Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Отчет

По лабораторной работе №8

**«(Статические поля и методы)»**

по дисциплине «Программирование - 3семестр»

Студент группы ПИ-02 Гранкин А.А.

Преподаватель Троицкий В.С.

Барнаул 2021

Задание:

Модифицировать ваши проекты на С++, C# и Java путем добавления в один из классов как минимум одного статического поля и одного статического метода.

1) Придумать для чего в вашем классе может быть разумным использование статического поля и статического метода;

2) Создать эти статические поля и методы;

3) Продемонстрировать их использование;

!) Git обязателен;

!) В отчете по ЛР должна быть ссылка на Githab.

Решение:

C++

Класс Mark дополнен двухсвязным списком, который использует статическое поле lastMark и статические методы NewList() и reprint().

static Mark\* lastMark; // Адрес последнего элемента списка

Mark\* prev; // На предыдущий элемент списка

Mark\* next; // На следующий элемент списка

// Добавление элемента в конец списка

void Mark::Add(void)

{

if (lastMark == NULL)

this->prev = NULL;

else

{

lastMark->next = this;

prev = lastMark;

}

lastMark = this;

this->next = NULL;

}

// Вывод на дисплей содержимого списка

void Mark::reprint(void)

{

Mark\* uk; // Вспомогательный указатель

uk = lastMark;

if (uk == NULL)

{

cout << "Список пуст!";

return;

}

else

cout << "\nСодержимое списка:\n";

// Цикл печати в обратном порядке значений элементов списка:

while (uk != NULL)

{

cout << uk->value << '\t';

uk = uk->prev;

}

}

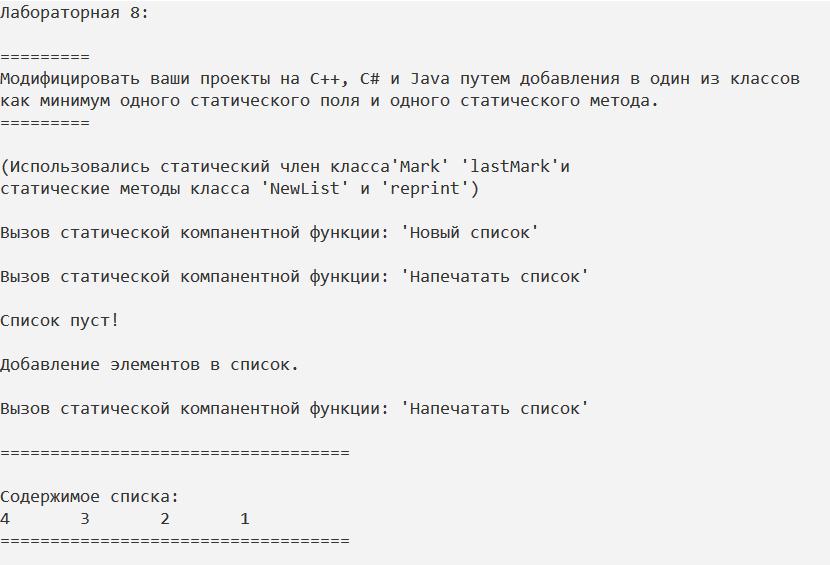
// Создание нового списка

void Mark::NewList()

{

lastMark = NULL;

}



C#

Класс Teacher дополнен двухсвязным списком, который использует статическое поле lastTeacher и статические методы NewList() и reprint().

private static Teacher lastTeacher;

private Teacher prev;

private Teacher next;

// Новый список

public static void NewList()

{

lastTeacher = null;

}

// Добавление элемента в конец списка

public void Add()

{

if (lastTeacher == null)

this.prev = null;

else

{

lastTeacher.next = this;

prev = lastTeacher;

}

lastTeacher = this;

this.next = null;

}

// Вывод на дисплей содержимого списка

public static void reprint()

{

Teacher uk; // Вспомогательная ссылка

uk = lastTeacher;

if (uk == null)

{

Console.WriteLine("Список пуст!");

return;

}

else

Console.WriteLine("\nСодержимое списка:\n");

// Цикл печати в обратном порядке значений элементов списка:

while (uk != null)

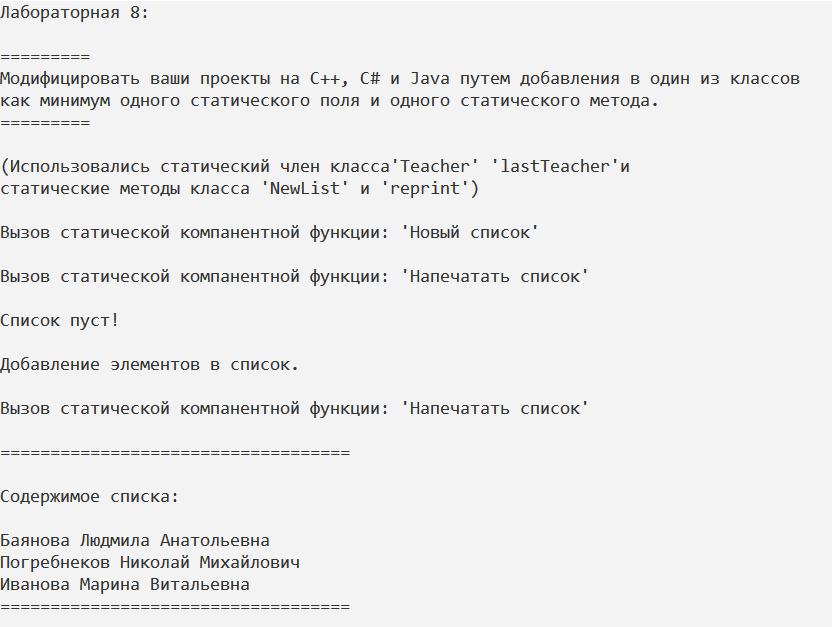
{

Console.WriteLine(uk.GetFam() + " " + uk.GetName() + " " + uk.GetOtch() + " " + "\t");

uk = uk.prev;

}

}



Java

Класс Mark дополнен двухсвязным списком, который использует статическое поле lastMark и статические методы NewList() и reprint().

**private** **static** Mark *lastMark*;

**private** Mark prev;

**private** Mark next;

// Новый список

**public** **static** **void** NewList()

{

*lastMark* = **null**;

}

// Добавление элемента в конец списка

**public** **void** Add()

{

**if** (*lastMark* == **null**)

**this**.prev = **null**;

**else**

{

*lastMark*.next = **this**;

prev = *lastMark*;

}

*lastMark* = **this**;

**this**.next = **null**;

}

// Вывод на дисплей содержимого списка

**public** **static** **void** reprint()

{

Mark uk; // Вспомогательная ссылка

uk = *lastMark*;

**if** (uk == **null**)

{

System.***out***.println("Список пуст!");

**return**;

}

**else**

System.***out***.println("\nСодержимое списка:\n");

// Цикл печати в обратном порядке значений элементов списка:

**while** (uk != **null**)

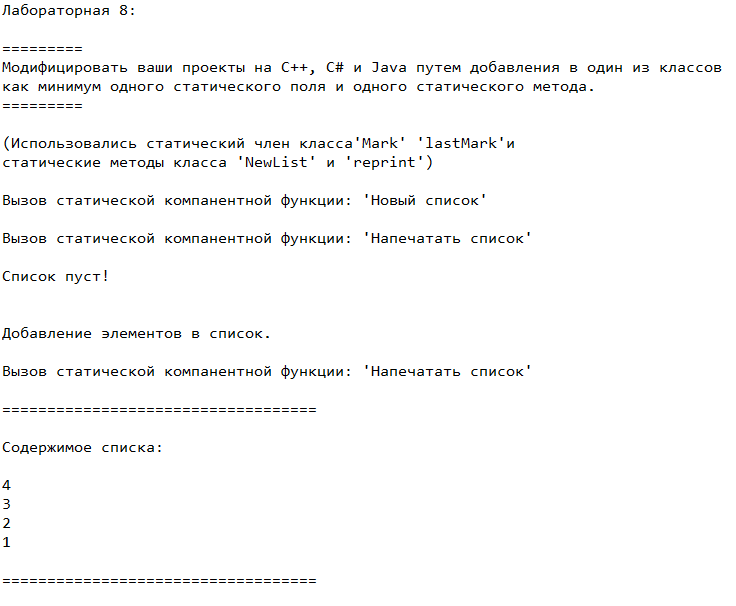
{

System.***out***.println(uk.value + "\t");

uk = uk.prev;

}

}



Приложение А. Тексты программ.

C++

Файл Teacher.h

#pragma once

#include <iostream>

#include <stdio.h>

#include <string>

#include <locale.h>

using namespace std;

class Student;

// 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();

// Фун-ии получения данных из полей через ссылку

void GetFamLink(string &Fam);

void GetNameLink(string &Name);

void GetOtchLink(string &Otch);

// Деструктор

~Teacher();

// Инициализация структуры Учитель

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

// Ввод учителя

void InpFIO();

// Ввод учителя на экран

void DisplayInfo();

// Дружественная функция обмена ФИО классов Teacher и Student

friend void NameExchange(Teacher &teacher, Student &student);

// Обработка строк

void AdditionSapces();

};

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;

}

// Фун-ии получения данных из полей через ссылку

void Teacher::GetFamLink(string& Fam)

{

Fam = this->fam;

}

void Teacher::GetNameLink(string& Name)

{

Name = this->name;

}

void Teacher::GetOtchLink(string& Otch)

{

Otch = this->otch;

}

// Инициализация класса Учитель

void Teacher::Set(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::InpFIO()

{

cout << "Введите Фамилию учителя: ";

cin >> fam;

cout << "Введите имя учителя: ";

cin >> name;

cout << "Введите отчество учителя: ";

cin >> otch;

}

// Ввод учителя

void Teacher::DisplayInfo()

{

cout << "ФИО учителя: " << fam << " " << name << " " << otch << endl;

}

// Обработка строк

void Teacher::AdditionSapces()

{

fam = fam + " ";

name = name + " ";

otch = otch + " ";

}

// Деструктор

Teacher::~Teacher()

{

;

}

Файл Lesson.h

#pragma once

#include "Teacher.h"

// Класс Уроки

class Lesson

{

private:

string nameLesson = ""; // Название предмета

Teacher teacher; // Учитель

public:

// Конструктор

Lesson();

// Фун-ии получения данных из полей

string GetNameLess();

Teacher GetTeacher();

// Деструктор

~Lesson();

// Инициализация класса Уроки

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

// Обнуление полей

void Null();

// Ввод названия урока

void InpInConsol(Teacher t);

// Вывод уроков

void DisplayInfo();

};

Файл Lesson.cpp

#include "Lesson.h"

// Конструктор

Lesson::Lesson()

{

this->nameLesson = "";

this->teacher.Set("", "", "");

}

// Инициализация класса Уроки

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

{

nameLesson = Name\_Less;

teacher = t;

}

// Обнуление полей

void Lesson::Null()

{

this->nameLesson = "";

this->teacher.Set("", "", "");

}

// Фун-ии получения данных из полей

string Lesson::GetNameLess()

{

return this->nameLesson;

}

Teacher Lesson::GetTeacher()

{

return this->teacher;

}

// Ввод названия урока

void Lesson::InpInConsol(Teacher t)

{

cout << "Введите название предмета: ";

cin >> nameLesson;

cout << endl;

teacher = t;

}

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

void Lesson::DisplayInfo()

{

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

teacher.DisplayInfo();

}

// Деструктор

Lesson::~Lesson()

{

;

}

Файл Mark.h

#pragma once

#include "Lesson.h"

// Класс Оценка

class Mark

{

private:

int value = 0; // Оценка

static Mark\* lastMark; // Адрес последнего элемента списка

Mark\* prev; // На предыдущий элемент списка

Mark\* next; // На следующий элемент списка

public:

// Конструктор

Mark();

// Конструктор

Mark(int value);

// Деструктор

~Mark();

// Инициализация структуры Оценка

void Set(int v);

// Получение оценки

int Get();

// Получение оценки через указатель

void GetPoint(int\* Value);

// Ввод данных в структуру Оценка

void InpMark();

// Вывод данных из структуры

void DisplayValue();

// Добавление элемента в конец списка

void Add(void);

// Вывод на дисплей содержимого списка

static void reprint(void);

// Создание нового списка

static void NewList();

// Перегрузка оператора '+' (Mark + int)

friend Mark operator+(const Mark& mark, int value);

// Перегрузка префиксного оператора '++'

Mark& operator++();

// Перегрузка постфиксного оператора '++'

Mark operator++ (int);

};

Файл Mark.cpp

#include "Mark.h"

// Конструктор

Mark::Mark()

{

value = 0;

}

// Конструктор

Mark::Mark(int value)

{

this->value = value;

}

// Инициализация структуры Оценка

void Mark::Set(int v)

{

value = v;

}

// Получение оценки

int Mark::Get()

{

return this->value;

}

// Получение оценки через указатель

void Mark::GetPoint(int\* Value)

{

\*Value = 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::DisplayValue()

{

cout << "Оценка: " << value << endl;

}

// Добавление элемента в конец списка

void Mark::Add(void)

{

if (lastMark == NULL)

this->prev = NULL;

else

{

lastMark->next = this;

prev = lastMark;

}

lastMark = this;

this->next = NULL;

}

// Вывод на дисплей содержимого списка

void Mark::reprint(void)

{

Mark\* uk; // Вспомогательный указатель

uk = lastMark;

if (uk == NULL)

{

cout << "Список пуст!";

return;

}

else

cout << "\nСодержимое списка:\n";

// Цикл печати в обратном порядке значений элементов списка:

while (uk != NULL)

{

cout << uk->value << '\t';

uk = uk->prev;

}

}

// Создание нового списка

void Mark::NewList()

{

lastMark = NULL;

}

// Деструктор

Mark::~Mark()

{

;

}

// Выполняем Mark + int

Mark operator+(const Mark& mark, int value)

{

if (mark.value + value > 5)

return mark;

else

return Mark(mark.value + value);

}

// Перегрузка префиксного оператора '++'

Mark& Mark::operator++()

{

value += 1;

return \*this;

}

// Перегрузка постфиксного оператора '++'

Mark Mark::operator++ (int)

{

Mark m1 = \*this;

++\* this;

return m1;

}

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 countLess = 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 GetMarkByNumber(int i);

// Возвращает урок по индексу в массиве

Lesson GetLessByNumber(int i);

// Очищение всех полей структуры Ученик (Student)

void Null();

// Инициализация структуры Ученик

void Set(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 AddLess(Lesson ls);

// Добавление к ученику урока и оценки

bool AddLessAndMark(Lesson ls, Mark m);

// Добавление к ученику оценки к конкретному предмету

bool AddMark(Lesson ls, Mark m);

// Вывод ученика с предметами, которые у него есть

void DisplayShortInfo();

// Вывод ученика с всеми предметами, даже пустыми

void DispFullInfo();

// Дружественная функция обмена ФИО классов Teacher и Student

friend void NameExchange(Teacher& teacher, Student& student);

};

Файл Student.cpp

#include "Student.h"

// Конструктор

Student::Student()

{

fam = "";

name = "";

otch = "";

Lesson L;

Mark M;

L.Null();

M.Set(0);

countLess = 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::Null()

{

fam = "";

name = "";

otch = "";

Lesson L;

Mark M;

L.Null();

M.Set(0);

this->countLess = 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::GetMarkByNumber(int i)

{

Mark m;

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

m = marks[i];

return m;

}

Lesson Student::GetLessByNumber(int i)

{

Lesson l;

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

l = lessons[i];

return l;

}

// Инициализация структуры Ученик

void Student::Set(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->countLess = 0;

int i = 0;

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

{

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

this->countLess++;

}

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::AddLess(Lesson ls)

{

bool fl = false;

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

{

if (lessons[i].GetNameLess() == "")

{

lessons[i] = ls;

fl = true;

}

}

return fl;

}

// Добавление к ученику урока и оценки

// 0 - не удачно, 1 - удачно

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

{

bool fl = false;

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

{

if (lessons[i].GetNameLess() == "")

{

lessons[i] = ls;

marks[i] = m;

fl = true;

}

}

return fl;

}

// Добавление к ученику оценки к конкретному предмету

// 0 - не удачно, 1 - удачно

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

{

bool fl = false;

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

{

if (lessons[i].GetNameLess() == ls.GetNameLess())

{

if (lessons[i].GetTeacher().GetFam() == ls.GetTeacher().GetFam())

{

if (lessons[i].GetTeacher().GetName() == ls.GetTeacher().GetName())

{

if (lessons[i].GetTeacher().GetOtch() == ls.GetTeacher().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::DisplayShortInfo()

{

cout << "ФИО ученика: " << fam << " " << name << " " << otch << endl;

cout << "Уроки:" << endl;

int i = 0;

while (lessons[i].GetNameLess() != "")

{

cout << " \* " << lessons[i].GetNameLess() << " \* " << marks[i].Get() << " \* " << endl;

i++;

}

}

// Вывод ученика с всеми предметами, даже пустыми

void Student::DispFullInfo()

{

cout << "ФИО ученика: " << fam << " " << name << " " << otch << endl;

cout << "Уроки:" << endl;

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

{

lessons[i].DisplayInfo();

}

cout << "Оценки:" << endl;

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

{

marks[i].DisplayValue();

}

}

// Деструктор

Student::~Student()

{

;

}

// Дружественная функция обмена ФИО классов Teacher и Student

void NameExchange(Teacher& teacher, Student& student)

{

string fam, name, otch;

fam = teacher.fam;

teacher.fam = student.fam;

student.fam = fam;

name = teacher.name;

teacher.name = student.name;

student.name = name;

otch = teacher.otch;

teacher.otch = student.otch;

student.otch = otch;

}

Файл Class.h

#pragma once

#include "Student.h"

// Класс Класс

class MyClass

{

private:

string className = ""; // Название класса

string yearOfStudy = ""; // Год обучения

int contStuds = 0; // Текущее кол-во учеников

Student students[32]; // Массив учеников

public:

// Конструктор

MyClass();

// Деструктор

~MyClass();

// Очищение полей структуры Класс

void Null();

// Инициализация структуры Класс

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

// Функции получения значений полей

string GetName();

string GetYear();

// Ввод данных в структуру Класс

void InpNameYear();

// Добавление к классу ученика

bool AddStud(Student st);

// Выод данных из структуры Класс

void DisplayShortInfo();

// Выод всех данных из структуры Класс

void DispFullInfo();

// Функция по нахождению лучших учеников в классе

bool BestStud();

// Функция по нахождению худших учеников в классе

bool BedStud();

};

Файл Class.cpp

#include "Class.h"

// Конструктор

MyClass::MyClass()

{

className = "";

yearOfStudy = "";

contStuds = 0;

Student S;

S.Null();

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

{

students[i] = S;

}

}

// Очищение полей структуры Класс

void MyClass::Null()

{

className = "";

yearOfStudy = "";

contStuds = 0;

Student S;

S.Null();

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

{

students[i] = S;

}

}

// Инициализация структуры Класс

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

{

this->className = Name;

this->yearOfStudy = Year;

this->contStuds = 0;

Student S;

S.Null();

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

{

students[i] = mas[i];

}

}

// Функции получения значений полей

string MyClass::GetName()

{

return className;

}

string MyClass::GetYear()

{

return yearOfStudy;

}

// Ввод данных в структуру Класс

void MyClass::InpNameYear()

{

cout << "Введите название класса: ";

cin >> className;

cout << endl;

cout << "Введите год обучения: ";

cin >> yearOfStudy;

cout << endl;

}

// Добавление к классу ченика

bool MyClass::AddStud(Student st)

{

bool fl = false;

if (contStuds < 20)

{

students[contStuds] = st;

contStuds++;

fl = true;

}

return fl;

}

// Вывод данных из структуры Класс

// fl\_out (0 или 1)

void MyClass::DisplayShortInfo()

{

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

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

cout << endl;

cout << "Ученики:" << endl;

int i = 0;

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

{

students[i].DisplayShortInfo();

i++;

}

}

// Выод всех данных из структуры Класс

void MyClass::DispFullInfo()

{

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

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

cout << endl;

cout << "Ученики:" << endl;

int i = 0;

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

{

students[i].DispFullInfo();

i++;

}

}

// Функция по нахождению лучших учеников в классе

bool MyClass::BestStud()

{

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].GetLessByNumber(j).GetNameLess() != "")

{

sum += students[i].GetMarkByNumber(j).Get();

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].DisplayShortInfo();

}

exit = true;

}

return exit;

}

// Функция по нахождению худших учеников в классе

bool MyClass::BedStud()

{

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].GetLessByNumber(j).GetNameLess() != "")

{

sum += students[i].GetMarkByNumber(j).Get();

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].DisplayShortInfo();

}

exit = true;

}

return exit;

}

// Деструктор

MyClass::~MyClass()

{

;

}

Файл lab2\_t1.cpp

// lab2\_t1.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include "Class.h"

// Инициализация статическоко компанента (указателя):

Mark\* Mark::lastMark = NULL;

int main()

{

//setlocale(LC\_ALL, "Russian");

//system("chcp 1251");

//system("cls");

/\*

// Создание обЪектов классов

//----------------------------------

Teacher teachers[3];

Lesson lessons[3];

Mark marks1[3];

Mark marks2[3];

Mark marks3[3];

Student students[3];

MyClass classNum1;

//----------------------------------

// Инициализация полей объектов классов

//---------------------------------------------------------------------------------

int j = 0;

teachers[0].Set("Иванова", "Марина", "Витальевна");

teachers[1].Set("Погребнеков", "Николай", "Михайлович");

teachers[2].Set("Баянова", "Людмила", "Анатольевна");

lessons[0].Set("Химия", teachers[0]);

lessons[1].Set("Обществознание", teachers[1]);

lessons[2].Set("Математика", teachers[2]);

marks1[0].Set(4);

marks1[1].Set(4);

marks1[2].Set(4);

marks2[0].Set(5);

marks2[1].Set(4);

marks2[2].Set(5);

marks3[0].Set(5);

marks3[1].Set(5);

marks3[2].Set(5);

students[0].Set("Сидоров", "Антон", "Витальевич", marks1, 3, lessons, 3);

students[1].Set("Герасимов", "Владимир", "Анатольевич", marks2, 3, lessons, 3);

students[2].Set("Иванов", "Николай", "Львович", marks3, 3, lessons, 3);

classNum1.Set("2В", "2010", students, 3);

//----------------------------------------------------------------------------------

int i = 0;

// Вывод всех данных на экран:

//----------------------------------------------------------------------------------

cout << "Вывод всех учителей:" << endl;

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

teachers[i].DisplayInfo();

cout << "\n\n";

cout << "Вывод всех уроков:" << endl;

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

lessons[i].DisplayInfo();

cout << "\n\n";

cout << "Вывод всех учеников:" << endl;

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

students[i].DisplayShortInfo();

cout << "\n\n";

cout << "Вывод Класса учеников:" << endl;

classNum1.DisplayShortInfo();

//----------------------------------------------------------------------------------

cout << "\n\n";

// Лучшие и хучшие ученики в классе

//----------------------------------------------------------------------------------

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

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

classNum1.BestStud();

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

cout << "\n\n" << endl;

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

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

classNum1.BedStud();

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

//----------------------------------------------------------------------------------

cout << "\n\n" << endl;

cout << "\n\n" << endl;

cout << "Лабораторная 7:" << endl;

cout << "\n\n" << endl;

cout << "1)" << endl;

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

cout << "Тест возврата значения из метода класса Teacher через ссылку (&)" << endl;

string Fam; string& F = Fam; teachers[0].GetFamLink(F);

string Name; string& N = Name; teachers[0].GetNameLink(Name);

string Otch; string& O = Otch; teachers[0].GetOtchLink(Otch);

cout << F << endl;

cout << N << endl;

cout << O << endl;

cout << "\n" << endl;

cout << "Тест возврата значения из метода класса Mark через указатель (\*)" << endl;

int\* mark\_p = new int;

marks1[0].GetPoint(mark\_p);

cout << \*mark\_p << endl;

cout << "\n\n" << endl;

cout << "2)" << endl;

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

cout << "Тест списка оценок с использованием указателя \'this\'" << endl;

// Формирование объектов класса Mark:

Mark A(1);

Mark B(2);

Mark C(3);

Mark D(4);

// Вызов статической компанентной функции:

Mark::reprint();

// Включение созданных компанентов в двусвязанный список:

A.Add(); B.Add(); C.Add(); D.Add();

// Печать в обратном порядке значений элементов списка:

Mark::reprint();

cout << "\n\n" << endl;

cout << "3)" << endl;

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

cout << "Дружественная функция обмена ФИО классов \n \'учитель\' Teacher и \'ученик\' Student:" << endl;

teachers[0].DisplayInfo();

cout << endl;

students[0].DisplayShortInfo();

cout << "Обмен ФИО:" << endl;

NameExchange(teachers[0], students[0]);

cout << endl;

teachers[0].DisplayInfo();

cout << endl;

students[0].DisplayShortInfo();

cout << "\n\n" << endl;

cout << "4)" << endl;

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

cout << "Перегрузка оператора \'+\' для класса \'оценка\' Mark" << endl;

Mark markPoor(2);

markPoor.DisplayValue();

cout << "Прибавим к оценке выше 2 балла:" << endl;

markPoor = markPoor + 2;

markPoor.DisplayValue();

cout << "\n" << endl;

cout << "Перегрузка оператора \'++\' (два варианта, префиксный и постфиксный) \n для класса \'оценка\' Mark" << endl;

markPoor.Set(2);

markPoor.DisplayValue();

cout << endl;

cout << "Использование оператора \'++\' (префиксный):" << endl;

++markPoor;

markPoor.DisplayValue();

cout << endl;

cout << "Использование оператора \'++\' (постфиксный):" << endl;

markPoor++;

markPoor.DisplayValue();

cout << "\n\n" << endl;

cout << "5)" << endl;

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

cout << "Заменить массивы char на std::string, продемонстрировать работу с этим классом" << endl;

cout << "Тест ввода в переменную типа std::string на примере ввода учителя:" << endl;

teachers[0].AdditionSapces();

teachers[0].DisplayInfo();

\*/

// Лабораторная №8

//===========================================================================================================

cout << ("Лабраторная 8:\n\n");

cout << ("=========\nМодифицировать ваши проекты на С++, C# и Java путем добавления в один из классов\nкак минимум одного статического поля и одного статического метода.\n=========\n\n");

cout << ("(Использовались статический член класса\'Mark\' \'lastMark\'и\nстатические методы класса \'NewList\' и \'reprint\')\n\n");

// Формирование объектов класса Mark:

Mark List1(1);

Mark List2(2);

Mark List3(3);

Mark List4(4);

cout << ("Вызов статической компанентной функции: \'Новый список\'\n\n");

// Вызов статической компанентной функции:

Mark::NewList();

cout << ("Вызов статической компанентной функции: \'Напечатать список\'\n\n");

// Вызов статической компанентной функции:

Mark::reprint();

cout << ("\n\nДобавление элементов в список.\n\n");

// Включение созданных компанентов в двусвязанный список:

List1.Add(); List2.Add(); List3.Add(); List4.Add();

cout << ("Вызов статической компанентной функции: \'Напечатать список\'\n\n");

cout << ("===================================\n");

// Печать в обратном порядке значений элементов списка:

Mark::reprint();

cout << ("\n===================================\n");

}

Java

Файл Teacher.java

**package** School;

**import** java.util.Scanner;

//Класс Учитель

**public** **class** Teacher {

**private** String fam = ""; // Фамилия

**private** String name = ""; // Имя

**private** String otch = ""; // Отчество

// Конструктор

**public** Teacher()

{

fam = "";

name = "";

otch = "";

}

// Фун-ии получения данных из полей

**public** String GetFam()

{

**return** **this**.fam;

}

**public** String GetName()

{

**return** **this**.name;

}

**public** String GetOtch()

{

**return** **this**.otch;

}

// Инициализация структуры Учитель

**public** **void** Set(String Fam\_s, String Name\_s, String Otch\_s)

{

**this**.fam = Fam\_s;

**this**.name = Name\_s;

**this**.otch = Otch\_s;

}

// Ввод учителя

**public** **void** InpFIO()

{

//String encoding = System.getProperty("console.encoding", "utf-8");

//Scanner in = new Scanner(System.in, encoding);

//String os = System.getProperty("os.name");

//String consoleEncoding = System.getProperty("console.encoding", os.startsWith("Windows") ? "cp866" : "utf-8");

//Scanner in = new Scanner(System.in, consoleEncoding);

Scanner in = **new** Scanner(System.***in***);

System.***out***.println("Введите Фамилию учителя: ");

fam = in.nextLine();

System.***out***.println("Введите имя учителя: ");

name = in.nextLine();

System.***out***.println("Введите отчество учителя: ");

otch = in.nextLine();

in.close();

}

// Вывод учителя

**public** **void** DisplayInfo()

{

System.***out***.println("ФИО учителя: " + fam + " " + name + " " + otch);

}

// Обработка строк

**public** **void** AdditionSapces()

{

fam = fam + " ";

name = name + " ";

otch = otch + " ";

}

// Обработка строк, сосздание новой строки из подстрок

**public** String First4SimbFromFIO()

{

**return** (fam.substring(0, 4) + name.substring(0, 4) + otch.substring(0, 4));

}

}

Файл Lesson.java

**package** School;

**import** java.util.\*;

//Класс Уроки

**public** **class** Lesson {

**private** String nameLesson = ""; // Название предмета

**private** Teacher teacher = **new** Teacher(); // Учитель

// Конструктор

**public** Lesson()

{

**this**.nameLesson = "";

**this**.teacher.Set("", "", "");

}

// Инициализация структуры Уроки

**public** **void** Set(String Name\_Less, Teacher t)

{

nameLesson = Name\_Less;

teacher = t;

}

// Обнуление полей

**public** **void** Null()

{

**this**.nameLesson = "";

**this**.teacher.Set("", "", "");

}

// Фун-ии получения данных из полей

**public** String GetNameLess()

{

**return** **this**.nameLesson;

}

**public** Teacher GetTeacher()

{

**return** **this**.teacher;

}

// Ввод названия урока

**public** **void** InpInConsol(Teacher t)

{

Scanner in = **new** Scanner(System.***in***);

//String os = System.getProperty("os.name");

//String consoleEncoding = System.getProperty("console.encoding", os.startsWith("Windows") ? "cp866" : "utf-8");

//Scanner in = new Scanner(System.in, consoleEncoding);

System.***out***.print("Введите название предмета: ");

nameLesson = in.nextLine();

System.***out***.println();

teacher = t;

in.close();

}

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

**public** **void** DisplayInfo()

{

System.***out***.println("Название предмета: " + nameLesson);

teacher.DisplayInfo();

}

// Вывод урока с подписью, что этот урок очень важен

**public** **void** DisplayImportantInfo()

{

System.***out***.println("Этот урок очень важен!\n");

// Вызыв метода через указатель this

**this**.DisplayInfo();

}

}

Файл Mark.java

**package** School;

**import** java.util.Scanner;

//Класс Оценка

**public** **class** Mark {

**private** **int** value; // Оценка

**private** **static** Mark *lastMark*;

**private** Mark prev;

**private** Mark next;

// Конструктор

**public** Mark()

{

value = 0;

}

// Конструктор с параметрами

**public** Mark(**int** value)

{

**this**.value = value;

}

// Инициализация структуры Оценка

**public** **void** Set(**int** v)

{

value = v;

}

// Получение оценки

**public** **int** Get()

{

**return** value;

}

// Ввод данных в структуру Оценка

**public** **void** InpMark()

{

Scanner in = **new** Scanner(System.***in***);

//String os = System.getProperty("os.name");

//String consoleEncoding = System.getProperty("console.encoding", os.startsWith("Windows") ? "cp866" : "utf-8");

//Scanner in = new Scanner(System.in, consoleEncoding);

String str = **new** String();

//value = in.nextInt();

**int** fl = 0;

**int** number = 0;

**while** (fl == 0)

{

System.***out***.print("Введите оценку: ");

**if** (in.hasNextInt())

{

number = in.nextInt();

str = in.nextLine();

value = number;

fl = 1;

}

**else**

{

System.***out***.println("Извините, но это явно не число. Введите число заново!");

System.***out***.println("\n");

}

}

System.***out***.println();

in.close();

}

// Вывод данных из структуры

**public** **void** DisplayValue()

{

System.***out***.println("Оценка: " + value);

}

// Новый список

**public** **static** **void** NewList()

{

*lastMark* = **null**;

}

// Добавление элемента в конец списка

**public** **void** Add()

{

**if** (*lastMark* == **null**)

**this**.prev = **null**;

**else**

{

*lastMark*.next = **this**;

prev = *lastMark*;

}

*lastMark* = **this**;

**this**.next = **null**;

}

// Вывод на дисплей содержимого списка

**public** **static** **void** reprint()

{

Mark uk; // Вспомогательная ссылка

uk = *lastMark*;

**if** (uk == **null**)

{

System.***out***.println("Список пуст!");

**return**;

}

**else**

System.***out***.println("\nСодержимое списка:\n");

// Цикл печати в обратном порядке значений элементов списка:

**while** (uk != **null**)

{

System.***out***.println(uk.value + "\t");

uk = uk.prev;

}

}

}

Файл Student.java

**package** School;

**import** java.util.Scanner;

//Класс Ученик

**public** **class** Student {

**private** String fam = ""; // Фамилия

**private** String name = ""; // Имя

**private** String otch = ""; // Отчество

**private** Mark[] marks = **new** Mark[20]; // Массив оценок

**private** Lesson[] lessons = **new** Lesson[20]; // Массив уроков

// Конструктор

**public** Student()

{

fam = "";

name = "";

otch = "";

Lesson L = **new** Lesson();

Mark M = **new** Mark();

L.Null();

M.Set(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)

**public** **void** Null()

{

fam = "";

name = "";

otch = "";

Lesson L = **new** Lesson();

Mark M = **new** Mark();

L.Null();

M.Set(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

**public** String GetFam()

{

**return** **this**.fam;

}

**public** String GetName()

{

**return** **this**.name;

}

**public** String GetOtch()

{

**return** **this**.otch;

}

**public** Mark GetMarkByNumber(**int** i)

{

Mark m = **new** Mark();

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

m = marks[i];

**return** m;

}

**public** Lesson GetLessByNumber(**int** i)

{

Lesson l = **new** Lesson();

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

l = lessons[i];

**return** l;

}

// Инициализация структуры Ученик

**public** **void** Set(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;

**int** i = 0;

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

{

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

}

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

{

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

}

}

// Ввод ФИО ученика в структуру

**public** **void** InpStudFIO()

{

Scanner in = **new** Scanner(System.***in***);

//String os = System.getProperty("os.name");

//String consoleEncoding = System.getProperty("console.encoding", os.startsWith("Windows") ? "cp866" : "utf-8");

//Scanner in = new Scanner(System.in, consoleEncoding);

System.***out***.print("Введите Фамилию ученика: ");

fam = in.nextLine();

System.***out***.print("Введите Имя ученика: ");

name = in.nextLine();

System.***out***.print("Введите Отчество ученика: ");

otch = in.nextLine();

in.close();

}

// Добавление к ученику урока

// 0 - не удачно, 1 - удачно

**public** **boolean** AddLess(Lesson ls)

{

**boolean** fl = **false**;

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

{

**if** (lessons[i].GetNameLess() == "")

{

lessons[i] = ls;

fl = **true**;

}

}

**return** fl;

}

// Добавление к ученику урока и оценки

// 0 - не удачно, 1 - удачно

**public** **boolean** AddLessAndMark(Lesson ls, Mark m)

{

**boolean** fl = **false**;

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

{

**if** (lessons[i].GetNameLess() == "")

{

lessons[i] = ls;

marks[i] = m;

fl = **true**;

}

}

**return** fl;

}

// Добавление к ученику оценки к конкретному предмету

// 0 - не удачно, 1 - удачно

**public** **boolean** AddMark(Lesson ls, Mark m)

{

**boolean** fl = **false**;

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

{

**if** (lessons[i].GetNameLess() == ls.GetNameLess())

{

**if** (lessons[i].GetTeacher().GetFam() == ls.GetTeacher().GetFam())

{

**if** (lessons[i].GetTeacher().GetName() == ls.GetTeacher().GetName())

{

**if** (lessons[i].GetTeacher().GetOtch() == ls.GetTeacher().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)

**public** **void** DisplayShortInfo()

{

System.***out***.println("ФИО ученика: " + fam + " " + name + " " + otch);

System.***out***.println("Уроки:");

**int** i = 0;

**while** (lessons[i].GetNameLess() != "")

{

System.***out***.println(" \* " + lessons[i].GetNameLess() + " \* " + marks[i].Get() + " \* ");

i++;

}

}

// Вывод ученика с всеми предметами, даже пустыми

**public** **void** DispFullInfo()

{

System.***out***.println("ФИО ученика: " + fam + " " + name + " " + otch);

System.***out***.println("Уроки:");

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

{

lessons[i].DisplayInfo();

}

System.***out***.println("Оценки:");

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

{

marks[i].DisplayValue();

}

}

}

Файл MyClass.java

**package** School;

**import** java.util.Scanner;

//Класс Класс

**public** **class** MyClass {

**private** String className = ""; // Название класса

**private** String yearOfStudy = ""; // Год обучения

**private** **int** contStuds = 0; // Текущее кол-во учеников

**private** Student[] students = **new** Student[32]; // Массив учеников

// Конструктор

**public** MyClass()

{

className = "";

yearOfStudy = "";

contStuds = 0;

Student S = **new** Student();

S.Null();

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

{

students[i] = S;

}

}

// Очищение полей структуры Класс

**public** **void** Null()

{

className = "";

yearOfStudy = "";

contStuds = 0;

Student S = **new** Student();

S.Null();

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

{

students[i] = S;

}

}

// Инициализация структуры Класс

**public** **void** Set(String Name, String Year, Student mas[], **int** LenStud)

{

**this**.className = Name;

**this**.yearOfStudy = Year;

**this**.contStuds = 0;

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

{

students[i] = mas[i];

}

}

// Функции получения значений полей

**public** String GetName()

{

**return** className;

}

**public** String GetYear()

{

**return** yearOfStudy;

}

// Ввод данных в структуру Класс

**public** **void** InpNameYear()

{

Scanner in = **new** Scanner(System.***in***);

//String os = System.getProperty("os.name");

//String consoleEncoding = System.getProperty("console.encoding", os.startsWith("Windows") ? "cp866" : "utf-8");

//Scanner in = new Scanner(System.in, consoleEncoding);

System.***out***.print("Введите название класса: ");

className = in.nextLine();

System.***out***.println("");

System.***out***.print("Введите год обучения: ");

yearOfStudy = in.nextLine();

System.***out***.println("");

in.close();

}

// Добавление к классу ченика

**public** **boolean** AddStud(Student st)

{

**boolean** fl = **false**;

**if** (contStuds < 20)

{

students[contStuds] = st;

contStuds++;

fl = **true**;

}

**return** fl;

}

// Вывод данных из структуры Класс

**public** **void** DisplayShortInfo()

{

System.***out***.println("Название класса: " + className);

System.***out***.println("Год обучения: " + yearOfStudy);

System.***out***.println("");

System.***out***.println("Ученики:");

**int** i = 0;

**while** (students[i].GetFam() != "")

{

students[i].DisplayShortInfo();

i++;

}

}

// Выод всех данных из структуры Класс

**public** **void** DispFullInfo()

{

System.***out***.println("Название класса: " + className);

System.***out***.println("Год обучения: " + yearOfStudy);

System.***out***.println("");

System.***out***.println("Ученики:");

**int** i = 0;

**while** (students[i].GetFam() != "")

{

students[i].DispFullInfo();

i++;

}

}

// Функция по нахождению лучших учеников в классе

**public** **boolean** BestStud()

{

**double**[] mas\_m = **new** **double**[32];

**boolean** exit = **false**;

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

{

mas\_m[i] = 0;

}

**int** i = 0;

**int** k = 0;

**double** sum = 0;

**while** (students[i].GetFam() != "")

{

**int** j = 0;

k = 0;

sum = 0;

**while** (students[i].GetLessByNumber(j).GetNameLess() != "")

{

sum += students[i].GetMarkByNumber(j).Get();

k++;

j++;

}

mas\_m[i] = 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].DisplayShortInfo();

}

exit = **true**;

}

**return** exit;

}

// Функция по нахождению худших учеников в классе

**public** **boolean** BedStud()

{

**double**[] mas\_m = **new** **double**[32];

**boolean** exit = **false**;

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

{

mas\_m[i] = 0;

}

**int** i = 0;

**int** k = 0;

**double** sum = 0;

**while** (students[i].GetFam() != "")

{

**int** j = 0;

k = 0;

sum = 0;

**while** (students[i].GetLessByNumber(j).GetNameLess() != "")

{

sum += students[i].GetMarkByNumber(j).Get();

k++;

j++;

}

mas\_m[i] = 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].DisplayShortInfo();

}

exit = **true**;

}

**return** exit;

}

}

Файл Start.java

**package** School;

**import** java.util.Scanner;

**public** **class** Start {

**public** **static** **void** main(String[] args)

{

//Scanner in = new Scanner(System.in);

//String os = System.getProperty("os.name");

//String consoleEncoding = System.getProperty("console.encoding", os.startsWith("Windows") ? "cp866" : "utf-8");

//Scanner in = new Scanner(System.in, consoleEncoding);

/\*

// Создание обЪектов классов

//----------------------------------

Teacher[] teachers = new Teacher[3];

Lesson[] lessons = new Lesson[3];

Mark[] marks1 = new Mark[3];

Mark[] marks2 = new Mark[3];

Mark[] marks3 = new Mark[3];

Student[] students = new Student[3];

MyClass classNum1 = new MyClass();

//----------------------------------

// Инициализация полей объектов классов

//---------------------------------------------------------------------------------

int j = 0;

for (j = 0; j < 3; j++)

teachers[j] = new Teacher();

teachers[0].Set("Иванова", "Марина", "Витальевна");

teachers[1].Set("Погребнеков", "Николай", "Михайлович");

teachers[2].Set("Баянова", "Людмила", "Анатольевна");

for (j = 0; j < 3; j++)

lessons[j] = new Lesson();

lessons[0].Set("Химия", teachers[0]);

lessons[1].Set("Обществознание", teachers[1]);

lessons[2].Set("Математика", teachers[2]);

for (j = 0; j < 3; j++)

{

marks1[j] = new Mark();

marks2[j] = new Mark();

marks3[j] = new Mark();

}

marks1[0].Set(4);

marks1[1].Set(4);

marks1[2].Set(4);

marks2[0].Set(5);

marks2[1].Set(4);

marks2[2].Set(5);

marks3[0].Set(5);

marks3[1].Set(5);

marks3[2].Set(5);

for (j = 0; j < 3; j++)

students[j] = new Student();

students[0].Set("Сидоров", "Антон", "Витальевич", marks1, 3, lessons, 3);

students[1].Set("Герасимов", "Владимир", "Анатольевич", marks2, 3, lessons, 3);

students[2].Set("Иванов", "Николай", "Львович", marks3, 3, lessons, 3);

classNum1.Set("2В", "2010", students, 3);

//----------------------------------------------------------------------------------

int i = 0;

\*/

/\*

// Вывод всех данных на экран:

//----------------------------------------------------------------------------------

System.out.println("Вывод всех учителей:");

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

teachers[i].DisplayInfo();

System.out.println("\n\n");

System.out.println("Вывод всех уроков:");

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

lessons[i].DisplayInfo();

System.out.println("\n\n");

System.out.println("Вывод всех учеников:");

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

students[i].DisplayShortInfo();

System.out.println("\n\n");

System.out.println("Вывод Класса учеников:");

classNum1.DisplayShortInfo();

//----------------------------------------------------------------------------------

System.out.println("\n\n");

// Лучшие и хучшие ученики в классе

//----------------------------------------------------------------------------------

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция поиска лучших учеников в классе\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Лучшие ученики класса №1:");

classNum1.BestStud();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\n\n");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Функция поиска худших учеников в классе\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Худшие ученики класса №1:");

classNum1.BedStud();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//----------------------------------------------------------------------------------

System.out.println("\n\n");

//System.out.println("Тест ввода русских символов на примере ввода учителя:");

//teachers[0].InpFIO();

//teachers[0].DisplayInfo();

\*/

/\*

System.out.println("6) Продемонстрировать работу с массивом объектов");

System.out.println("===========================================================================================");

System.out.println("На примере объектов класса Оценка");

marks1[0].Set(4);

marks1[1].Set(4);

marks1[2].Set(4);

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

marks1[i].DisplayValue();

System.out.println("\n\n");

System.out.println("7) Продемонстрировать возврат целочисленного значения из метода через вспомогательный класс");

System.out.println("===========================================================================================");

System.out.println("На примере объектов класса Student и Mark");

Mark markAuxiliary = new Mark();

markAuxiliary = students[0].GetMarkByNumber(1);

markAuxiliary.DisplayValue();

System.out.println("\n\n");

System.out.println("8) Продемонстрировать разумное использование оператора this");

System.out.println("===========================================================================================");

System.out.println("На примере класса Оценка");

// Формирование объектов класса Mark:

Mark List1 = new Mark(1);

Mark List2 = new Mark(2);

Mark List3 = new Mark(3);

Mark List4 = new Mark(4);

Mark.NewList();

// Вызов статической компанентной функции:

Mark.reprint();

// Включение созданных компанентов в двусвязанный список:

List1.Add(); List2.Add(); List3.Add(); List4.Add();

// Печать в обратном порядке значений элементов списка:

Mark.reprint();

System.out.println("\n\n");

System.out.println("9) Продемонстрировать обработку строк (String)");

System.out.println("===========================================================================================");

System.out.println("На примере класса Учитель, сравнение строк\n");

System.out.println("У учителей:");

teachers[0].DisplayInfo();

teachers[1].DisplayInfo();

if (teachers[0].GetFam().equals(teachers[1].GetFam()))

System.out.println("Фамилии одинаковые!\n");

else

System.out.println("Фамилии разные!\n");

System.out.println("На примере класса Учитель, добавление ко всем полям пробелов");

teachers[0].AdditionSapces();

teachers[0].DisplayInfo();

System.out.println();

System.out.println("На примере класса Учитель, подстроки");

System.out.println(teachers[0].First4SimbFromFIO());

\*/

// Лабораторная №8

//===========================================================================================================

System.***out***.print("Лабраторная 8:\n\n");

System.***out***.print("=========\nМодифицировать ваши проекты на С++, C# и Java путем добавления в один из классов\nкак минимум одного статического поля и одного статического метода.\n=========\n\n");

System.***out***.print("(Использовались статический член класса\'Mark\' \'lastMark\'и\nстатические методы класса \'NewList\' и \'reprint\')\n\n");

// Формирование объектов класса Mark:

Mark List1 = **new** Mark(1);

Mark List2 = **new** Mark(2);

Mark List3 = **new** Mark(3);

Mark List4 = **new** Mark(4);

System.***out***.print("Вызов статической компанентной функции: \'Новый список\'\n\n");

// Вызов статической компанентной функции:

Mark.*NewList*();

System.***out***.print("Вызов статической компанентной функции: \'Напечатать список\'\n\n");

// Вызов статической компанентной функции:

Mark.*reprint*();

System.***out***.print("\n\nДобавление элементов в список.\n\n");

// Включение созданных компанентов в двусвязанный список:

List1.Add(); List2.Add(); List3.Add(); List4.Add();

System.***out***.print("Вызов статической компанентной функции: \'Напечатать список\'\n\n");

System.***out***.print("===================================\n");

// Печать в обратном порядке значений элементов списка:

Mark.*reprint*();

System.***out***.print("\n===================================\n");

}

}

C#

Файл Teacher.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_csh

{

// Класс Учитель

public class Teacher

{

// Фамилия

private string fam;

// Имя

private string name;

// Отчество

private string otch;

private static Teacher lastTeacher;

private Teacher prev;

private Teacher next;

public string Fam

{

get => fam;

set => fam = value;

}

public string Name

{

get => name;

set => name = value;

}

public string Otch

{

get => otch;

set => otch = value;

}

// Конструктор

public Teacher()

{

fam = "";

name = "";

otch = "";

}

// Конструктор с параметрами

public Teacher(string Fam\_s, string Name\_s, string Otch\_s)

{

this.fam = Fam\_s;

this.name = Name\_s;

this.otch = Otch\_s;

}

// Фун-ии получения данных из полей

public string GetFam()

{

return this.fam;

}

public string GetName()

{

return this.name;

}

public string GetOtch()

{

return this.otch;

}

// Задание полей класса Учитель

public void Set(string Fam\_s, string Name\_s, string Otch\_s)

{

this.fam = Fam\_s;

this.name = Name\_s;

this.otch = Otch\_s;

}

// Ввод ФИО учителя

public void InpFIO()

{

Console.Write("Введите Фамилию учителя: ");

fam = Console.ReadLine();

Console.Write("Введите имя учителя: ");

name = Console.ReadLine();

Console.Write("Введите отчество учителя: ");

otch = Console.ReadLine();

}

// Ввод учителя

public void DisplayInfo()

{

Console.Write("ФИО учителя: " + fam + " " + name + " " + otch + "\n");

}

// Обработка строк

public void AdditionSapces()

{

fam = fam + " ";

name = name + " ";

otch = otch + " ";

}

// Новый список

public static void NewList()

{

lastTeacher = null;

}

// Добавление элемента в конец списка

public void Add()

{

if (lastTeacher == null)

this.prev = null;

else

{

lastTeacher.next = this;

prev = lastTeacher;

}

lastTeacher = this;

this.next = null;

}

// Вывод на дисплей содержимого списка

public static void reprint()

{

Teacher uk; // Вспомогательная ссылка

uk = lastTeacher;

if (uk == null)

{

Console.WriteLine("Список пуст!");

return;

}

else

Console.WriteLine("\nСодержимое списка:\n");

// Цикл печати в обратном порядке значений элементов списка:

while (uk != null)

{

Console.WriteLine(uk.GetFam() + " " + uk.GetName() + " " + uk.GetOtch() + " " + "\t");

uk = uk.prev;

}

}

}

}

Файл Lesson.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_csh

{

// Класс Урок

public class Lesson

{

private string nameLesson = ""; // Название предмета

private Teacher teacher = new Teacher(); // Учитель

// Конструктор

public Lesson()

{

this.nameLesson = "";

this.teacher.Set("", "", "");

}

// Задание полей класса Урок

public void Set(string Name\_Less, Teacher t)

{

nameLesson = Name\_Less;

teacher = t;

}

// Обнуление полей

public void Null()

{

this.nameLesson = "";

this.teacher.Set("", "", "");

}

// Фун-ии получения данных из полей

public string GetNameLess()

{

return this.nameLesson;

}

public Teacher GetTeacher()

{

return this.teacher;

}

// Ввод названия урока

public void InpInConsol(Teacher t)

{

Console.Write("Введите название предмета: ");

nameLesson = Console.ReadLine();

Console.Write("\n");

teacher = t;

}

// Вывод урока

public void DisplayInfo()

{

Console.Write("Название предмета: " + nameLesson + "\n");

teacher.DisplayInfo();

}

// Вывод урока с подписью, что этот урок очень важен

public void DisplayImportantInfo()

{

Console.Write("Этот урок очень важен!\n");

// Вызыв метода через указатель this

this.DisplayInfo();

}

}

}

Файл Mark.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_csh

{

// Структура оценка

public struct Mark

{

private int value /\*= 0\*/; // Оценка

// Конструктор

//public Mark()

//{

// value = 0;

//}

// Конструктор с параметром

public Mark(int Value)

{

value = Value;

}

// Задание полей класса Оценка

public void Set(int v)

{

value = v;

}

// Получение оценки

public int Get()

{

return value;

}

// Ввод данных в класс Оценка

public void InpMark()

{

string mark;

Console.Write("Введите оценку: ");

mark = Console.ReadLine();

value = Convert.ToInt32(mark);

Console.Write("\n");

}

// Вывод данных на дисплей

public void DisplayValue()

{

Console.Write("Оценка: " + value + "\n");

}

// Проверка возврата значения по ref

public void TestRef4(ref int Value)

{

Value = 4;

}

// Проверка возврата значения по out

public void TestOut5(out int Value)

{

Value = 5;

}

public static Mark operator +(Mark operand1, Mark operand2)

=> new Mark(operand1.value + operand2.value);

public static Mark operator ++(Mark operand1)

=> new Mark(operand1.value + 1);

}

}

Файл Student.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_csh

{

// Класс ученик

class Student

{

private string fam = ""; // Фамилия

private string name = ""; // Имя

private string otch = ""; // Отчество

private Mark[] marks = new Mark [20]; // Массив оценок

private Lesson[] lessons = new Lesson[20]; // Массив уроков

private int countLess = 0; // Текущее кол-во уроков

// Конструктор

public Student()

{

fam = "";

name = "";

otch = "";

Lesson L = new Lesson();

Mark M = new Mark();

L.Null();

M.Set(0);

countLess = 0;

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

{

lessons[i] = L;

}

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

{

marks[i] = M;

}

}

// Очищение всех полей класса Ученик (Student)

public void Null()

{

fam = "";

name = "";

otch = "";

Lesson L = new Lesson();

Mark M = new Mark();

L.Null();

M.Set(0);

this.countLess = 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

public string GetFam()

{

return this.fam;

}

public string GetName()

{

return this.name;

}

public string GetOtch()

{

return this.otch;

}

public Mark GetMarkByNumber(int i)

{

Mark m = new Mark();

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

m = marks[i];

return m;

}

public Lesson GetLessByNumber(int i)

{

Lesson l = new Lesson();

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

l = lessons[i];

return l;

}

// Задание полей класса Ученик

public void Set(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.countLess = 0;

int i = 0;

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

{

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

this.countLess++;

}

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

{

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

}

}

// Ввод ФИО ученика в класс

public void InpStudFIO()

{

Console.Write("Введите Фамилию ученика: ");

fam = Console.ReadLine();

Console.Write("Введите Имя ученика: ");

name = Console.ReadLine();

Console.Write("Введите Отчество ученика: ");

otch = Console.ReadLine();

}

// Добавление к ученику урока

// 0 - не удачно, 1 - удачно

public bool AddLess(Lesson ls)

{

bool fl = false;

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

{

if (lessons[i].GetNameLess() == "")

{

lessons[i] = ls;

fl = true;

}

}

return fl;

}

// Добавление к ученику урока и оценки

// 0 - не удачно, 1 - удачно

public bool AddLessAndMark(Lesson ls, Mark m)

{

bool fl = false;

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

{

if (lessons[i].GetNameLess() == "")

{

lessons[i] = ls;

marks[i] = m;

fl = true;

}

}

return fl;

}

// Добавление к ученику оценки к конкретному предмету

// 0 - не удачно, 1 - удачно

public bool AddMark(Lesson ls, Mark m)

{

bool fl = false;

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

{

if (lessons[i].GetNameLess() == ls.GetNameLess())

{

if (lessons[i].GetTeacher().GetFam() == ls.GetTeacher().GetFam())

{

if (lessons[i].GetTeacher().GetName() == ls.GetTeacher().GetName())

{

if (lessons[i].GetTeacher().GetOtch() == ls.GetTeacher().GetOtch())

{

marks[i] = m;

fl = true;

}

}

}

}

}

return fl;

}

// Вывод ученика

public void DisplayShortInfo()

{

Console.Write("ФИО ученика: " + fam + " " + name + " " + otch + "\n");

Console.Write("Уроки:" + "\n");

int i = 0;

while (lessons[i].GetNameLess() != "")

{

Console.Write(" \* " + lessons[i].GetNameLess() + " \* " + marks[i].Get() + " \* " + "\n");

i++;

}

}

// Вывод ученика со всеми предметами, даже пустыми

public void DispFullInfo()

{

Console.Write("ФИО ученика: " + fam + " " + name + " " + otch + "\n");

Console.Write("Уроки:" + "\n");

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

{

lessons[i].DisplayInfo();

}

Console.Write("Оценки:" + "\n");

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

{

marks[i].DisplayValue();

}

}

}

}

Файл MyClass.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab6\_csh

{

// Класс Мой Класс

class MyClass

{

private string className = ""; // Название класса

private string yearOfStudy = ""; // Год обучения

private int contStuds = 0; // Текущее кол-во учеников

private Student[] students = new Student[32]; // Массив учеников

// Конструктор

public MyClass()

{

className = "";

yearOfStudy = "";

contStuds = 0;

Student S = new Student();

S.Null();

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

{

students[i] = S;

}

}

// Очищение полей класса Мой Класс

public void Null()

{

className = "";

yearOfStudy = "";

contStuds = 0;

Student S = new Student();

S.Null();

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

{

students[i] = S;

}

}

// Задание полей класса Мой Класс

public void Set(string Name, string Year, Student[] mas, int LenStud)

{

this.className = Name;

this.yearOfStudy = Year;

this.contStuds = 0;

Student S = new Student();

S.Null();

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

{

students[i] = mas[i];

}

}

// Функции получения значений полей

public string GetName()

{

return className;

}

public string GetYear()

{

return yearOfStudy;

}

// Ввод данных в Мой Класс

public void InpNameYear()

{

Console.Write("Введите название класса: ");

className = Console.ReadLine();

Console.Write("\n");

Console.Write("Введите год обучения: ");

yearOfStudy = Console.ReadLine();

Console.Write("\n");

}

// Добавление к классу ученика

public bool AddStud(Student st)

{

bool fl = false;

if (contStuds < 20)

{

students[contStuds] = st;

contStuds++;

fl = true;

}

return fl;

}

// Вывод данных из класса Мой Класс

public void DisplayShortInfo()

{

Console.Write("Название класса: " + className + "\n");

Console.Write("Год обучения: " + yearOfStudy + "\n");

Console.Write("\n");

Console.Write("Ученики:" + "\n");

int i = 0;

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

{

students[i].DisplayShortInfo();

i++;

}

}

// Выод всех данных из класса Мой Класс

public void DispFullInfo()

{

Console.Write("Название класса: " + className + "\n"); ;

Console.Write("Год обучения: " + yearOfStudy + "\n");

Console.Write("\n");

Console.Write("Ученики:" + "\n");

int i = 0;

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

{

students[i].DispFullInfo();

i++;

}

}

// Функция по нахождению лучших учеников в 'классе'

public bool BestStud()

{

double[] mas\_m = new double[32];

bool exit = false;

int i = 0;

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

{

mas\_m[i] = 0;

}

i = 0;

int k = 0;

double sum = 0;

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

{

int j = 0;

k = 0;

sum = 0;

while (students[i].GetLessByNumber(j).GetNameLess() != "")

{

sum += students[i].GetMarkByNumber(j).Get();

k++;

j++;

}

mas\_m[i] = 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].DisplayShortInfo();

}

exit = true;

}

return exit;

}

// Функция по нахождению худших учеников в 'классе'

public bool BedStud()

{

double[] mas\_m = new double[32];

bool exit = false;

int i = 0;

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

{

mas\_m[i] = 0;

}

i = 0;

int k = 0;

double sum = 0;

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

{

int j = 0;

k = 0;

sum = 0;

while (students[i].GetLessByNumber(j).GetNameLess() != "")

{

sum += students[i].GetMarkByNumber(j).Get();

k++;

j++;

}

mas\_m[i] = 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].DisplayShortInfo();

}

exit = true;

}

return exit;

}

}

}

Файл Program.cs

using System;

namespace lab6\_csh

{

class Program

{

static void Main(string[] args)

{

// Создание обЪектов классов

//----------------------------------

Teacher[] teachers = new Teacher[3];

Lesson[] lessons = new Lesson[3];

Mark[] marks1 = new Mark[3];

Mark[] marks2 = new Mark[3];

Mark[] marks3 = new Mark[3];

Student[] students = new Student[3];

MyClass classNum1 = new MyClass();

//----------------------------------

// Инициализация полей объектов классов

//---------------------------------------------------------------------------------

int j = 0;

for (j = 0; j < 3; j++)

teachers[j] = new Teacher();

teachers[0].Set("Иванова", "Марина", "Витальевна");

teachers[1].Set("Погребнеков", "Николай", "Михайлович");

teachers[2].Set("Баянова", "Людмила", "Анатольевна");

for (j = 0; j < 3; j++)

lessons[j] = new Lesson();

lessons[0].Set("Химия", teachers[0]);

lessons[1].Set("Обществознание", teachers[1]);

lessons[2].Set("Математика", teachers[2]);

for (j = 0; j < 3; j++)

{

marks1[j] = new Mark();

marks2[j] = new Mark();

marks3[j] = new Mark();

}

marks1[0].Set(4);

marks1[1].Set(4);

marks1[2].Set(4);

marks2[0].Set(5);

marks2[1].Set(4);

marks2[2].Set(5);

marks3[0].Set(5);

marks3[1].Set(5);

marks3[2].Set(5);

for (j = 0; j < 3; j++)

students[j] = new Student();

students[0].Set("Сидоров", "Антон", "Витальевич", marks1, 3, lessons, 3);

students[1].Set("Герасимов", "Владимир", "Анатольевич", marks2, 3, lessons, 3);

students[2].Set("Иванов", "Николай", "Львович", marks3, 3, lessons, 3);

classNum1.Set("2В", "2010", students, 3);

//----------------------------------------------------------------------------------

int i = 0;

// Вывод всех данных на экран:

//----------------------------------------------------------------------------------

/\*

Console.Write("Вывод всех учителей:");

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

teachers[i].DisplayInfo();

Console.Write("\n\n");

Console.Write("Вывод всех уроков:");

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

lessons[i].DisplayInfo();

Console.Write("\n\n");

Console.Write("Вывод всех учеников:");

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

students[i].DisplayShortInfo();

Console.Write("\n\n");

Console.Write("Вывод Класса учеников:");

classNum1.DisplayShortInfo();

//----------------------------------------------------------------------------------

Console.Write("\n\n");

// Лучшие и хучшие ученики в классе

//----------------------------------------------------------------------------------

Console.Write("\*\*\*\*\*\*\*Функция поиска лучших учеников в классе\*\*\*\*\*\*\*\*\*\n");

Console.Write("Лучшие ученики класса №1:");

classNum1.BestStud();

Console.Write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

Console.Write("\n\n");

Console.Write("\*\*\*\*\*\*\*Функция поиска худших учеников в классе\*\*\*\*\*\*\*\*\n");

Console.Write("Худшие ученики класса №1:");

classNum1.BedStud();

Console.Write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

//----------------------------------------------------------------------------------

Console.Write("\n\n");

//Console.Write("Тест ввода русских символов на примере ввода учителя:" + "\n");

//teachers[0].InpFIO();

//teachers[0].DisplayInfo();

\*/

// Лабораторная №7

//===========================================================================================================

/\*

Console.Write("Лабраторная 7:\n\n");

Console.Write("10) Для полей добавить свойства и продемонстрировать работу с ними\n");

Console.Write("==================================================================\n");

Console.Write("На примере класса Учитель\n");

Teacher Oleg = new Teacher();

Oleg.Fam = "Симонов";

Oleg.Name = "Олег";

Oleg.Otch = "Максимович";

Console.WriteLine("ФИО: " + Oleg.Fam + " " + Oleg.Name + " " + Oleg.Otch);

Console.Write("\n\n");

Console.Write("11) Cоздать массив объектов и продемонстрировать работу с ним\n");

Console.Write("==================================================================\n");

Console.Write("На примере класса Учитель\n");

Teacher[] teachs = new Teacher[3];

for (j = 0; j < 3; j++)

teachs[j] = new Teacher();

teachs[0].Set("Иванова", "Марина", "Витальевна");

teachs[1].Set("Погребнеков", "Николай", "Михайлович");

teachs[2].Set("Баянова", "Людмила", "Анатольевна");

Console.Write("Вывод всех учителей:\n");

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

teachs[i].DisplayInfo();

Console.Write("\n\n");

Console.Write("12) В отдельной ветке проекта заменить класс (class) \nна структуру (struct). Продемонстрировать различие \nмежду присваиванием объектов класса и структуры\n");

Console.Write("==================================================================\n");

Console.Write("На примере класса Учитель и структуры Оценка\n");

Mark good = new Mark();

Teacher Ivan = new Teacher();

good.Set(4);

Ivan.Set("Сидоров", "Иван", "Анатольевич");

Console.WriteLine("Элемент структуры до копирования:");

good.DisplayValue();

Console.Write("\n");

Console.WriteLine("Элемент класса до копирования:");

Ivan.DisplayInfo();

Console.Write("\n");

Console.WriteLine("Элемент структуры копируется на:");

marks2[0].DisplayValue();

Console.Write("\n");

Console.WriteLine("Элемент класса копируется на:");

Oleg.DisplayInfo();

Console.Write("\n");

good = marks2[0];

Ivan = Oleg;

Console.WriteLine("Элемент структуры после копирования:");

good.DisplayValue();

Console.Write("\n");

Console.WriteLine("Элемент класса после копирования:");

Ivan.DisplayInfo();

Console.Write("\n\n");

Console.WriteLine("Если изменить те объекты, на которые менялись объект класса и структуры:");

marks2[0].Set(3);

Oleg.Set("Погребнеков", "Николай", "Михайлович");

marks2[0].DisplayValue();

Oleg.DisplayInfo();

Console.Write("\n");

Console.WriteLine("То новые объектыбудут таковыми:");

good.DisplayValue();

Ivan.DisplayInfo();

Console.WriteLine("Объект класса изменился, потому что копируется ссылка \nна объект, а объект структуры не изменился, т.к. \nкопируются значения полей.");

Console.Write("\n\n");

Console.Write("13) Продемонстрировать работу с массивом объектов\n");

Console.Write("==================================================================\n");

Console.Write("На примере структуры Оценка\n");

Mark[][] marksTest = new Mark[1][];

marksTest[0] = new Mark[5];

Console.Write("Массив оценок:\n");

for (j = 0; j < 5; j++)

{

marksTest[0][j].Set(j + 1);

marksTest[0][j].DisplayValue();

}

Console.Write("\n\n");

Console.Write("14) Продемонстрировать возврат значения через параметр out и через\nпараметр ref. Показать различие этих механизмов\n");

Console.Write("==================================================================\n");

Console.Write("На примере структуры оценка\n");

// Инициализация объектов структуры

Mark markForRef = new Mark();

Mark markForOut = new Mark();

// Инициализация переменной для передачи по ref

int ArgRef = 2;

// Описание переменной для передачи по out

int ArgOut;

markForRef.Set(1);

// Передача аргумента по ref

markForRef.TestRef4(ref ArgRef);

Console.WriteLine("Аргумент функции после использования по ref: " + ArgRef);

markForOut.Set(2);

// Передача аргумента по out

markForOut.TestOut5(out ArgOut);

Console.WriteLine("Аргумент функции после использования по out: " + ArgOut);

Console.Write("\n\n");

Console.Write("15) Продемонстрировать разумное использование оператора this\n");

Console.Write("==================================================================\n");

Console.Write("На примере класса Учитель, двусвязанный список\n\n");

// Формирование объектов класса Teacher:

teachers[0].Set("Иванова", "Марина", "Витальевна");

teachers[1].Set("Погребнеков", "Николай", "Михайлович");

teachers[2].Set("Баянова", "Людмила", "Анатольевна");

Teacher.NewList();

// Вызов статической компанентной функции:

Teacher.reprint();

// Включение созданных компанентов в двусвязанный список:

teachers[0].Add(); teachers[1].Add(); teachers[2].Add();

// Печать в обратном порядке значений элементов списка:

Teacher.reprint();

Console.Write("\n\n");

Console.Write("16) Продемонстрировать перегрузку операторов '+', '++'\n");

Console.Write("==================================================================\n");

Console.Write("На примере структуры Оценка\n");

Mark mark1 = new Mark();

Mark mark2 = new Mark();

Console.Write("Оценка 1 и оценка 2:\n");

mark1.Set(1);

mark2.Set(2);

mark1.DisplayValue();

mark2.DisplayValue();

Console.Write("Сумма оценок через оператор \'+\':\n");

mark1 = mark1 + mark2;

mark2 = mark1;

mark1.DisplayValue();

Console.Write("Прибавление к сумме 1 через постфиксный оператор \'++\': ");

(mark1++).DisplayValue();

Console.Write("Прибавление к сумме 1 через префиксный оператор \'++\': ");

(++mark2).DisplayValue();

Console.Write("\n\n");

Console.Write("17) Продемонстрировать обработку строк (string)\n");

Console.Write("==================================================================\n");

Console.Write("На примере класса Учитель, добавление ко всем полям пробелов\n");

teachers[0].AdditionSapces();

teachers[0].DisplayInfo();

Console.Write("==================================================================\n");

//===========================================================================================================

\*/

// Лабораторная №8

//===========================================================================================================

Console.Write("Лабраторная 8:\n\n");

Console.Write("=========\nМодифицировать ваши проекты на С++, C# и Java путем добавления в один из классов\nкак минимум одного статического поля и одного статического метода.\n=========\n\n");

Console.Write("(Использовались статический член класса\'Teacher\' \'lastTeacher\'и\nстатические методы класса \'NewList\' и \'reprint\')\n\n");

// Формирование объектов класса Teacher:

teachers[0].Set("Иванова", "Марина", "Витальевна");

teachers[1].Set("Погребнеков", "Николай", "Михайлович");

teachers[2].Set("Баянова", "Людмила", "Анатольевна");

Console.Write("Вызов статической компанентной функции: \'Новый список\'\n\n");

// Вызов статической компанентной функции:

Teacher.NewList();

Console.Write("Вызов статической компанентной функции: \'Напечатать список\'\n\n");

// Вызов статической компанентной функции:

Teacher.reprint();

Console.Write("\n");

Console.Write("Добавление элементов в список.\n\n");

// Включение созданных компанентов в двусвязанный список:

teachers[0].Add(); teachers[1].Add(); teachers[2].Add();

Console.Write("Вызов статической компанентной функции: \'Напечатать список\'\n\n");

Console.Write("===================================\n");

// Печать в обратном порядке значений элементов списка:

Teacher.reprint();

Console.Write("===================================\n");

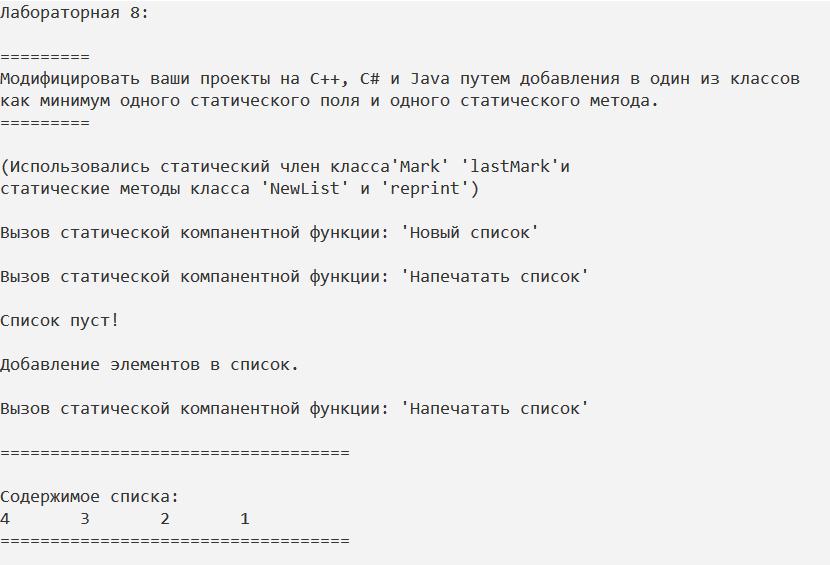
}

}

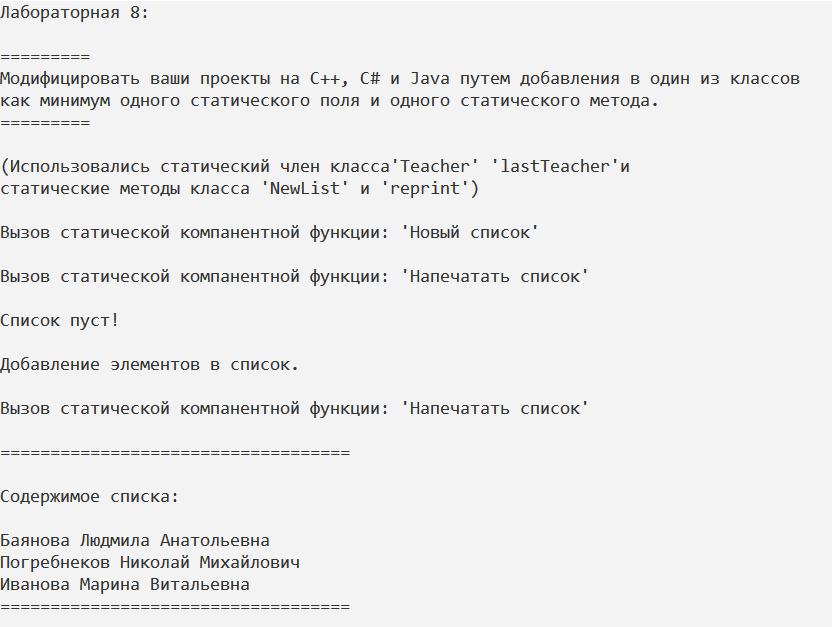
}

Приложение Б. Тесты программ.

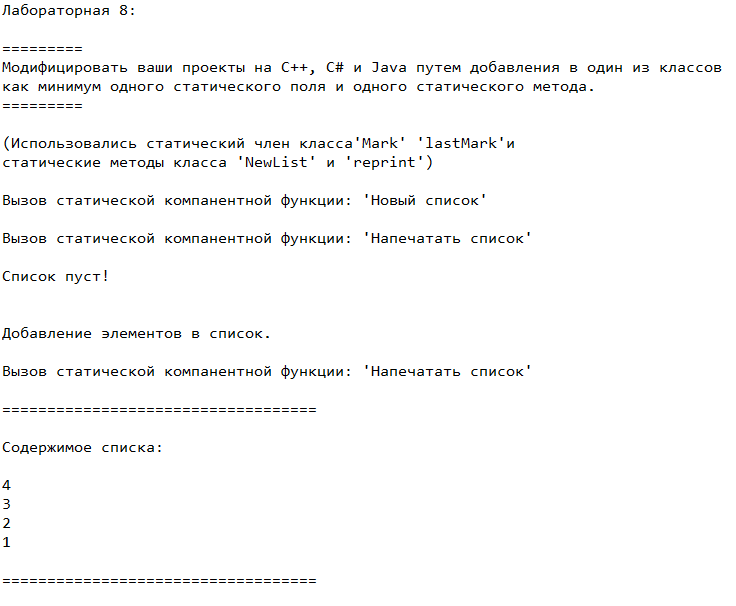
C++



C#



Java



Приложение В. Ссылки на GitHub.

C++

[AlexGr5/lab2\_t2 at lab8 (github.com)](https://github.com/AlexGr5/lab2_t2/tree/lab8)

Java

[AlexGr5/Lab4-Java at Lab8 (github.com)](https://github.com/AlexGr5/Lab4-Java/tree/Lab8)

C#

[AlexGr5/lab6-csh at Lab8 (github.com)](https://github.com/AlexGr5/lab6-csh/tree/Lab8)