МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №2**

по дисциплине: Объектно-ориентированное программирование

тема: «Модульное программирование. Интерфейсы»

Выполнил: ст. группы ПВ-233

Ситников Алексей Павлович

Проверил:

Белгород 2025 г.

**Цель работы**: Получение навыков модульной декомпозиции предметной области, создания модулей. Разработка интерфейсов.

Код программы:

main:

#include <iostream>

#include <windows.h>

#include "classes.h"

int main(void) {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

std::vector<Student> students;

std::vector<Group> Groups;

std::vector<teacher> teachers;

std::cout << "Команды:\n1) Добавить нового ученика\n2) Удалить ученика\n3) Поставить оценку\n4) Добавить учителя\n5) Удалить учителя\n6) Вывести оценки ученика\n7) Вывести состав группы\n8) Поменять группу\n9) Создать группу\n10) Вывести группы\n11) Вывести учителей\nВведите 0 чтобы выйти";

std::cout << "Введите команду\n";

short comand;

std::cin >> comand;

while (comand != 0) {

if (comand == 1) {

std::cout << "Введите имя фамилию и id группы\n";

std::string tempName;

std::string tempSurname;

int tempid;

std::cin >> tempName >> tempSurname >> tempid;

if (tempid >= id\_group) {

std::cout << "id не существует\n";

}

else{

Student newStudent(tempName, tempSurname, &Groups[tempid]);

students.push\_back(newStudent);

std::cout << "Ученик добавлен\n\n";

}

}

else if (comand == 2) {

std::cout << "Введите id ученика\n";

int tempid;

std::cin >> tempid;

if (tempid >= id\_student) {

std::cout << "id не существует\n";

}

else {

students[tempid].deleteStudent();

students.erase(students.begin() + tempid);

for (Student& temp : students) {

if (temp.getId() > tempid) {

temp.shiftStudents();

}

}

id\_student--;

std::cout << "Ученик удалён\n\n";

}

}

else if (comand == 3) {

std::cout << "Введите id ученика, название предмета, его оценку и id преподавателя\n";

int tempidSt;

std::string subject;

short score;

int tepidTc;

std::cin >> tempidSt >> subject >> score >> tepidTc;

if (tempidSt >= id\_student || tepidTc >= id\_teacher) {

std::cout << "id не существует\n";

}

else {

students[tempidSt].creatGrade(subject, score, tepidTc);

std::cout << "Оценка создана\n\n";

}

}

else if (comand == 4) {

std::cout << "Введите Имя фамилию учителя\n";

std::string name;

std::string surname;

std::cin >> name >> surname;

teacher newTeacher(name, surname);

teachers.push\_back(newTeacher);

std::cout << "Учитель добавлен\n\n";

}

else if (comand == 5) {

int tempid;

std::cout << "Введите id учителя\n";

std::cin >> tempid;

if (tempid >= id\_teacher) {

std::cout << "id не существует\n";

}

else {

teachers.erase(teachers.begin() + tempid);

for (teacher& temp : teachers) {

if (temp.id > tempid) {

temp.shiftTeachers();

}

}

id\_teacher--;

std::cout << "Учитель удалён\n\n";

}

}

else if (comand == 6) {

std::cout << "Введите id ученика\n";

int tempidSt;

std::cin >> tempidSt;

if (tempidSt >= id\_student) {

std::cout << "id не существует\n";

}

else {

students[tempidSt].printGrade();

}

}

else if (comand == 7) {

std::cout << "Введите id группы\n";

int tempid;

std::cin >> tempid;

if (tempid >= id\_group) {

std::cout << "id не существует\n";

}

else {

Groups[tempid].printStudents();

}

}

else if (comand == 8) {

std::cout << "Введите id ученика и новой группы\n";

int tempidSt;

int tempidGr;

std::cin >> tempidSt >> tempidGr;

int lastid = students[tempidSt].getGroup()->getId();

if (tempidGr >= id\_group || tempidSt >= id\_student) {

std::cout << "id не существует\n";

}

else {

students[tempidSt].changeGroup(&Groups[tempidGr]);

std::cout << "Ученик " + students[tempidSt].getFio() + " переведён из группы: " + Groups[lastid].getName() + " в группу " + Groups[tempidGr].getName() + "\n\n";

}

}

else if (comand == 9) {

std::cout << "Введите имя группы\n";

std::string name;

std::cin.ignore();

std::getline(std::cin, name);

Group newGroup(name);

Groups.push\_back(newGroup);

std::cout << "Группа создана\n\n";

}

else if (comand == 10) {

for (Group temp : Groups) {

temp.printGroup();

}

}

else if (comand == 11) {

std::cout << "Учителя:\n";

for (teacher temp : teachers) {

temp.printTeacher();

}

std::cout << std::endl;

}

std::cout << "\nВведите команду\n";

std::cin >> comand;

}

system("pause");

return 0;

}  
  
classes.h

#include <string>

#include <unordered\_map>

#include <vector>

#include <map>

extern int id\_group;

extern int id\_teacher;

extern int id\_student;

typedef struct grade {

private:

int id\_teacher;

std::string subject;

short score;

public:

grade(std::string subject\_, short score\_, int id\_teacher\_);

short getScore();

}grade;

typedef struct grades {

private:

std::map<std::string, std::vector<grade>> data;

public:

void creatGrade(std::string subject, short score, int id\_teacher);

void printGrade();

}grades;

typedef struct Group {

private:

std::vector<std::string> students;

int id;

std::string name;

public:

int getId();

std::string getName();

std::vector<std::string>\* getStudents();

Group(std::string name\_);

void printStudents();

void printGroup();

void deleteStudent(std::string str);

}Group;

typedef struct Student{

private:

std::string name;

std::string surname;

grades data;

int id;

Group\* group;

public:

std::string getFio();

int getId();

Group\* getGroup();

Student(std::string name\_, std::string surname\_, Group\* group\_);

void creatGrade(std::string subject, short score, int id\_teacher);

void printGrade();

void changeGroup(Group\* newGroup);

void shiftStudents();

void deleteStudent();

}student;

typedef struct teacher {

int id;

std::string name;

std::string surname;

teacher(std::string name\_, std::string surname\_);

void shiftTeachers();

void printTeacher();

}teacher;

classes.cpp  
#include "classes.h"

#include <iostream>

#include <algorithm>

int id\_group = 0;

int id\_teacher = 0;

int id\_student = 0;

grade::grade(std::string subject\_, short score\_, int id\_teacher\_) {

score = score\_;

id\_teacher = id\_teacher\_;

subject = subject\_;

}

short grade::getScore() {

return score;

}

void grades::creatGrade(std::string subject, short score, int id\_teacher) {

score = score < 2 ? 2 : score;

score = score > 5 ? 5 : score;

grade temp(subject, score, id\_teacher);

data[subject].push\_back(temp);

}

void grades::printGrade() {

for (auto& tempData : data) {

std::cout << tempData.first << ": ";

for (grade& tempGrades : tempData.second) {

std::cout << tempGrades.getScore() << ' ';

}

std::cout << std::endl;

}

}

int Group::getId() {

return id;

}

std::string Group::getName() {

return name;

}

std::vector<std::string>\* Group::getStudents() {

return &students;

}

Group::Group(std::string name\_) {

name = name\_;

id = id\_group++;

}

void Group::printStudents() {

std::sort(students.begin(), students.end());

std::cout << "Студенты группы " + name + ':' << std::endl;

for (std::string& student : students) {

std::cout << student << std::endl;

}

std::cout << std::endl;

}

void Group::printGroup() {

std::cout << "id: " << id << ", name: " << name << std::endl << std::endl;

}

void Group::deleteStudent(std::string str) {

students.erase(std::remove(students.begin(), students.end(), str), students.end());

}

std::string Student::getFio() {

return surname + ' ' + name;

}

int Student::getId() {

return id;

}

Group\* Student::getGroup() {

return group;

}

Student::Student(std::string name\_, std::string surname\_, Group\* group\_) {

name = name\_;

surname = surname\_;

group = group\_;

id = id\_student++;

std::string tempFio = surname\_ + ' ' + name\_ + ' ' + std::to\_string(id);

group->getStudents()->push\_back(tempFio);

}

void Student::creatGrade(std::string subject, short score, int id\_teacher) {

data.creatGrade(subject, score, id\_teacher);

}

void Student::printGrade() {

std::cout << "Оценки ученика: " + surname + " " + name << std::endl;

data.printGrade();

}

void Student::changeGroup(Group\* newGroup) {

std::string tempFio = surname + ' ' + name + ' ' + std::to\_string(id);

group->getStudents()->erase(std::remove(group->getStudents()->begin(), group->getStudents()->end(), tempFio), group->getStudents()->end());

group = newGroup;

group->getStudents()->push\_back(tempFio);

}

void Student::shiftStudents() {

std::string tempFio = surname + ' ' + name + ' ' + std::to\_string(id);

group->getStudents()->erase(std::remove(group->getStudents()->begin(), group->getStudents()->end(), tempFio), group->getStudents()->end());

id--;

tempFio = surname + ' ' + name + ' ' + std::to\_string(id);

group->getStudents()->push\_back(tempFio);

}

void Student::deleteStudent() {

std::string tempFio = surname + ' ' + name + ' ' + std::to\_string(id);

group->deleteStudent(tempFio);

}

teacher::teacher(std::string name\_, std::string surname\_) {

id = id\_teacher++;

name = name\_;

surname = surname\_;

std::string tempFio = name\_ + ' ' + surname\_;

}

void teacher::shiftTeachers() {

id--;

}

void teacher::printTeacher() {

std::cout << id << " " + surname + " " + name + '\n';

}

Вывод программы:  
Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.Изображение выглядит как текст, снимок экрана, меню

Контент, сгенерированный ИИ, может содержать ошибки.

**Вывод:** в ходе выполнения лабораторной работы были получены навыки модульной декомпозиции предметной области, создания модулей и разработки интерфейсов. Программа была успешно разделена на три модуля, что позволило сделать её более гибкой, расширяемой и поддерживаемой. Использование интерфейсов обеспечило абстракцию и упростило взаимодействие между модулями.