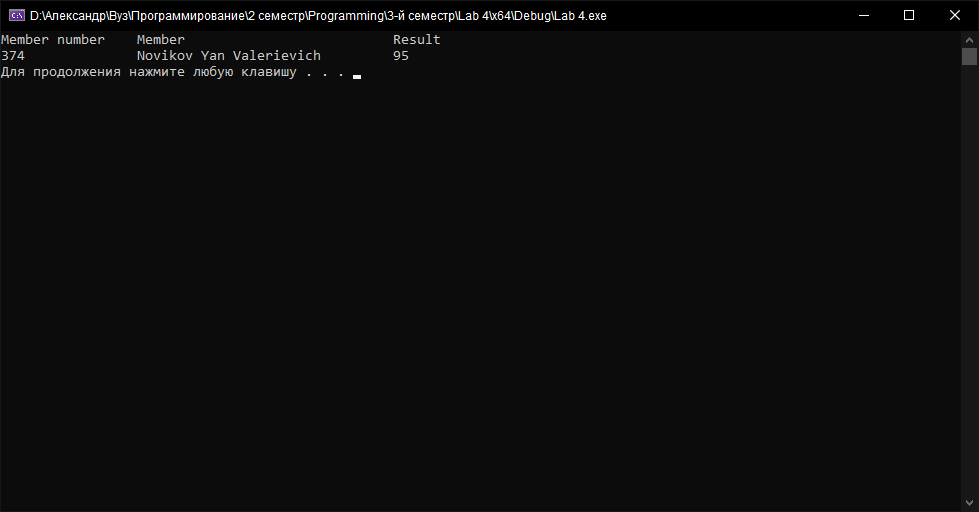
Add result



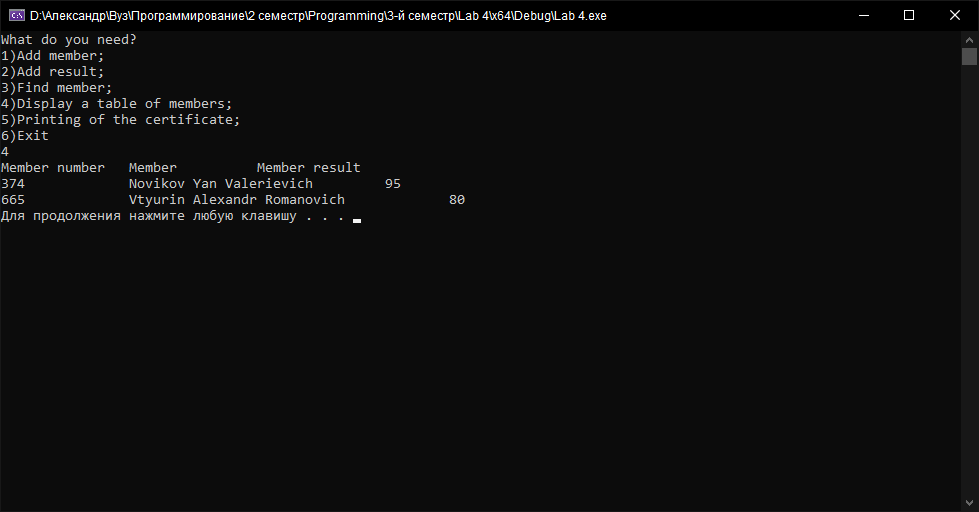


Find member





Display a table of members



Printing of the certificate

