

Задание:

Модифицировать программу из ЛР № 2.   Заменить struct на class; Функции становятся методами класса; Поля становятся private, доступ к ним при необходимости осуществляется через методы public; Дополнительно в main продемонстрировать работу с динамическим массивом объектов класса и с массивом динамических объектов класса; применение операторов new, delete, delete[]; использование (\*a) и ->.

Решение:

Файл Teacher.h

#pragma once

#include <iostream>

#include <stdio.h>

#include <string>

#include <locale.h>

using namespace std;

// Класс Учитель

class Teacher

{

private:

string Fam = ""; // Фамилия

string Name = ""; // Имя

string Otch = ""; // Отчество

public:

// Конструктор

Teacher();

// Фун-ии получения данных из полей

string GetFam();

string GetName();

string GetOtch();

// Деструктор

~Teacher();

// Инициализация структуры Учитель

void SetTeach(string Fam, string Name, string Otch);

// Ввод учителя

void InpTeachFIO();

// Ввод учителя на экран

void DisplayTeach();

};

Файл Teacher.cpp

#include "Teacher.h"

// Конструктор

Teacher::Teacher()

{

Fam = "";

Name = "";

Otch = "";

}

// Фун-ии получения данных из полей

string Teacher::GetFam()

{

return this->Fam;

}

string Teacher::GetName()

{

return this->Name;

}

string Teacher::GetOtch()

{

return this->Otch;

}

// Инициализация структуры Учитель

void Teacher::SetTeach(string Fam\_s, string Name\_s, string Otch\_s)

{

this->Fam = Fam\_s;

this->Name = Name\_s;

this->Otch = Otch\_s;

}

// Ввод учителя

void Teacher::InpTeachFIO()

{

cout << "Введите Фамилию учителя: ";

cin >> Fam;

cout << "Введите имя учителя: ";

cin >> Name;

cout << "Введите отчество учителя: ";

cin >> Otch;

}

// Ввод учителя

void Teacher::DisplayTeach()

{

cout << "ФИО учителя: " << Fam << " " << Name << " " << Otch << endl;

}

// Деструктор

Teacher::~Teacher()

{

;

}

Файл Lesson.h

#pragma once

#include "Teacher.h"

// Класс Уроки

class Lesson

{

private:

string Name = ""; // Название предмета

Teacher Teach; // Учитель

public:

// Конструктор

Lesson();

// Фун-ии получения данных из полей

string GetName();

Teacher GetTeach();

// Деструктор

~Lesson();

// Инициализация структуры Уроки

void SetLess(string Name\_Less, Teacher t);

// Обнуление полей

void NullLess();

// Ввод названия урока

void InpLessName(Teacher t);

// Вывод уроков

void out\_Less();

};

Файл Lesson.cpp

#include "Lesson.h"

// Конструктор

Lesson::Lesson()

{

this->Name = "";

this->Teach.SetTeach("", "", "");

}

// Инициализация структуры Уроки

void Lesson::SetLess(string Name\_Less, Teacher t)

{

Name = Name\_Less;

Teach = t;

}

// Обнуление полей

void Lesson::NullLess()

{

this->Name = "";

this->Teach.SetTeach("", "", "");

}

// Фун-ии получения данных из полей

string Lesson::GetName()

{

return this->Name;

}

Teacher Lesson::GetTeach()

{

return this->Teach;

}

// Ввод названия урока

void Lesson::InpLessName(Teacher t)

{

cout << "Введите название предмета: ";

cin >> Name;

cout << endl;

Teach = t;

}

// Вывод уроков, передаем адрес, где будет храниться информация

void Lesson::out\_Less()

{

cout << "Название предмета: " << Name << endl;

Teach.DisplayTeach();

}

// Деструктор

Lesson::~Lesson()

{

;

}

Файл Mark.h

#pragma once

#include "Lesson.h"

// Класс Оценка

class Mark

{

private:

int Value = 0; // Оценка

public:

// Конструктор

Mark();

// Деструктор

~Mark();

// Инициализация структуры Оценка

void SetMark(int v);

// Получение оценки

int GetMark();

// Ввод данных в структуру Оценка

void InpMark();

// Вывод данных из структуры

void DispMark();

};

Файл Mark.cpp

#include "Mark.h"

// Конструктор

Mark::Mark()

{

Value = 0;

}

// Инициализация структуры Оценка

void Mark::SetMark(int v)

{

Value = v;

}

// Получение оценки

int Mark::GetMark()

{

return Value;

}

// Ввод данных в структуру Оценка

void Mark::InpMark()

{

string mark;

cout << "Введите оценку: ";

cin >> mark;

Value = stoi(mark);

cout << endl;

}

// Вывод данных из структуры

void Mark::DispMark()

{

cout << "Оценка: " << Value << endl;

}

// Деструктор

Mark::~Mark()

{

;

}

Файл Student.h

#pragma once

#include "Mark.h"

// Класс Ученик

class Student

{

private:

string Fam = ""; // Фамилия

string Name = ""; // Имя

string Otch = ""; // Отчество

Mark Marks[20]; // Массив оценок

Lesson Lessons[20]; // Массив уроков

int N = 0; // Текущее кол-во уроков

public:

// Конструктор

Student();

// Деструктор

~Student();

// Функции получения значений полей

string GetFam();

string GetName();

string GetOtch();

// Возвращает оценку по индексу в массиве

Mark GetMarkI(int i);

// Возвращает урок по индексу в массиве

Lesson GetLessI(int i);

// Очищение всех полей структуры Ученик (Student)

void NullStud();

// Инициализация структуры Ученик

void SetStud(string Fam, string Name, string Otch, Mark mas\_m[], int LenMark, Lesson mas\_l[], int LenLess);

// Ввод ФИО ученика в структуру

void InpStudFIO();

// Добавление к ученику урока

bool AddLessToStud(Lesson ls);

// Добавление к ученику урока и оценки

bool AddLMtoStud(Lesson ls, Mark m);

// Добавление к ученику оценки к конкретному предмету

bool AddMarkToStud(Lesson ls, Mark m);

// Вывод ученика с предметами, которые у него есть

void DispStud();

// Вывод ученика с всеми предметами, даже пустыми

void DispFullStud();

};

Файл Student.cpp

#include "Student.h"

// Конструктор

Student::Student()

{

Fam = "";

Name = "";

Otch = "";

Lesson L;

Mark M;

L.NullLess();

M.SetMark(0);

N = 0;

for (int i = 0; i < 20; i++)

{

Lessons[i] = L;

}

for (int i = 0; i < 20; i++)

{

Marks[i] = M;

}

}

// Очищение всех полей структуры Ученик (Student)

void Student::NullStud()

{

Fam = "";

Name = "";

Otch = "";

Lesson L;

Mark M;

L.NullLess();

M.SetMark(0);

this->N = 0;

for (int i = 0; i < 20; i++)

{

Lessons[i] = L;

}

for (int i = 0; i < 20; i++)

{

Marks[i] = M;

}

}

// Функции получения значений полей

string Student::GetFam()

{

return this->Fam;

}

string Student::GetName()

{

return this->Name;

}

string Student::GetOtch()

{

return this->Otch;

}

Mark Student::GetMarkI(int i)

{

Mark m;

if (i < 20 && i > -1)

m = Marks[i];

return m;

}

Lesson Student::GetLessI(int i)

{

Lesson l;

if (i < 20 && i > -1)

l = Lessons[i];

return l;

}

// Инициализация структуры Ученик

void Student::SetStud(string Fam, string Name, string Otch, Mark mas\_m[], int LenMark, Lesson mas\_l[], int LenLess)

{

this->Fam = Fam;

this->Name = Name;

this->Otch = Otch;

this->N = 0;

int i = 0;

for ( i = 0; i < LenLess && i < 20; i++)

{

Lessons[i] = mas\_l[i];

this->N++;

}

for (int i = 0; i < LenMark && i < 20; i++)

{

Marks[i] = mas\_m[i];

}

}

// Ввод ФИО ученика в структуру

void Student::InpStudFIO()

{

cout << "Введите Фамилию ученика: ";

cin >> Fam;

cout << "Введите Имя ученика: ";

cin >> Name;

cout << "Введите Отчество ученика: ";

cin >> Otch;

}

// Добавление к ученику урока

// 0 - не удачно, 1 - удачно

bool Student::AddLessToStud(Lesson ls)

{

bool fl = false;

for (int i = 0; (i < 20) && (fl == 0); i++)

{

if (Lessons[i].GetName() == "")

{

Lessons[i] = ls;

fl = true;

}

}

return fl;

}

// Добавление к ученику урока и оценки

// 0 - не удачно, 1 - удачно

bool Student::AddLMtoStud(Lesson ls, Mark m)

{

bool fl = false;

for (int i = 0; (i < 20) && (fl == 0); i++)

{

if (Lessons[i].GetName() == "")

{

Lessons[i] = ls;

Marks[i] = m;

fl = true;

}

}

return fl;

}

// Добавление к ученику оценки к конкретному предмету

// 0 - не удачно, 1 - удачно

bool Student::AddMarkToStud(Lesson ls, Mark m)

{

bool fl = false;

for (int i = 0; (i < 20) && (fl == 0); i++)

{

if (Lessons[i].GetName() == ls.GetName())

{

if (Lessons[i].GetTeach().GetFam() == ls.GetTeach().GetFam())

{

if (Lessons[i].GetTeach().GetName() == ls.GetTeach().GetName())

{

if (Lessons[i].GetTeach().GetOtch() == ls.GetTeach().GetOtch())

{

Marks[i] = m;

fl = true;

}

}

}

}

}

return fl;

}

// Вывод ученика

// Передаем ученика и флаг для различного вывода уроков и оценок

// fl\_out (0 или 1)

void Student::DispStud()

{

cout << "ФИО ученика: " << Fam << " " << Name << " " << Otch << endl;

cout << "Уроки:" << endl;

int i = 0;

while (Lessons[i].GetName() != "")

{

cout << " \* " << Lessons[i].GetName() << " \* " << Marks[i].GetMark() << " \* " << endl;

i++;

}

}

// Вывод ученика с всеми предметами, даже пустыми

void Student::DispFullStud()

{

cout << "ФИО ученика: " << Fam << " " << Name << " " << Otch << endl;

cout << "Уроки:" << endl;

for (int i = 0; i < 20; i++)

{

Lessons[i].out\_Less();

}

cout << "Оценки:" << endl;

for (int i = 0; i < 20; i++)

{

Marks[i];

}

}

// Деструктор

Student::~Student()

{

;

}

Файл Class.h

#pragma once

#include "Student.h"

// Класс Класс

class MyClass

{

private:

string Name = ""; // Название класса

string Year = ""; // Год обучения

int N = 0; // Текущее кол-во учеников

Student Students[32]; // Массив учеников

public:

// Конструктор

MyClass();

// Деструктор

~MyClass();

// Очищение полей структуры Класс

void NullMyClass();

// Инициализация структуры Класс

void SetMyClass(string Name, string Year, Student mas[], int LenStud);

// Функции получения значений полей

string GetName();

string GetYear();

// Ввод данных в структуру Класс

void InpNameYearMyClass();

// Добавление к классу ученика

bool AddStudToMyClass(Student st);

// Выод данных из структуры Класс

void DispMyClass();

// Выод всех данных из структуры Класс

void DispFullMyClass();

// Функция по нахождению лучших учеников в классе

bool BestStudInMyClass();

// Функция по нахождению худших учеников в классе

bool BedStudInMyClass();

};

Файл Class.cpp

#include "Class.h"

// Конструктор

MyClass::MyClass()

{

Name = "";

Year = "";

N = 0;

Student S;

S.NullStud();

for (int i = 0; i < 32; i++)

{

Students[i] = S;

}

}

// Очищение полей структуры Класс

void MyClass::NullMyClass()

{

Name = "";

Year = "";

N = 0;

Student S;

S.NullStud();

for (int i = 0; i < 32; i++)

{

Students[i] = S;

}

}

// Инициализация структуры Класс

void MyClass::SetMyClass(string Name, string Year, Student mas[], int LenStud)

{

this->Name = Name;

this->Year = Year;

this->N = 0;

Student S;

S.NullStud();

for (int i = 0; i < 32 && i < LenStud; i++)

{

Students[i] = mas[i];

}

}

// Функции получения значений полей

string MyClass::GetName()

{

return Name;

}

string MyClass::GetYear()

{

return Year;

}

// Ввод данных в структуру Класс

void MyClass::InpNameYearMyClass()

{

cout << "Введите название класса: ";

cin >> Name;

cout << endl;

cout << "Введите год обучения: ";

cin >> Year;

cout << endl;

}

// Добавление к классу ченика

bool MyClass::AddStudToMyClass(Student st)

{

bool fl = false;

if (N < 20)

{

Students[N] = st;

N++;

fl = true;

}

return fl;

}

// Вывод данных из структуры Класс

// fl\_out (0 или 1)

void MyClass::DispMyClass()

{

cout << "Название класса: " << Name << endl;;

cout << "Год обучения: " << Year << endl;

cout << endl;

cout << "Ученики:" << endl;

int i = 0;

while (Students[i].GetFam() != "")

{

Students[i].DispStud();

i++;

}

}

// Выод всех данных из структуры Класс

void MyClass::DispFullMyClass()

{

cout << "Название класса: " << Name << endl;;

cout << "Год обучения: " << Year << endl;

cout << endl;

cout << "Ученики:" << endl;

int i = 0;

while (Students[i].GetFam() != "")

{

Students[i].DispFullStud();

i++;

}

}

// Функция по нахождению лучших учеников в классе

bool MyClass::BestStudInMyClass()

{

double mas\_m[32];

bool exit = false;

for (int i = 0; i < 32; i++)

{

mas\_m[i] = 0;

}

int i = 0;

int k = 0;

int sum = 0;

while (Students[i].GetFam() != "")

{

int j = 0;

k = 0;

sum = 0;

while (Students[i].GetLessI(j).GetName() != "")

{

sum += Students[i].GetMarkI(j).GetMark();

k++;

j++;

}

mas\_m[i] = double(sum) / k;

i++;

}

if (i > 0)

{

double maxM = 0;

maxM = mas\_m[0];

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

{

if (maxM < mas\_m[j])

maxM = mas\_m[j];

}

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

{

if (mas\_m[j] == maxM)

Students[j].DispStud();

}

exit = true;

}

return exit;

}

// Функция по нахождению худших учеников в классе

bool MyClass::BedStudInMyClass()

{

double mas\_m[32];

bool exit = false;

for (int i = 0; i < 32; i++)

{

mas\_m[i] = 0;

}

int i = 0;

int k = 0;

int sum = 0;

while (Students[i].GetFam() != "")

{

int j = 0;

k = 0;

sum = 0;

while (Students[i].GetLessI(j).GetName() != "")

{

sum += Students[i].GetMarkI(j).GetMark();

k++;

j++;

}

mas\_m[i] = double(sum) / k;

i++;

}

if (i > 0)

{

double minM = 0;

minM = mas\_m[0];

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

{

if (minM > mas\_m[j])

minM = mas\_m[j];

}

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

{

if (mas\_m[j] == minM)

Students[j].DispStud();

}

exit = true;

}

return exit;

}

// Деструктор

MyClass::~MyClass()

{

;

}

Файл lab2\_t1.cpp

#include "Class.h"

int main()

{

setlocale(LC\_ALL, "Russian");

system("chcp 1251");

system("cls");

cout << "========================Тест класса Учитель===========================" << endl;

Teacher t\_Chem;

Teacher\* t\_Biology = new Teacher;

t\_Chem.SetTeach("", "", "");

t\_Biology->SetTeach("", "", "");

cout << "Введите ФИО 2-ух учителей:" << endl;

cout << endl;

t\_Chem.InpTeachFIO();

t\_Chem.DisplayTeach();

cout << endl;

t\_Biology->InpTeachFIO();

(\*t\_Biology).DisplayTeach();

cout << "===================================================" << endl;

cout << endl;

cout << endl;

cout << "========================Тест класса Уроки===========================" << endl;

Lesson Chem;

Lesson\* Biology = new Lesson;

Chem.NullLess();

(\*Biology).NullLess();

cout << "Введите 2 урока:" << endl;

cout << endl;

cout << "Для учителя - ";

t\_Chem.DisplayTeach();

Chem.InpLessName(t\_Chem);

cout << "Вывод:" << endl;

Chem.out\_Less();

cout << endl;

cout << "Для учителя - ";

(\*t\_Biology).DisplayTeach();

Biology->InpLessName(\*t\_Biology);

cout << "Вывод:" << endl;

Biology->out\_Less();

cout << "===================================================" << endl;

cout << endl;

cout << endl;

cout << "=======================Тест класса Ученик и Оценки============================" << endl;

Student st1;

Student\* st2 = new Student;

Student st3;

st1.NullStud();

(\*st2).NullStud();

st3.NullStud();

cout << "Введите 3 ученика:" << endl;

cout << endl;

cout << "Первый ученик:" << endl;

st1.InpStudFIO();

cout << endl;

cout << "Выберите, какие уроки у первого ученика:" << endl;

cout << endl;

Chem.out\_Less();

string str = "";

cout << "Добавить этот предмет ученику? (д - добавить, другой набор символов - нет): ";

cin >> str;

if (str == "д")

{

if (st1.AddLessToStud(Chem))

{

cout << "Урок добавлен!" << endl;

cout << endl;

cout << "Добавить оценку (1,2,3,4,5 - добавить, другой набор символов - нет):" << endl;

str = "";

cin >> str;

Mark mr;

mr.SetMark(0);

mr.SetMark(stoi(str));

if ((mr.GetMark() >= 1) || (mr.GetMark() <= 5))

{

if (st1.AddMarkToStud(Chem, mr))

cout << "Оценка добавлена!" << endl;

else

cout << "Оценка не добавлена!" << endl;

}

}

else

cout << "Урок не добавлен!" << endl;

}

else

cout << "Урок не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод:" << endl;

st1.DispStud();

cout << endl;

cout << "Выберите, какие уроки у первого ученика:" << endl;

cout << endl;

Biology->out\_Less();

str = "";

cout << "Добавить этот предмет ученику? (д - добавить, другой набор символов - нет): ";

cin >> str;

if (str == "д")

{

if (st1.AddLessToStud(\*Biology))

{

cout << "Урок добавлен!" << endl;

cout << endl;

cout << "Добавить оценку (1,2,3,4,5 - добавить, другой набор символов - нет):" << endl;

str = "";

cin >> str;

Mark mr;

mr.SetMark(0);

mr.SetMark(stoi(str));

if ((mr.GetMark() >= 1) || (mr.GetMark() <= 5))

{

if (st1.AddMarkToStud(\*Biology, mr))

cout << "Оценка добавлена!" << endl;

else

cout << "Оценка не добавлена!" << endl;

}

}

else

cout << "Урок не добавлен!" << endl;

}

else

cout << "Урок не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод:" << endl;

st1.DispStud();

cout << endl;

cout << "Второй ученик:" << endl;

st2->InpStudFIO();

cout << endl;

cout << "Выберите, какие уроки у первого ученика:" << endl;

Chem.out\_Less();

str = "";

cout << "Добавить этот предмет ученику? (д - добавить, другой набор символов - нет): ";

cin >> str;

if (str == "д")

{

if (st2->AddLessToStud(Chem))

{

cout << "Урок добавлен!" << endl;

cout << endl;

cout << "Добавить оценку (1,2,3,4,5 - добавить, другой набор символов - нет):" << endl;

str = "";

cin >> str;

Mark mr;

mr.SetMark(0);

mr.SetMark(stoi(str));

if ((mr.GetMark() >= 1) || (mr.GetMark() <= 5))

{

if ((\*st2).AddMarkToStud(Chem, mr))

cout << "Оценка добавлена!" << endl;

else

cout << "Оценка не добавлена!" << endl;

}

}

else

cout << "Урок не добавлен!" << endl;

}

else

cout << "Урок не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод:" << endl;

(\*st2).DispStud();

cout << endl;

cout << "Выберите, какие уроки у первого ученика:" << endl;

cout << endl;

(\*Biology).out\_Less();

str = "";

cout << "Добавить этот предмет ученику? (д - добавить, другой набор символов - нет): ";

cin >> str;

if (str == "д")

{

if (st2->AddLessToStud(\*Biology))

{

cout << "Урок добавлен!" << endl;

cout << endl;

cout << "Добавить оценку (1,2,3,4,5 - добавить, другой набор символов - нет):" << endl;

str = "";

cin >> str;

Mark mr;

mr.SetMark(0);

mr.SetMark(stoi(str));

if ((mr.GetMark() >= 1) || (mr.GetMark() <= 5))

{

if (st2->AddMarkToStud(\*Biology, mr))

cout << "Оценка добавлена!" << endl;

else

cout << "Оценка не добавлена!" << endl;

}

}

else

cout << "Урок не добавлен!" << endl;

}

else

cout << "Урок не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод:" << endl;

st2->DispStud();

cout << endl;

cout << "Третий ученик:" << endl;

st3.InpStudFIO();

cout << endl;

cout << "Выберите, какие уроки у первого ученика:" << endl;

Chem.out\_Less();

str = "";

cout << "Добавить этот предмет ученику? (д - добавить, другой набор символов - нет): ";

cin >> str;

if (str == "д")

{

if (st3.AddLessToStud(Chem))

{

cout << "Урок добавлен!" << endl;

cout << endl;

cout << "Добавить оценку (1,2,3,4,5 - добавить, другой набор символов - нет):" << endl;

str = "";

cin >> str;

Mark mr;

mr.SetMark(0);

mr.SetMark(stoi(str));

if ((mr.GetMark() >= 1) || (mr.GetMark() <= 5))

{

if (st3.AddMarkToStud(Chem, mr))

cout << "Оценка добавлена!" << endl;

else

cout << "Оценка не добавлена!" << endl;

}

}

else

cout << "Урок не добавлен!" << endl;

}

else

cout << "Урок не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод:" << endl;

st3.DispStud();

cout << endl;

cout << "Выберите, какие уроки у первого ученика:" << endl;

cout << endl;

Biology->out\_Less();

str = "";

cout << "Добавить этот предмет ученику? (д - добавить, другой набор символов - нет): ";

cin >> str;

if (str == "д")

{

if (st3.AddLessToStud(\*Biology))

{

cout << "Урок добавлен!" << endl;

cout << endl;

cout << "Добавить оценку (1,2,3,4,5 - добавить, другой набор символов - нет):" << endl;

str = "";

cin >> str;

Mark mr;

mr.SetMark(0);

mr.SetMark(stoi(str));

if ((mr.GetMark() >= 1) || (mr.GetMark() <= 5))

{

if (st3.AddMarkToStud(\*Biology, mr))

cout << "Оценка добавлена!" << endl;

else

cout << "Оценка не добавлена!" << endl;

}

}

else

cout << "Урок не добавлен!" << endl;

}

else

cout << "Урок не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод:" << endl;

st3.DispStud();

cout << "===================================================" << endl;

cout << endl;

cout << endl;

cout << "=======================Тест класса Класс============================" << endl;

cout << "Создайте 2 класса учеников:" << endl;

cout << endl;

MyClass cl1;

MyClass\* cl2 = new MyClass;

cl1.NullMyClass();

(\*cl2).NullMyClass();

cout << "Введите данные для первого класса" << endl;

cl1.InpNameYearMyClass();

cout << endl;

st1.DispStud();

cout << "Добавить данного ученика в класс? (д - добавить, другой набор символов - нет): ";

str = "";

cin >> str;

if (str == "д")

{

if (cl1.AddStudToMyClass(st1))

{

cout << "Ученик добавлен!" << endl;

cout << endl;

}

else

cout << "Ученик не добавлен!" << endl;

}

else

cout << "Ученик не добавлен!" << endl;

cout << endl;

cout << endl;

st2->DispStud();

cout << "Добавить данного ученика в класс? (д - добавить, другой набор символов - нет): ";

str = "";

cin >> str;

if (str == "д")

{

if (cl1.AddStudToMyClass(\*st2))

{

cout << "Ученик добавлен!" << endl;

cout << endl;

}

else

cout << "Ученик не добавлен!" << endl;

}

else

cout << "Ученик не добавлен!" << endl;

cout << endl;

cout << endl;

st3.DispStud();

cout << "Добавить данного ученика в класс? (д - добавить, другой набор символов - нет): ";

str = "";

cin >> str;

if (str == "д")

{

if (cl1.AddStudToMyClass(st3))

{

cout << "Ученик добавлен!" << endl;

cout << endl;

}

else

cout << "Ученик не добавлен!" << endl;

}

else

cout << "Ученик не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод класаа №1" << endl;

cl1.DispMyClass();

cout << endl;

cout << endl;

cout << endl;

cout << "Введите данные для второго класса" << endl;

cl2->InpNameYearMyClass();

cout << endl;

st1.DispStud();

cout << "Добавить данного ученика в класс? (д - добавить, другой набор символов - нет): ";

str = "";

cin >> str;

if (str == "д")

{

if (cl2->AddStudToMyClass(st1))

{

cout << "Ученик добавлен!" << endl;

cout << endl;

}

else

cout << "Ученик не добавлен!" << endl;

}

else

cout << "Ученик не добавлен!" << endl;

cout << endl;

cout << endl;

st2->DispStud();

cout << "Добавить данного ученика в класс? (д - добавить, другой набор символов - нет): ";

str = "";

cin >> str;

if (str == "д")

{

if ((\*cl2).AddStudToMyClass(\*st2))

{

cout << "Ученик добавлен!" << endl;

cout << endl;

}

else

cout << "Ученик не добавлен!" << endl;

}

else

cout << "Ученик не добавлен!" << endl;

cout << endl;

cout << endl;

st3.DispStud();

cout << "Добавить данного ученика в класс? (д - добавить, другой набор символов - нет): ";

str = "";

cin >> str;

if (str == "д")

{

if ((\*cl2).AddStudToMyClass(st3))

{

cout << "Ученик добавлен!" << endl;

cout << endl;

}

else

cout << "Ученик не добавлен!" << endl;

}

else

cout << "Ученик не добавлен!" << endl;

cout << endl;

cout << endl;

cout << "Вывод класаа №2" << endl;

cout << endl;

(\*cl2).DispMyClass();

cout << "===================================================" << endl;

cout << endl;

cout << endl;

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция поиска лучших учеников в классе\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Лучшие ученики класса №1:" << endl;

cl1.BestStudInMyClass();

cout << endl;

cout << endl;

cout << "Лучшие ученики класса №2:" << endl;

(\*cl2).BestStudInMyClass();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

cout << endl;

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция поиска худших учеников в классе\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Худшие ученики класса №1:" << endl;

cl1.BedStudInMyClass();

cout << endl;

cout << endl;

cout << "Худшие ученики класса №2:" << endl;

cl2->BedStudInMyClass();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

cout << endl;

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Динамический массив объектов класса Ученик\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

Student\* mas\_s = new Student[3];

mas\_s[0] = st1;

mas\_s[1] = \*st2;

mas\_s[2] = st3;

for (int i = 0; i < 3; i++)

mas\_s[i].DispStud();

Mark m1;

m1.SetMark(5);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

cout << endl;

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Массив динамических объектов класса Ученик\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

Mark mas\_M[20];

Lesson mas\_L[20];

string Fam, Name, Otch;

Teacher teach;

for (int i = 0; i < 5; i++)

{

Fam = "Иванова" + to\_string(i);

Name = "Марина" + to\_string(i);

Otch = "Васильвна" + to\_string(i);

teach.SetTeach(Fam, Name, Otch);

mas\_M[i].SetMark(4 + i % 2);

mas\_L[i].SetLess("Урок" + to\_string(i), teach);

}

Student\* mas\_s2[5];

for (int i = 0; i < 5; i++)

{

mas\_s2[i] = new Student;

mas\_s2[i]->SetStud("Сидоров" + to\_string(i), "Николай" + to\_string(i), "Генадьевич" + to\_string(i), mas\_M, 20, mas\_L, 20);

}

for (int i = 0; i < 5; i++)

mas\_s2[i]->DispStud();

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

system("Pause");

delete t\_Biology;

delete Biology;

delete st2;

delete cl2;

delete[] mas\_s;

}

Тесты программы:













