МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Уровень образования | |  | Высшее образование – бакалавриат | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Информационные технологии и программирование | | |
| Курс | I | | | Группа | 4306 |

**Отчёт по контрольной работе № 2**

**Вариант № 7**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 425 |  |  |  | Васильченко Мария Сергеевна |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверил(и): |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Федин Алексей Константинович |
|  |  |  |  |

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc133567454)

[2 Исходные данные 3](#_Toc133567455)

[3 Особые ситуации 3](#_Toc133567456)

[4 Математические методы и алгоритмы решения задач 3](#_Toc133567457)

[5 Блок-схемы алгоритмов программы 4](#_Toc133567458)

[6 Форматы представления данных 6](#_Toc133567459)

[7 Структура программы 8](#_Toc133567460)

[8 Описание хода выполнения контрольной работы 12](#_Toc133567461)

[9 Результат работы программы 13](#_Toc133567462)

[10 Выводы по заданию 17](#_Toc133567463)

[11 Исходный текст программы 17](#_Toc133567464)

# 1 Постановка задачи

Необходимо разработать Базовый класс – средство передвижения. Поля в нем: вес, мощность мотора, скорость. Производный класс – самолет; производный класс второго поколения – военный самолет, транспортный самолет..

# 2 Исходные данные

В качестве исходных данных программа использует введенные пользователем и/или считаные из файла данные о студентах.

# 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

* Ввод пользователем некорректного имени файла
* Загрузка некорректных данных с файла
* Некорректный ввод значений переменных

# 4 Математические методы и алгоритмы решения задач

При отборе студентов по уникальному ключу используется метод перебора. Программа определённое количество итераций сравнивает уникальные ключи студентов с ключом введенным пользователем, хранящимися в массиве объектов, и если находит нужного студента, то записывает объект в новый массив и присваивает статус либо «Отчислен», либо «На степендии», если статус не был устоновлен пользователем, то «Без степендии».

При расчете среднего балла группы студента используется метод перебора. Программа определённое количество итераций сравнивает наименование группы студентов с наименованием введенным пользователем, и если находит, то складывает средние баллы студентов и делит на их колличество.

# 5 Блок-схемы алгоритмов программы



Рисунок 1 – Блок-схема алгоритма отбора по заданному наименованию.



Рисунок 2 – Блок-схема алгоритма отбора по заданному наименованию и цене не выше указанной.

# 6 Форматы представления данных

Программа использует следующие переменные:

Таблица 1 — Список переменных, используемых в программе

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| variant | int | Пользовательский выбор во всех меню |
| boarMenu | int | Граница допустимых значений пользовательского ввода в меню |
| people | vector<shared\_ptr<Person>> | Контейнер, содержащий умные указатели на родительский класс |
| list | vector<shared\_ptr<Person>> | Контейнер, содержащий умные указатели на родительский класс |
| mUI | ui | Экземпляр класса ui, для вызовов методов класса |
| mReader | fileReader | Экземпляр класса fileReader, для вызовов методов класса |
| mTest | test | Экземпляр класса test, для вызовов методов класса |
| name | string | Имя файла, которое задает пользователь |
| myFile | ofstream | Вывод данных в файл |
| input | ifstream | Ввод данных из файла |
| isDataSaved | bool | Проверка на то, сохранились ли данные в файл |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| data | string | Считываемая строка из файла |
| token | string | Считываемые подстроки из файла, нужные для заполнения полей класса |
| way | string | В зависимости от передаваемой строки выводится информация |
| mInput | string | Вывод информации для ввода |
| mOutput | string | Вывод информации для вывода |
| out | ofstream | Вывод данных для тестовых файлов |
| out2 |
| out3 |
| out4 |
| out5 |
| sum | double | Сумма средних баллов студентов, состояящих в одной группе |
| average | double | Средний балл группы |
| tokenSearche | string | Уникальный код студента введенный пользователем |

Для хранения данных о студентах используются объекты класса Student, наследованные от класса Person. В Student задаются фамилия, имя, средний балл, группа, балл группы и генерируется токен. Модульные тесты описаны как методы класса test, представлено 5 различных наборов данных, которые проверяют 6 основных алгоритмов программы — ввод данных из файла, заполнение полей класса подстроками, обнуление объектов с отрицательными или нулевыми полями, отбор студентов по заданному условию, отбор и инцилизация среднего балла группы.

# 7 Структура программы

Программа разделена на 6 исходных и 5 заголовочных файлов.

Таблица 2 — Список заголовочных файлов

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Person.h | Содержит объявления абстрактного класса Person и его чисто абстрактных методов. |
| Student.h | Содержит объявления класса наследника Student от абстрактного класса родителяя Person. |
| fileReaser.h | Содержит объявления класса fileReader и его методов для работы с файлами. |
| selection.h | Содержит объявления класса selection и его методов отбора нужных по условиям студентов. |
| ui.h | Содержит объявления класса ui и consoleInOut и их методы для работы с консольным вводом и выводом. |
| test.h | Содержит объявления класса test и его методов для выполнения модульных тестов. |

Таблица 3 — Список исходных файлов

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| main.cpp | Содержит определения функций запуска программы и механизма работы зацикленных меню |
| fileReader.cpp | Содержит определения функций для работы с файлами |
| consoleInOut | Содержит определения функций для работы с консольным вводом и выводом |
| ui.cpp | Содержит определения функций проверки коректности введенных пользователем данных |
| selection.cpp | Содержит определения функций отбора нужных по заданию студентов |

Продолжение таблицы 3

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Person.cpp | Содержит иницилизацию уникального id для каждого студента |
| Student.cpp | Содержит определение функции для инициализации полей класса наследника Student и родителя Person и для получения полей. |
| test.cpp | Содержит определение функций реализации модульных тестов программы |

Таблица 4 – Функции, составляющие программу

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| launchMenu | Реализация внешнего зацикленного меню и 2-х вложенных зацикленных меню. |
| selectInput | Выбор способа ввода данных |
| selectImplTask | Выбор способа иницилизации поля статус для студентов в зависимости от условий |
| selectOutput | Выбор способа вывода отобранных студентов |
| printInputWays | Вывод информации по навигации в меню способов ввода данных |
| printAction | Вывод информации по навигации в меню способов иницилизации и отбора данных |
| printOutputWays | Вывод информации по навигации в меню способов вывода полученного контейнера студентов |
| outConsole | Реализует вывод студентов в консоль |
| printProduct | Реализует вывод полей студентов в консоли |

Продолжение таблицы 4

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| filling | Реализует заполнение полей класса переданными данными |
| getDouble | Вызывает функцию getInput требуя вернуть ввод типа double. |
| getInt | Вызывает функцию getInput требуя вернуть ввод типа int. |
| getBool | Вызывает функцию getInput требуя вернуть ввод типа bool. |
| getSize | Возвращает ввод размера массива типа int и зацикливается если ввод неположительный |
| getMenuVar | Возвращает ввод пункта меню типа int и зацикливается если ввод не соответствует пунктам меню |
| getInput | Возвращает ввод числа переданного типа и зацикливается если ввод некорректен |
| fillFile | Реализует заполнение файла данными об отобранных студентах |
| readFile | Реализует чтение и проверку данных из файла в массив. |
| removeNegative | Обнуляет объекты с пустыми полями (фамилия, адресс, группа) и некорректным средним баллом |
| implFile | Реализует чтение данных из файла. |

Продолжение таблицы 4

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| filling | Реализует заполнения полей класса переданными данными. |
| printInform | Информирует пользователя о формате файла и данных в нем. |
| startTest | Запускает функции тестов. |
| testRemoveNegative | Тестирует функцию removeNegative |
| fillingForTest | Заполняет тестовыми данными поля класса |
| testParsingTXT | Тестирования функции filling |
| testImplFile | Тестирование функции implFile |
| creatingTestFiles | Создание тестовых файлов |
| checkingOutput | Проверяет был ли выведен предупредительный текст о некорректности введенных данных в файле |
| checkingData | Проверяет соответствует ли полученные данные ожидаемым |
| printCorrect | Проверяет отличались ли полученные данные от ожидаемых, если нет, информирует о пройденных тестов. |
| printInvalid | Информирует пользователя об ошибках в модульном тестировании |
| selectStudent | Иницилизация поля статус у отобранных студентов по уникальному коду. |
| expelledStudent | Удаление студентов со статусом «Отчислен» |
| initGroupAverage | Иницилизация среднего балла группы у студентов, находящихся в заданной группе. |
| printInputInfo | Информирования об необходимости ввода |

Таблица 5 – Методы классов, составляющих программу

|  |  |  |
| --- | --- | --- |
| **Имя** | **Класс** | **Описание** |
| Person | Person | При вызове конструктора переменная id увеличивается на 1 |
| getSurname | Person | Чисто абстрактный метод |
| setSurname | Person | Чисто абстрактный метод |
| getToken | Person | Чисто абстрактный метод |
| getAddress | Person | Чисто абстрактный метод |
| setAddress | Person | Чисто абстрактный метод |
| isPersonExist | Person | Чисто абстрактный метод |
| getToken | Student | Получить уникальный код студента |
| getSurname | Student | Получить фамилию студента |
| getAddress | Student | Получить адресс студента |
| getGroup | Student | Получить группу студента |
| getGPA | Student | Получить средний балл студента |
| setSurname | Student | Установить фамилию студента |
| setGPA | Student | Установаить средний балл студента |
| isPersonExist | Student | Проверят, не путсые ли поля класса и коректный ли средний балл студентов |
| setGroupAverage | Student | Устанавливает средний балл группы |
| getGroupAverage | Student | Получить средний балл группы |
| setStatus | Student | Установить статус студента |
| getStatus | Student | Получить средний балл студента |

# 8 Описание хода выполнения контрольной работы

1. В ходе контрольной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio C++ 2022. В нём был создан проект. Так же проект был размещен на github в публичном репозитории, в котором отслеживались все commit и исправления в проекте.
2. В начальном варианте программы не использовались конструкции try-catch из-за чего функции считывания данных из файла использовали большое количество bool переменных (флагов). С изучением данной конструкции надобность флагов пропала.
3. Были изучены парадигмы наследования и полиморфизма для реализации абстрактного класса и чисто виртуальных методов в нем.
4. В начальном варианте программы хранение данных о студентах было реализовано при помощи динамического выделения памяти с помощью ключегого слова new. Был осуществлен переход на хранение данных в контейнере vector типа класса наследника. В дальнешем контенер стал хранить в себе умные указатели на класс родителя.
5. При переходе на контенеры изчезла необходимость постоянно передавать отдельную переменную содержащую количество объектов класса в массиве.

# 9 Результат работы программы

При выборе соответствующих пунктов меню, в результате, программа выводит список студентов с установленными статусами и с установленным средним баллом группы. Так же реализована возможность исключить отчисленных студентов из списка и вывести итоговый.

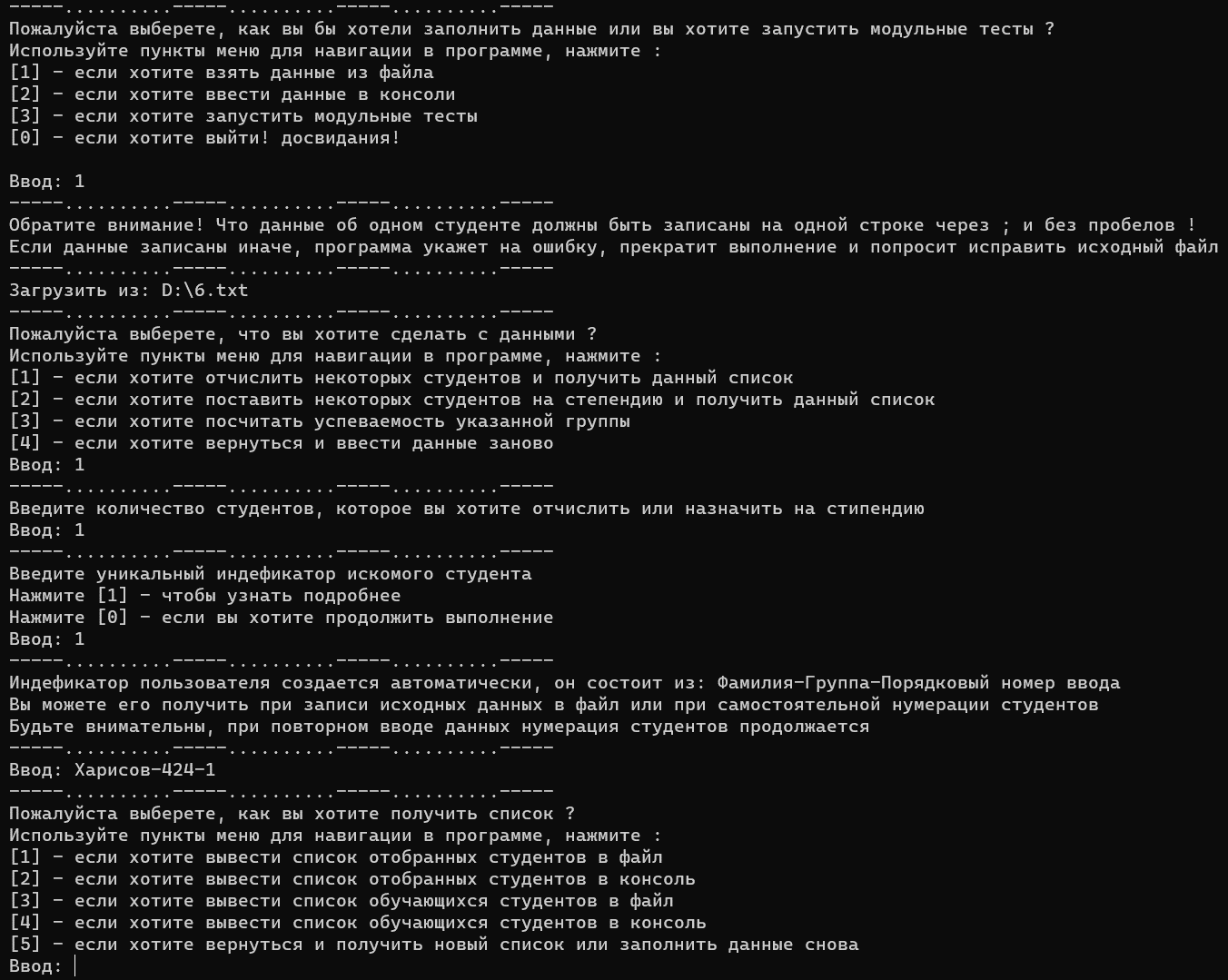


Рисунок 3 – Выбор способа вывода списка

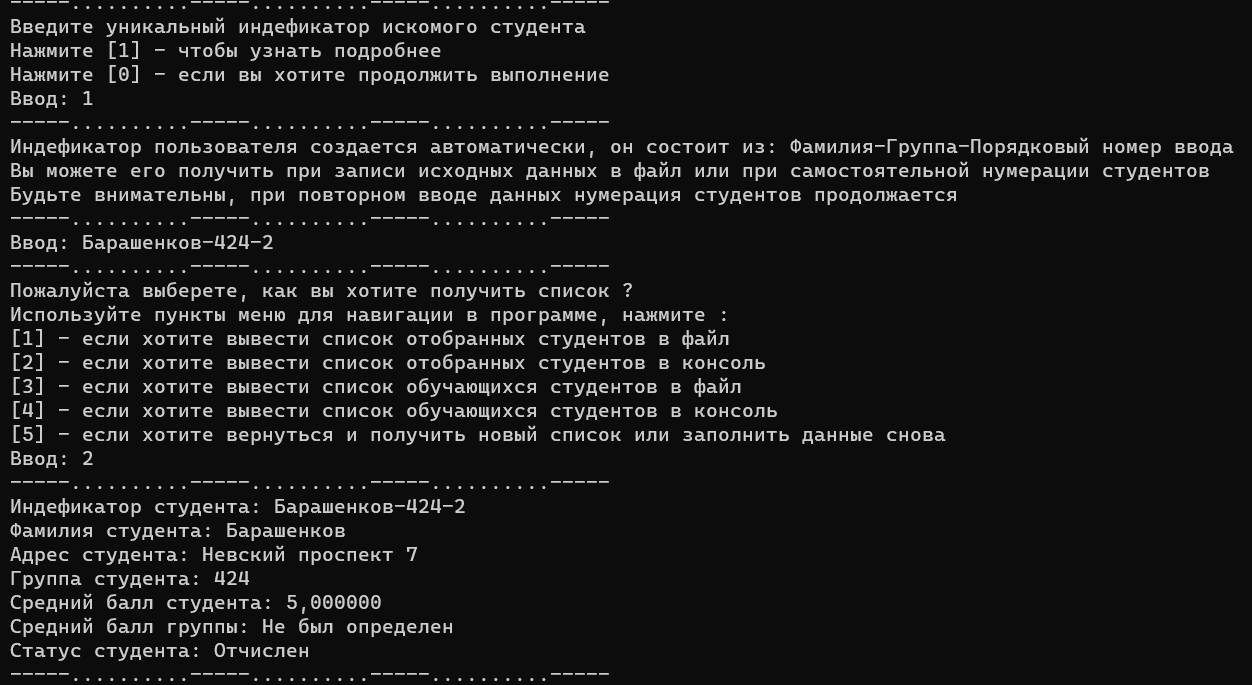


Рисунок 4 – Пример отчисления студента

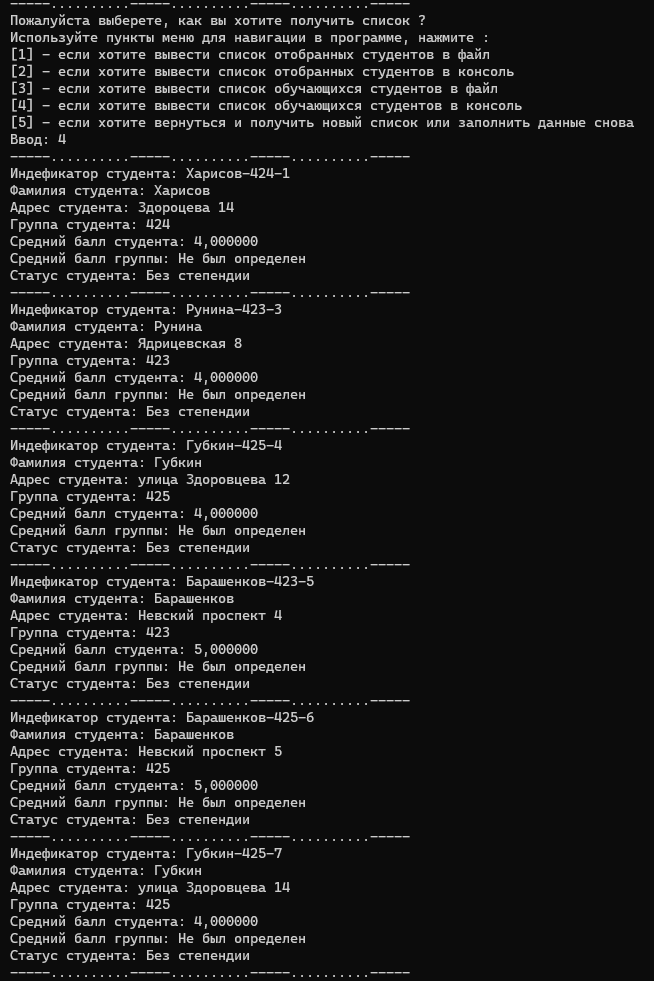


Рисунок 5 – Пример получения списка обучающихся студентов (не отчисленных)

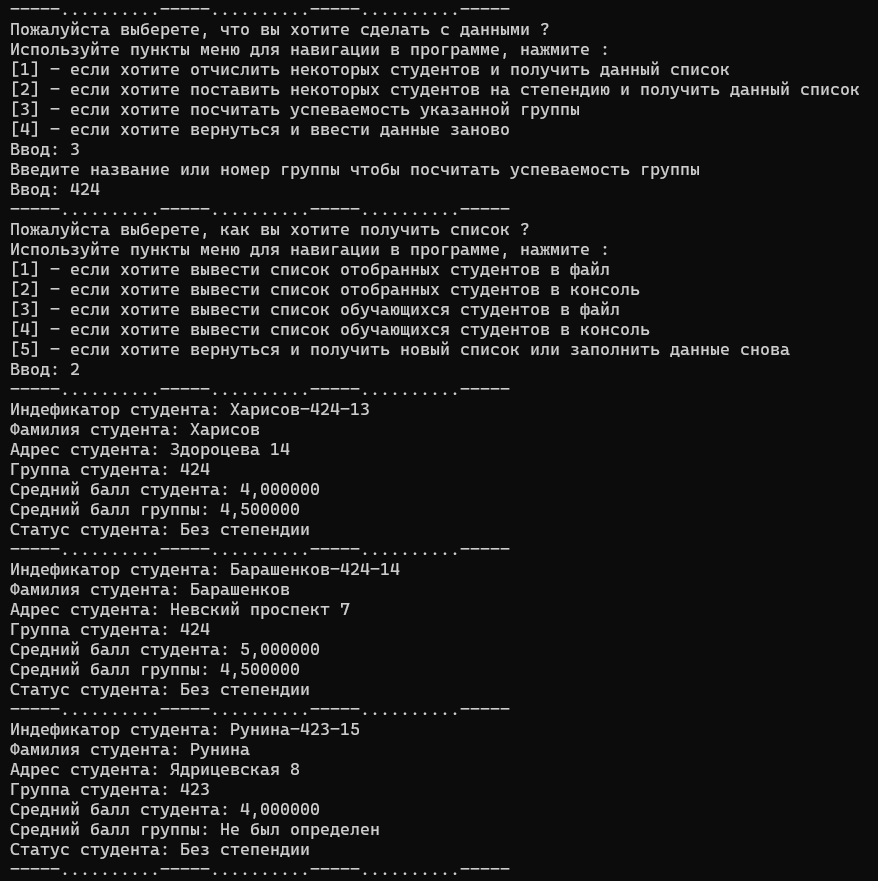


Рисунок 6 – Пример расчета среднего балла группы

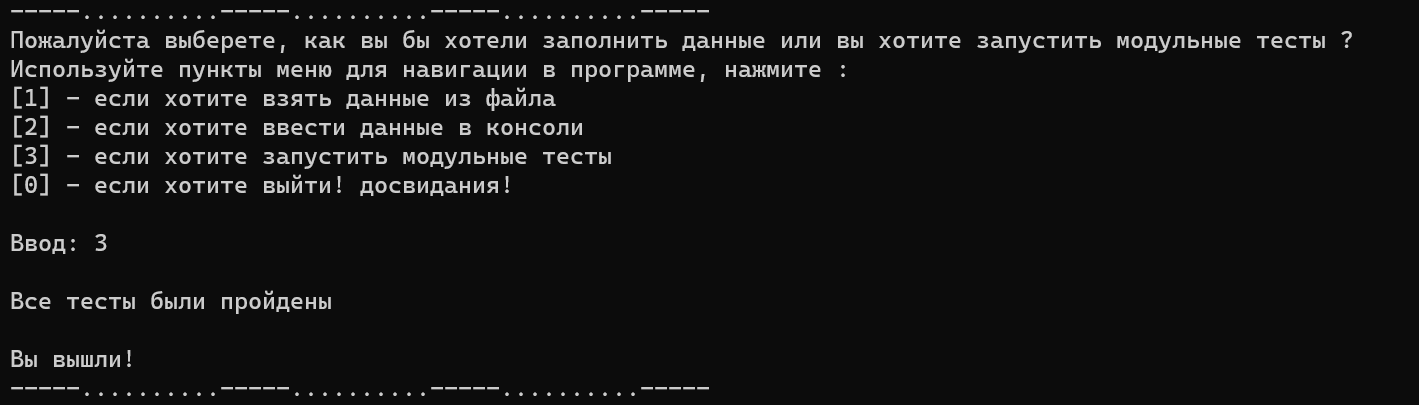


Рисунок 7 – Пример модульных тестов

# 10 Выводы по заданию

Задача, связанная с созданием программы для работы с классами наследниками и контейнирами с указателями на класс родителя. Решение этой задачи может вызвать некоторые трудности при переходе на новый язык программирования, но наличие опыта работы с другими языками поможет избежать простых ошибок. При возникновении проблем с недостатками знаний можно использовать различные ресурсы, такие как видеоуроки и тематические статьи, чтобы улучшить свои навыки программирования. Решение задачи может стать хорошим способом для тех, кто хочет научиться программированию на C++ и расширить свои навыки в этой области.

# 11 Исходный текст программы

**[ Начало программы ]**

**[ Начало файла main.cpp ]**

#include <iostream>

#include <vector>

#include <locale>

#include <cstdlib>

#include "fileReader.h"

#include "Student.h"

#include "Person.h"

#include "ui.h"

#include "test.h"

#include "selection.h"

using namespace UI;

using namespace reader;

using namespace testUnit;

using namespace studt;

using namespace select;

enum { task1 = 1, task2 = 2, task3 = 3, returnp = 4};

enum { exitp = 0, rfile = 1, rconsole = 2, unitTest = 3 };

enum { fileOutput = 1, coutput = 2, fnotExpoled = 3, cnotExpolled = 4, outreturn = 5};

static string expelled = "Отчислен";

static string scholarships = "На степендии";

void printInputWays() {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Пожалуйста выберете, как вы бы хотели заполнить данные или вы хотите запустить модульные тесты ?" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - если хотите взять данные из файла" << endl;

cout << "[2] - если хотите ввести данные в консоли" << endl;

cout << "[3] - если хотите запустить модульные тесты" << endl;

cout << "[0] - если хотите выйти! досвидания!" << endl << endl;

}

void printAction() {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Пожалуйста выберете, что вы хотите сделать с данными ?" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - если хотите отчислить некоторых студентов и получить данный список" << endl;

cout << "[2] - если хотите поставить некоторых студентов на степендию и получить данный список" << endl;

cout << "[3] - если хотите посчитать успеваемость указанной группы" << endl;

cout << "[4] - если хотите вернуться и ввести данные заново" << endl;

}

void printOutputWays() {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Пожалуйста выберете, как вы хотите получить список ?" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - если хотите вывести список отобранных студентов в файл" << endl;

cout << "[2] - если хотите вывести список отобранных студентов в консоль" << endl;

cout << "[3] - если хотите вывести список обучающихся студентов в файл" << endl;

cout << "[4] - если хотите вывести список обучающихся студентов в консоль" << endl;

cout << "[5] - если хотите вернуться и получить новый список или заполнить данные снова" << endl;

}

/\*

Функциональность программы:

1) мы можем отобрать в новый список отчисленных студентов, неотчисленных и тех

кто остался на стипендии - дальше мы можем вывести этот список и тд

2) мы можем посчитать средний бал группы - записать его в файл прикрепляя к каждой группе.

то есть например просим пользователя ввести название группы и почитать для нее средний балл

а потом добавляем его к данным, но не добавляем к массиву объектов.

\*/

vector<shared\_ptr<Person>> selectImplTask(int variant, vector<shared\_ptr<Person>>& students) {

selection selected{};

switch (variant)

{

case task1:

return selected.selectStudent(students, expelled);

case task2:

return selected.selectStudent(students, scholarships);

case task3:

selected.initGroupAverage(students);

return students;

case returnp:

cout << "Вы вышли! " << endl;

return students;

default:

return students;

}

}

void selectInput(int variant, vector<shared\_ptr<Person>>& students) {

ui mUI{};

consoleInOut cInOut{};

fileReader mReader{};

test mTest{};

switch (variant)

{

case rfile:

mReader.readFile(students, {});

break;

case rconsole:

cInOut.readConsole(students);

break;

case unitTest:

mTest.startTest();

case exitp:

cout << "Вы вышли!" << endl;

break;

}

}

void selectOutput(int variant, vector<shared\_ptr<Person>>& list, vector<shared\_ptr<Person>>& students) {

ui mUI{};

consoleInOut cInOut{};

fileReader mReader{};

test mTest{};

switch (variant)

{

case fileOutput:

mReader.fillFile(list, false);

break;

case coutput:

cInOut.outConsole(list, false);

break;

case fnotExpoled:

mReader.fillFile(students, true);

break;

case cnotExpolled:

cInOut.outConsole(students, true);

break;

case outreturn:

cout << "Вы вышли!" << endl;

break;

}

}

void launchMenu() {

ui mUI{};

int variant = 0;

int boardMenu = unitTest;

vector<shared\_ptr<Person>> people;

vector<shared\_ptr<Person>> list;

do

{

printInputWays();

variant = mUI.getMenuVar(exitp, unitTest);

try {

selectInput(variant, people);

}

catch (const exception& e) {

cout << e.what() << endl;

continue;

}

if (variant == exitp) break;

if (variant == unitTest) continue;

do {

printAction();

boardMenu = returnp;

variant = mUI.getMenuVar(task1, boardMenu);

if (variant == returnp) break;

try {

list = selectImplTask(variant, people);

}

catch (const exception& e) {

cout << e.what() << endl;

continue;

}

do {

printOutputWays();

boardMenu = outreturn;

variant = mUI.getMenuVar(fileOutput, boardMenu);

if (variant == outreturn) break;

try {

selectOutput(variant, list, people);

}

catch (const exception& e) {

cout << e.what() << endl;

continue;

}

} while (variant != outreturn);

} while (variant != returnp);

people.clear();

list.clear();

} while (variant != exitp);

}

void outGretting() {

cout << "Добро пожаловать в програмную реализацию Контрольной работы №2" << endl;

cout << "Лабораторную работу выполнил: Харисов Ильяс Ренатович, 424 группа, 21 вариант" << endl << endl;

cout << "Что нужно было сделать ?" << endl;

cout << "Контрольная работа № 2. Наследование" << endl

<< "В контрольной работе задается базовый и производный класс." << endl

<< "Необходимо разработать поля и методы, наследуемые из базового класса." << endl

<< "A также собственные компоненты производных классов.Базовый класс может быть абстрактным." << endl

<< "Реализовать возможность получения списка объектов в контейнере." << endl;

cout << "Базовый класс – личность; поля – фамилия, адрес. Производный класс – студент." << endl;

}

int main() {

system("chcp 1251");

setlocale(LC\_ALL, "Rus");

outGretting();

launchMenu();

system("pause");

}

**[ Конец файла main.cpp ]**

**[ Начало файла fileReader.h ]**

#pragma once

#include "Student.h"

#include "Person.h"

using namespace studt;

namespace reader {

class fileReader {

public:

void readFile(vector<shared\_ptr<Person>>&, string);

void fillFile(vector<shared\_ptr<Person>>&, bool);

void implFile(vector<shared\_ptr<Person>>&, string);

void filling(shared\_ptr<Student>, const char\*, int);

void removeNegative(vector<shared\_ptr<Person>>&);

};

}

**[ Конец файла fileReader.h ]**

**[ Начало файла fileReader.cpp ]**

#include <iostream>

#include <filesystem>

#include "fileReader.h"

#include "ui.h"

#include "Student.h"

#include "selection.h"

#include <string>

#include <fstream>

#include <cstring>

#include <iomanip>

#include <cmath>

using namespace std;

using namespace reader;

using namespace UI;

using namespace studt;

static const string mInput = "input";

static const string mOutput = "output";

enum {ch\_surname = 0, ch\_adress = 1, ch\_group = 2, ch\_gpa = 3};

void printInform(string way) {

if (way == mInput) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Обратите внимание! Что данные об одном студенте должны быть записаны на одной строке через ; и без пробелов !" << endl;

cout << "Если данные записаны иначе, программа укажет на ошибку, прекратит выполнение и попросит исправить исходный файл" << endl;

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Загрузить из: ";

}

if (way == mOutput) {

cout << "Введите название файла в который вы хотите сохранить результат" << endl;

}

}

// Разбиваем строку на подстроки, используя символ ';' в качестве разделителя

void fileReader::filling(shared\_ptr<Student> student, const char\* str\_ptr, int i) {

int iter = 0;

double gpa{};

char\* str\_copy = new char[strlen(str\_ptr) + 1];

strcpy\_s(str\_copy, strlen(str\_ptr) + 1, str\_ptr);

char\* context = nullptr;

char\* token = strtok\_s(str\_copy, ";", &context);

while (token != nullptr) {

switch (iter) {

case ch\_surname:

student->setSurname(token);

break;

case ch\_adress:

student->setAdress(token);

break;

case ch\_group:

student->setGroup(token);

break;

case ch\_gpa:

gpa = stod(token);

if (gpa <= 0 && gpa > 5) {

cout << "Некорректный средний балл студента! Исправьте данные в файле! В " << i + 1 << " строке" << endl;

}

student->setGPA(gpa);

break;

default:

cout << "Слишком много данных в одной строке! Данные будут проигнорированы" << endl;

}

token = strtok\_s(nullptr, ";", &context);

iter++;

}

delete[] str\_copy;

}

void fileReader::implFile(vector<shared\_ptr<Person>>& people, string name) {

int i = 0;

string data{};

ifstream input;

input.open(name);

if (input.is\_open()) {

while (getline(input, data)) {

if (data.empty()) {

cout << "Ввод данных о студенте на " << i + 1 << " строке не был произведен! Исправьте файл и введите данные заново" << endl;

i++;

continue;

}

else {

shared\_ptr<Student> student = make\_shared<Student>();

filling(student, data.c\_str(), i);

people.push\_back(student);

}

i++;

}

input.close();

}

else {

cout << endl << "Файла с таким именем не существует!" << endl << endl;

throw runtime\_error("error");

}

}

void fileReader::removeNegative(vector<shared\_ptr<Person>>& people) {

people.erase(remove\_if(people.begin(), people.end(),

[&](const shared\_ptr<Person>& person) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

return student && !(student->isPersonExist());

}), people.end());

}

void fileReader::readFile(vector<shared\_ptr<Person>>& people, string name) {

printInform(mInput);

cin >> name;

implFile(people, name);

removeNegative(people);

}

void fileReader::fillFile(vector<shared\_ptr<Person>>& people, bool getNonExcluded) {

printInform(mOutput);

string filePath = "";

bool isDataSaved = false;

ui mUI;

do {

cout << "Сохранить в: ";

getline(cin, filePath);

if (ifstream(filePath)) {

cout << "Файл уже существует." << endl;

cout << "[0] - Перезаписать существующий файл." << endl;

cout << "[1] - Повторить ввод." << endl;

int tryAnotherFile = mUI.getMenuVar(0, 1);

if (tryAnotherFile) {

continue;

}

}

ofstream myFile(filePath, ofstream::app);

error\_code ec{};

if (!myFile) {

cout << "Запись запрещена. Повторите ввод." << endl;

myFile.close();

continue;

}

myFile.close();

myFile.open(filePath, ofstream::trunc);

if (getNonExcluded) {

select::selection sl{};

sl.expelledStudent(people);

}

for (auto& person : people) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

myFile << "Индефикатор студента: " << student->getToken() << endl;

myFile << "Фамилия студента: " << student->getSurname() << endl;

myFile << "Адрес студента: " << student->getAddress() << endl;

myFile << "Группа студента: " << student->getGroup() << endl;

myFile << "Средний балл студента: " << to\_string(student->getGPA()) << endl;

myFile << "Средний балл группы: " << (student->getGroupAverage() == 0 ? "Не был определен" : to\_string(student->getGroupAverage())) << endl;

myFile << "Статус студента: " << student->getStatus() << endl;

myFile << "-----..........-----..........-----..........-----" << endl;

}

myFile.close();

cout << "Запись завершена." << endl;

isDataSaved = true;

} while (!isDataSaved);

}

**[ Конец файла fileReader.cpp ]**

**[ Начало файла Person.h ]**

#pragma once

#include <string>

using namespace std;

class Person {

protected:

string surname{};

string address{};

int id{};

private:

static int counter;

public:

virtual ~Person() = default;

Person() {

id = ++Person::counter;

}

virtual string getToken() const = 0;

virtual string getSurname() const = 0;

virtual string getAddress() const = 0;

virtual void setSurname(string) = 0;

virtual void setAdress(string) = 0;

virtual bool isPersonExist() const = 0;

};

**[ Конец файла Person.h ]**

**[ Начало файла Person.cpp ]**

#include "Person.h"

int Person::counter = 0;

**[ Конец файла Person.cpp ]**

**[ Начало файла Student.h ]**

#pragma once

#include <string>

#include <vector>

#include "Person.h"

namespace studt {

class Student : public Person {

private:

string group{};

double GPA{};

double average{};

string status{};

public:

string getToken() const;

string getSurname() const;

string getAddress() const;

string getGroup() const;

double getGPA() const;

void setSurname(string);

void setAdress(string);

void setGroup(string);

void setGPA(double);

bool isPersonExist() const;

void setGroupAverage(double);

double getGroupAverage() const;

void setStatus(string);

string getStatus() const;

void setSurnameCorrect(string);

void setAddressCorrect(string );

void setGroupCorrect(string);

void setGPACorrect(double);

};

}

**[ Конец файла Student.h ]**

**[ Начало файла Student.cpp ]**

#include "Student.h"

#include "ui.h"

#include <iostream>

using namespace studt;

using namespace std;

string Student::getToken() const{

return surname + "-" + group + "-" + to\_string(id);

}

string Student::getAddress() const{

return address;

}

string Student::getSurname() const{

return surname;

}

string Student::getGroup() const{

return group;

}

double Student::getGPA() const{

return GPA;

}

void Student::setSurname(string mSurname){

surname = mSurname;

}

void Student::setAdress(string mAdress) {

address = mAdress;

}

void Student::setGroup(string mGroup) {

group = mGroup;

}

void Student::setGPA(double gpa) {

GPA = gpa;

}

void Student::setStatus(string mStatus) {

status = mStatus;

}

string Student::getStatus() const {

return status.empty() ? "Без степендии" : status;

}

void Student::setGroupAverage(double mAverage) {

average = mAverage;

}

double Student::getGroupAverage() const {

return average;

}

bool Student::isPersonExist() const {

bool exist = ((GPA >= 1 && GPA <= 5) || !surname.empty() ||

!address.empty() ||

!group.empty());

return exist;

}

string getInput(string field) {

while (field.empty()) {

cout << "Введенна пустая строка" << endl;

cout << "Ввод: ";

getline(cin, field);

}

return field;

}

void Student::setSurnameCorrect(string mSurname){

surname = getInput(mSurname);

}

void Student::setAddressCorrect(string mAddress) {

address = getInput(mAddress);

}

void Student::setGroupCorrect(string mGroup) {

group = getInput(mGroup);

}

void Student::setGPACorrect(double gpa) {

UI::ui ui{};

bool inputCorrect = gpa <= 0 && gpa > 5;

while (!inputCorrect) {

cout << "Некорректный средний балл! Он должен находится в диапозоне: [1; 5]" << endl;

gpa = ui.getDouble();

if (gpa > 0 && gpa <= 5) {

inputCorrect = true;

}

}

GPA = gpa;

}

**[ Конец файла Student.cpp ]**

**[ Начало файла selection.h ]**

#pragma once

#include <vector>

#include "Student.h"

using namespace studt;

namespace select {

class selection {

public:

vector<shared\_ptr<Person>> selectStudent(vector<shared\_ptr<Person>>&, string);

void expelledStudent(vector<shared\_ptr<Person>>&);

void initGroupAverage(vector<shared\_ptr<Person>>&);

};

}

**[ Конец файла selection.h ]**

**[ Начало файла selection.cpp ]**

#include <iostream>

#include "ui.h"

#include "Student.h"

#include "selection.h"

#include <string>

#include <cstring>

#include <vector>

using namespace std;

using namespace UI;

using namespace studt;

using namespace select;

static string inputNumber = "number of student";

static string byToken = "search by surname";

static string getGroupAverage = "get average";

void printInputInfo(string way) {

ui ui{};

if (way == inputNumber) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите количество студентов, которое вы хотите отчислить или назначить на стипендию" << endl;

}

if (way == byToken) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите уникальный индефикатор искомого студента" << endl;

cout << "Нажмите [1] - чтобы узнать подробнее" << endl << "Нажмите [0] - если вы хотите продолжить выполнение" << endl;

if (ui.getBool() == 1) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Индефикатор пользователя создается автоматически, он состоит из: Фамилия-Группа-Порядковый номер ввода" << endl;

cout << "Вы можете его получить при записи исходных данных в файл или при самостоятельной нумерации студентов" << endl;

cout << "Будьте внимательны, при повторном вводе данных нумерация студентов продолжается" << endl;

cout << "-----..........-----..........-----..........-----" << endl;

}

cout << "Ввод: ";

}

if (way == getGroupAverage) {

cout << "Введите название или номер группы чтобы посчитать успеваемость группы" << endl;

cout << "Ввод: ";

}

}

vector<shared\_ptr<Person>> selection::selectStudent(vector<shared\_ptr<Person>>& people, string status) {

ui ui{};

string tokenSearche{};

int numberOfStudent = 0;

printInputInfo(inputNumber);

numberOfStudent = ui.getIntField();

vector<shared\_ptr<Person>> list;

for (int i = 0; i < numberOfStudent; i++) {

printInputInfo(byToken);

getline(cin, tokenSearche);

for (auto& person : people) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

string myToken = person->getToken();

if (myToken == tokenSearche) {

list.push\_back(student);

}

}

}

for (auto& person : list) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

student->setStatus(status);

}

return list;

}

void selection::expelledStudent(vector<shared\_ptr<Person>>& people) {

string removeWithStatus = "Отчислен";

people.erase(remove\_if(people.begin(), people.end(),

[&](const shared\_ptr<Person>& person) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

return student && student->getStatus() == removeWithStatus;

}), people.end());

}

void select::selection::initGroupAverage(vector<shared\_ptr<Person>>& people){

vector<shared\_ptr<Person>> list;

string groupSearche{};

double sum = 0;

double average = 0;

printInputInfo(getGroupAverage);

getline(cin, groupSearche);

for (auto& person : people) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

if (student->getGroup() == groupSearche) {

list.push\_back(student);

sum += student->getGPA();

}

}

average = sum / list.size();

for (const auto& person : list) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

student->setGroupAverage(average);

}

}

**[ Конец файла selection.cpp ]**

**[ Начало файла test.h ]**

#pragma once

namespace testUnit {

class test {

public:

void startTest();

};

}

**[ Конец файла test.h ]**

**[ Начало файла test.cpp ]**

#include <iostream>

#include <cassert>

#include <fstream>

#include <sstream>

#include "test.h"

#include "selection.h"

#include "fileReader.h"

using namespace testUnit;

using namespace reader;

using namespace std;

using namespace studt;

using namespace select;

const string testParsing = "parsing";

const string testFileRead = "read";

enum {

sizeOfTest = 5

};

enum {

withoutEnd = 1, countTest = 6

};

template<typename T, typename U>

void printInvalid(T received, U expected, string message) {

cout << message << endl;

cout << "Ожидалось: " << expected << endl << "Было получено: " << received << endl;

}

bool printCorrect(void (\*function)(), int count) {

stringstream buffer{};

streambuf\* old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

(\*function)();

cout.rdbuf(old\_cout\_buf);

string output = buffer.str();

if (output.empty()) {

return true;

}

else {

cout << "Не был пройден тест №" << count << endl;

cout << "Подробности об ошибке: " << endl;

cout << output;

return false;

}

}

void checkingData(vector<shared\_ptr<Person>>& people, int iter,

string surname, string address, string group, double gpa) {

shared\_ptr<Person> person = people[iter];

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

string message = {};

if (student->getSurname() != surname)

printInvalid(student->getSurname(), surname, message);

if (student->getAddress() != address)

printInvalid(student->getAddress(), address, message);

if (student->getGroup() != group)

printInvalid(student->getGroup(), group, message);

if (student->getGPA() != gpa)

printInvalid(student->getGPA(), gpa, message);

}

void checkingDataStudent(shared\_ptr<Student>& student,

string surname, string address, string group, double gpa) {

string message = {};

if (student->getSurname() != surname)

printInvalid(student->getSurname(), surname, message);

if (student->getAddress() != address)

printInvalid(student->getAddress(), address, message);

if (student->getGroup() != group)

printInvalid(student->getGroup(), group, message);

if (student->getGPA() != gpa)

printInvalid(student->getGPA(), gpa, message);

}

void creatingTestFiles() {

ofstream out("test1.txt");

out << "milkin;n prospeckt 23;425;4,1" << endl;

out << "breader;lenina 189;ОМ123-67;3,9" << endl;

out.close();

ofstream out2("test2.txt");

out2 << "" << endl;

out2 << "" << endl;

out2.close();

ofstream out3("test3.txt");

out3 << "hooks;n prospeckt 23;425;4,1;Extra" << endl;

out3.close();

ofstream out4("test4.txt");

out4 << "hooks;n prospeckt 23;425;4,1;Extra;Extra" << endl;

out4 << "jook;p prospeckt 29;423;-4,9" << endl;

out4 << "" << endl;

out4.close();

}

void testImplFile() {

string myOutput = {};

string message = {};

fileReader reader;

vector<shared\_ptr<Person>> people;

string outputDataError = "Слишком много данных в одной строке! Данные будут проигнорированы\n";

string outputNegative = "Некорректный средний балл студента!Исправьте данные в файле!В 2 строке\n";

string outputInputError3 = "Ввод данных о студенте на 3 строке не был произведен! Исправьте файл и введите данные заново\n";

string outputNullString = "Ввод данных о студенте на 1 строке не был произведен! Исправьте файл и введите данные заново\nВвод данных о студенте на 2 строке не был произведен! Исправьте файл и введите данные заново\n";

creatingTestFiles();

// Тест с корректными данными

stringstream buffer{};

streambuf\* old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(people, "test1.txt");

size\_t size = people.size();

cout.rdbuf(old\_cout\_buf);

string output = buffer.str();

message = "Был не пройден тест на проверку коректных данных\n";

if(size != 2) printInvalid(size, 2, message);

checkingData(people, 0, "milkin", "n prospeckt 23", "425", 4.1);

checkingData(people, 1, "breader", "lenina 189", "ОМ123-67", 3.9);

// Тест с нулевыми строками

people.clear();

vector<shared\_ptr<Person>> people2;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(people2, "test2.txt");

size = people2.size();

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

message = "Был не пройден тест c пустыми строками\n";

if (size != 0)

printInvalid(size, 0, message);

if (output != outputNullString) printInvalid(output, outputNullString, message);

// Тест с некорректным форматом данных

people2.clear();

vector<shared\_ptr<Person>> people3;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(people3, "test3.txt");

size = people3.size();

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

message = "Был не пройден тест с некорректным форматом данных\n";

if(output != outputDataError) printInvalid(output, outputDataError, message);

if(size != 1) printInvalid(size, 1, message);

// Тест с некоректным средним баллом

people3.clear();

vector<shared\_ptr<Person>> people4;

buffer = {};

old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

reader.implFile(people4, "test4.txt");

size = people4.size();

cout.rdbuf(old\_cout\_buf);

output = buffer.str();

message = "Был не пройден тест с некоректным средним баллом\n";

if(output != outputDataError + outputDataError + outputInputError3)

printInvalid(output, outputDataError + outputDataError + outputInputError3, message);

if(size != 2)

printInvalid(size, 2, message);

}

string checkingOutput(shared\_ptr<Student> student, const char\* str, int i) {

fileReader mReader;

stringstream buffer{};

streambuf\* old\_cout\_buf = cout.rdbuf(buffer.rdbuf());

mReader.filling(student, str, i);

cout.rdbuf(old\_cout\_buf);

return buffer.str();

}

void fillingTestData(vector<shared\_ptr<Person>>& people) {

string testSurname[sizeOfTest] = {

"Kharisov",

"Barash",

"Kolin",

"Russia",

"Roeva"

};

string testAddress[sizeOfTest] = {

"Kharisov",

"Barash",

"Kolin",

"Russia",

"Roeva"

};

string testGroup[sizeOfTest] = {

"Kharisov",

"Barash",

"Kolin",

"Russia",

"Roeva"

};

double testGPA[sizeOfTest] = {

3.1,

4.2,

5,

1.9,

2.01

};

for (int i = 0; i < sizeOfTest; i++) {

shared\_ptr<Student> student = make\_shared<Student>();

student->setSurname(testSurname[i]);

student->setAdress(testAddress[i]);

student->setGroup(testGroup[i]);

student->setGPA(testGPA[i]);

people.push\_back(student);

}

}

void test::startTest() {

creatingTestFiles();

int count = 1;

if (

printCorrect(&testImplFile, count++)

) {

cout << endl << "Все тесты были пройдены" << endl << endl;

}

else {

cout << "К глубокому сожалению не все тесты были пройдены" << endl;

}

}

**[ Конец файла test.cpp ]**

**[ Начало файла ui.h ]**

#pragma once

#include "Person.h"

#include <vector>

#include <memory>

using namespace std;

namespace UI {

class ui

{

public:

int getInt();

int getMenuVar(int, int);

int getSize();

double getDouble();

bool getBool();

int getIntField();

double getDoubleField();

};

class consoleInOut

{

public:

void readConsole(vector<shared\_ptr<Person>>&);

void outConsole(vector<shared\_ptr<Person>>&, bool);

};

}

**[ Конец файла ui.h ]**

**[ Начало файла ui.cpp ]**

##include "ui.h"

#include "Student.h"

#include <iostream>

using namespace UI;

using namespace studt;

using namespace std;

template<typename T>

T getInput() {

T userInput = 0;

cout << "Ввод: ";

cin >> userInput;

if (cin.fail()) {

cout << "Некоректный ввод. Попробуйте еще раз" << endl;

while (cin.fail()) {

cin.clear();

cin.ignore(INT\_MAX, '\n');

cout << "Ввод: ";

cin >> userInput;

}

}

cin.ignore(INT\_MAX, '\n');

return userInput;

}

int ui::getMenuVar(int lower, int upper) {

bool inputCorrect = false;

int in = 0;

while (!inputCorrect) {

in = getInt();

if (in >= lower && in <= upper) {

inputCorrect = true;

}

else {

cout << "Нет такого пункта меню!" << endl;

}

}

return in;

}

int ui::getSize() {

bool inputCorrect = false;

int in = 0;

while (!inputCorrect) {

in = getInt();

if (in >= 1) {

inputCorrect = true;

}

else {

cout << "Введен отрицательный размер! Попробуйте еще раз" << endl;

}

}

return in;

}

int ui::getInt() {

return getInput<int>();

}

bool ui::getBool() {

return getInput<bool>();

}

double ui::getDouble() {

return getInput<double>();

}

int ui::getIntField() {

ui ui;

bool inputCorrect = false;

int in = 0;

while (!inputCorrect) {

in = ui.getInt();

if (in >= 1) {

inputCorrect = true;

}

else {

cout << "Введено неположительное число! Попробуйте еще раз" << endl;

}

}

return in;

}

double ui::getDoubleField() {

ui ui;

bool inputCorrect = false;

double in = 0;

while (!inputCorrect) {

in = ui.getDouble();

if (in >= 1) {

inputCorrect = true;

}

else {

cout << "Введено неположительное число! Попробуйте еще раз" << endl;

}

}

return in;

}

**[ Конец файла ui.cpp ]**

**[ Начало файла consoleInOut.cpp ]**

#include "ui.h"

#include "fileReader.h"

#include "Student.h"

#include "selection.h"

#include <iostream>

using namespace UI;

using namespace studt;

using namespace std;

using namespace reader;

void filling(shared\_ptr<Student> student) {

ui mUI{};

string surname{};

string adress{};

string group{};

double gpa{};

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Введите фамилию студента" << endl;

cout << "Ввод: ";

getline(cin, surname);

student->setSurnameCorrect(surname);

cout << "Введите адрес студента" << endl;

cout << "Ввод: ";

getline(cin, adress);

student->setAddressCorrect(adress);

cout << "Введите группу студента" << endl;

cout << "Ввод: ";

getline(cin, group);

student->setGroupCorrect(group);

cout << "Введите средний балл студента" << endl;

gpa = mUI.getDoubleField();

student->setGPACorrect(gpa);

}

void printInformOut() {

cout << "Вам предлагается ввести исходные данные в файл" << endl;

cout << "Используйте пункты меню для навигации в программе, нажмите :" << endl;

cout << "[1] - Записать" << endl;

cout << "[0] - Не записывать и продолжить выполнение программы" << endl;

}

void outInitialDataInFile(vector<shared\_ptr<Person>> people) {

ui ui{};

fileReader fr;

printInformOut();

bool variant = ui.getBool();

if (variant == 0) return;

fr.fillFile(people, false);

}

void consoleInOut::readConsole(vector<shared\_ptr<Person>>& people) {

ui mUI{};

int numberOfStudent = 0;

cout << "Вы решили вводить информацию о студентах через консоль, следуйте инструкциям программы" << endl;

cout << "Введите количество студентов: " << endl;

numberOfStudent = mUI.getSize();

cout << "-----..........-----..........-----..........-----" << endl;

for (int i = 0; i < numberOfStudent; i++) {

cout << "Вы вводите информацию о студенте №" << i+1 << endl;

shared\_ptr<Student> student = make\_shared<Student>();

filling(student);

people.push\_back(student);

cout << "-----..........-----..........-----..........-----" << endl;

}

outInitialDataInFile(people);

}

void printStudent(const shared\_ptr<Student>& student) {

cout << "-----..........-----..........-----..........-----" << endl;

cout << "Индефикатор студента: " << student->getToken() << endl;

cout << "Фамилия студента: " << student->getSurname() << endl;

cout << "Адрес студента: " << student->getAddress() << endl;

cout << "Группа студента: " << student->getGroup() << endl;

cout << "Средний балл студента: " << to\_string(student->getGPA()) << endl;

cout << "Средний балл группы: " << (student->getGroupAverage() == 0 ? "Не был определен" : to\_string(student->getGroupAverage())) << endl;

cout << "Статус студента: " << student->getStatus() << endl;

}

void consoleInOut::outConsole(vector<shared\_ptr<Person>>& people, bool getNonExcluded) {

if (people.size() != 0) {

if (getNonExcluded) {

select::selection sl{};

sl.expelledStudent(people);

}

for (auto& person : people) {

shared\_ptr<Student> student = dynamic\_pointer\_cast<Student>(person);

printStudent(student);

}

}

else {

cout << "Попробуйте ввести данные снова" << endl << endl;

throw runtime\_error("error");

}

}

**[ Конец файла consoleInOut.cpp ]**