**МИНОБРНАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ   
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ТУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт прикладной математики и компьютерных наук

Кафедра информационной безопасности

**ЯЗЫКИ ПРОГРАММИРОВАНИЯ**

Отчет по выполнению лабораторной работы №15  
Вариант №14

Выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр.230711 Павлова Виктория Сергеевна

Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. ИБ Басалова Галина Валерьевна

Тула 2022

# ЛАБОРАТОРНАЯ РАБОТА №15. НАСЛЕДОВАНИЕ В С++

## ЦЕЛЬ РАБОТЫ

Изучить основные принципы создания и использования иерархии классов; разработать приложения по своим вариантам заданий.

## ЗАДАНИЕ НА РАБОТУ

**Задание 1.** Ознакомиться с теоретическим материалом, приведенным в пункте «Краткие теоретические положения» данных методических указаний, а также с конспектом лекций и рекомендуемой литературой по данной теме.

**Задание 2.** Разработать иерархию классов «Человек, студент, преподаватель, лаборант, институт». Кроме указанных в варианте задания свойств и методов, можно добавить свои, необходимые по смыслу предметной области, свойства и методы классов. Минимальные требования:

- не менее двух виртуальных функций,

- не менее трех свойств у классов-потомков;

- не менее трех методов;

- наличие конструкторов у всех классов.

Составить диаграмму классов и показать ее для согласования преподавателю. После этого реализовать составленную иерархию классов на языке С++ (в виде подключаемых .h или .cpp файла или в виде DLL).

**Задание 3.** Разработать основную программу (cpp-файл с функцией main), в которой используются созданные классы. В программе должны демонстрироваться возможности созданных классов.

## ХОД РАБОТЫ

Согласно заданию варианта №8, необходимо разработать программу с иерархией классов «Человек, студент, преподаватель, лаборант, институт». Диаграмма разработанных классов представлена на рисунке 1.



Рисунок 1 – Диаграмма классов

Описание разработанных классов и структур представлено в таблицах 1-5.

Таблица 1 – Описание разработанного базового класса Human

|  |  |  |
| --- | --- | --- |
| class Human | | |
| **Поля/свойства (элементы данных) класса** | | |
| Название и тип | Описание | |
| string fullName | ФИО | |
| string birthDate | Дата рождения в формате «ДД.ММ.ГГГГ» | |
| int age | Возраст | |
| char gender | Буква, обозначающая пол (‘М’ либо ‘Ж’) | |
| **Методы (функции-элементы) класса** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |
| Human() | string FIO, int ageNumber, char genderSymbol, string date | Конструктор класса |
| virtual void Print() | void | Виртуальная функция вывода сведений о личности человека |
| virtual void PrintIncomeInfo() | void | Виртуальная функция вывода сведений о доходах человека |
| ~Human() | void | Деструктор класса |

Таблица 2 – Описание разработанного наследуемого класса Teacher

|  |  |  |
| --- | --- | --- |
| class Teacher : public Human | | |
| **Поля/свойства (элементы данных) класса** | | |
| Название и тип | Описание | |
| string subject | Название преподаваемого предмета | |
| int workExperience | Трудовой стаж в годах | |
| int salary | Зарплата в рублях | |
| **Методы (функции-элементы) класса** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |
| Teacher() | Human info, string subjectName, int years, int salaryNumber | Конструктор класса |
| void SetSalary() | int salarySize | Установить зарплату размера salarySize |
| void Print() | void | Вывод сведений о личности человека |
| void PrintIncomeInfo() | void | Вывод сведений о доходах человека |
| ~Teacher() | void | Деструктор класса |

Таблица 3 – Описание разработанного наследуемого класса Student

|  |  |  |
| --- | --- | --- |
| class Student : public Human | | |
| **Поля/свойства (элементы данных) класса** | | |
| Название и тип | Описание | |
| int groupNumber | Номер группы | |
| int ID | Номер зачётной книжки | |
| int scholarship | Стипендия в рублях | |
| **Методы (функции-элементы) класса** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |

Таблица 3 – Описание разработанного наследуемого класса Student (продолжение)

|  |  |  |
| --- | --- | --- |
| Student() | Human info, int group, int IDnumber, int scholar | Конструктор класса |
| void SetScholarship() | int scholarshipSize | Установить стипендию размера scholarshipSize |
| void Print() | void | Вывод сведений о личности человека |
| void PrintIncomeInfo() | void | Вывод сведений о доходах человека |
| void CancelScholarship() | void | Отменить стипендию |
| ~Student() | void | Деструктор класса |

Таблица 4 – Описание разработанного наследуемого класса Assistant

|  |  |  |
| --- | --- | --- |
| class Assistant : public Teacher | | |
| **Поля/свойства (элементы данных) класса** | | |
| Название и тип | Описание | |
| Teacher teacher | Главный преподаватель | |
| **Методы (функции-элементы) класса** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |
| Assistant() | Teacher laborant, Teacher boss | Конструктор класса |
| void Print() | void | Вывод сведений о личности человека |
| void PrintIncomeInfo() | void | Вывод сведений о доходах человека |
| ~Assistant() | void | Деструктор класса |

Для работы с данными классами также создана структура Institute, описание которой представлено в таблице 5.

Таблица 5 – Описание разработанной структуры Institute

|  |  |  |
| --- | --- | --- |
| struct Institute | | |
| **Поля/свойства (элементы данных) структуры** | | |
| Название и тип | Описание | |
| string name | Название института | |
| Human director | Директор института | |
| int studentsNumber | Количество обучающихся | |
| int workersCount | Количество сотрудников | |
| map <int, Student> studentsList | Список студентов | |
| vector <Teacher> workersList | Список сотрудников | |
| map <string, Assistant> assistList | Список лаборантов и их преподавателей | |
| **Методы (функции-элементы) структуры** | | |
| Название и тип возвращаемого значения | Аргументы | Описание |
| Institute() | string nameData, Human directorData | Конструктор структуры |
| void Print() | void | Вывод информации об институте |
| int CountStudents() | void | Возвращает число обучающихся |
| int CountWorkers() | void | Возвращает число сотрудников |
| void MakeDirector() | Human newDirector | Назначить нового директора |
| Student Enroll | Human student, int group, int ID | Зачислить студента |
| void Expell() | int ID | Отчислить студента |
| void Dismiss() | string name | Уволить преподавателя |
| void Employ() | Human human, int salarySize, int workYears, string subjectName | Нанять преподавателя |
| void MakeAssistant() | Teacher boss, Assistant lab | Закрепить lab за преподаваталем boss |
| ~Institute() | void | Деструктор структуры |

## КОД ПРОГРАММЫ

## Содержимое заголовочного файла Human.h

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Human

{

public:

string fullName; //ФИО

string birthDate; //дата рождения 'ДД.ММ.ГГГГ'

int age; //возраст

char gender; //пол (русский символ 'М' либо 'Ж')

Human();

Human(string FIO, int ageNumber, char genderSymbol, string date);

virtual void Print() {}; //вывести сведения о человеке

virtual void PrintIncomeInfo() {}; //вывести сведения о доходах

~Human(){};

};

Human::Human()

{

fullName = "no\_name";

age = -1;

gender = 'n';

birthDate = "no\_date";

};

Human::Human(string FIO, int ageNumber, char genderSymbol, string date)

{

fullName = FIO;

age = ageNumber;

gender = genderSymbol;

birthDate = date;

};

ostream& operator << (ostream& output, const Human& person)

{

cout << "\t\tФИО: " << person.fullName << '\n';

cout << "\t\tПол: " << person.gender << '\n';

cout << "\t\tВозраст: " << person.age;

return output;

};

bool operator!=(const Human& personA, const Human& personB)

{

int k = 0;

if (personA.age != personB.age) k++;

if (personA.fullName != personB.fullName) k++;

if (personA.gender != personB.gender) k++;

if (personA.birthDate != personB.birthDate) k++;

return k == 4;

}

bool operator==(const Human& personA, const Human& personB)

{

int k = 0;

if (personA.age == personB.age) k++;

if (personA.fullName == personB.fullName) k++;

if (personA.gender == personB.gender) k++;

if (personA.birthDate == personB.birthDate) k++;

return k == 4;

}

istream& operator >> (istream& input, Human& human)

{

cout << "\t\tВведите ФИО: "; getline(cin >> ws, human.fullName);

cout << "\t\tВведите возраст: "; cin >> human.age;

cout << "\t\tВведите пол: "; cin >> human.gender;

cout << "\t\tВведите дату рождения (ДД.ММ.ГГГГ): "; getline(cin >> ws, human.birthDate);

return input;

}

## Содержимое заголовочного файла Teacher.h

#pragma once

#include "Human.h"

class Teacher : public Human

{

public:

string subject; //название преподаваемого предмета

int workExperience; //трудовой стаж

int salary; //зарплата в рублях

Teacher();

Teacher(Human info, string subjectName, int years, int salaryNumber);

friend ostream& operator << (ostream& output, const Teacher& teacher);

void SetSalary(int salarySize); //назначить зарплату

void Print(); //вывести информацию о работнике

void PrintIncomeInfo();

~Teacher() {};

};

Teacher::Teacher()

{

subject = "no\_subject";

workExperience = -1;

salary = -1;

}

Teacher::Teacher(Human info, string subjectName, int years, int salaryNumber) : Human(info)

{

subject = subjectName;

workExperience = years;

salary = salaryNumber;

}

ostream& operator << (ostream& output, const Teacher& teacher)

{

cout << "\t\tФИО: " << teacher.fullName << '\n';

cout << "\t\tПол: " << teacher.gender << '\n';

cout << "\t\tВозраст: " << teacher.age;

cout << "\t\tНазвание предмета: " << teacher.subject << '\n';

cout << "\t\tТрудовой стаж: " << teacher.workExperience << "\n";

cout << "\t\tЗарплата: " << teacher.salary << " рублей.\n\n";

return output;

}

void Teacher::Print()

{

cout << \*this;

}

void Teacher::PrintIncomeInfo()

{

cout << "\n\t\tСотрудник: " << fullName << "\n";

cout << "\t\tРазмер зарплаты: " << salary << "\n";

}

void Teacher::SetSalary(int salarySize)

{

salary = salarySize;

cout << "\n\t\tСотрудник: " << fullName << " теперь имеет размер зарплаты: " << salary << ".";

}

bool operator==(Teacher teacherA, Teacher teacherB)

{

int k = 0;

if (teacherA.fullName == teacherB.fullName) k++;

if (teacherA.age == teacherB.age) k++;

if (teacherA.gender == teacherB.gender) k++;

if (teacherA.birthDate == teacherB.birthDate) k++;

if (teacherA.salary == teacherB.salary) k++;

if (teacherA.subject == teacherB.subject) k++;

if (teacherA.workExperience == teacherB.workExperience) k++;

return k == 7;

}

## Содержимое заголовочного файла Student.h

#pragma once

#include "Human.h"

class Student : public Human

{

public:

int groupNumber; //номер учебной группы

int ID; //номер зачётной книжки

int scholarship; //размер стипендии

Student();

Student(Human info, int group, int IDnumber, int scholar);

void Print(); //вывести всю информацию о студенте

void PrintIncomeInfo(); //вывести сведения о стипендии

void SetScholarship(int scholarshipSize); //назначить стипендию

void CancelScholarship(); //снять со стипендии

friend ostream& operator << (ostream& output, const Student& student);

~Student() {};

};

Student::Student()

{

groupNumber = -1;

ID = -1;

scholarship = -1;

}

Student::Student(Human info, int group, int IDnumber, int scholar)

{

fullName = info.fullName;

age = info.age;

birthDate = info.birthDate;

gender = info.gender;

groupNumber = group;

ID = IDnumber;

scholarship = scholar;

}

void Student::SetScholarship(int scholarshipSize)

{

scholarship = scholarshipSize;

}

void Student::CancelScholarship()

{

scholarship = 0;

}

void Student::PrintIncomeInfo()

{

cout << "\t\tСтудент #" << ID << '\n';

cout << "\t\tСтипендия: " << scholarship << " рублей.\n";

}

void Student::Print()

{

cout << "\n" << \*this;

}

ostream& operator << (ostream& output, const Student& student)

{

cout << "\t\tФИО: " << student.fullName << '\n';

cout << "\t\tПол: " << student.gender << '\n';

cout << "\t\tВозраст: " << student.age;

cout << "\t\tНомер учебной группы: " << student.groupNumber << "\n";

cout << "\t\tНомер зачётной книжки: " << student.ID << '\n';

cout << "\t\tСтипендия: " << student.scholarship << " рублей.\n";

return output;

}

## Содержимое заголовочного файла LabAssistant.h

#pragma once

#include "Teacher.h"

class Assistant : public Teacher

{

public:

Teacher teacher; //преподаватель, за которым закреплён лаборант-ассистент

Assistant();

Assistant(Teacher laborant, Teacher boss)

{

fullName = laborant.fullName;

age = laborant.age;

gender = laborant.gender;

birthDate = laborant.birthDate;

subject = laborant.subject;

salary = laborant.salary;

workExperience = laborant.workExperience;

teacher = boss;

}

void Print();

void PrintIncomeInfo();

~Assistant(){};

};

Assistant::Assistant() : Teacher()

{

subject = "no\_subject";

workExperience = -1;

salary = -1;

}

ostream& operator << (ostream& output, const Assistant& laborant)

{

cout << "\t\tФИО: " << laborant.fullName << '\n';

cout << "\t\tПол: " << laborant.gender << '\n';

cout << "\t\tВозраст: " << laborant.age;

cout << "\t\tНазвание предмета: " << laborant.subject << '\n';

cout << "\t\tТрудовой стаж: " << laborant.workExperience << "\n";

cout << "\t\tЗарплата: " << laborant.salary << " рублей.\n\n";

cout << "\t\tГлавный преподаватель: " << laborant.teacher.fullName << '\n';

return output;

}

void Assistant::Print()

{

cout << \*this;

}

void Assistant::PrintIncomeInfo()

{

cout << "Лаборант: " << (\*this).fullName << "\n";

cout << "Размер зарплаты: " << (\*this).salary;

}

## Содержимое заголовочного файла Institute.h

#pragma once

#include <map>

#include "Human.h"

#include "Student.h"

#include "Teacher.h"

#include "LabAssistant.h"

struct Institute

{

string name; //название института

Human director; //директор института

int studentsCount; //количество обучающихся

int workersCount; //количество сотрудников

map<int, Student> studentsList; //список студентов

vector <Teacher> workersList; //список сотрудников

map <string, Assistant> assistList;

Institute();

Institute(string nameData, Human directorData);

void Print();

int CountStudents(); //вернуть количество обучающихся

int CountWorkers(); //вернуть количество сотрудников

void MakeDirector(Human newDirector); //назначить директора

Student Enroll(Human student, int group, int ID); //зачислить студента

void Expell(int ID); //отчислить студента

void Dismiss(string name); //уволить преподавателя

Teacher Employ(Human human, //нанять преподавателя

int salarySize, int workYears, string subjectName);

void MakeAssistant(Teacher boss, Assistant lab); //закрепить за новым преподавателем

~Institute(){};

};

Institute::Institute(string nameData, Human directorData)

{

name = nameData;

director = directorData;

studentsCount = 0;

workersCount = 0;

}

Institute::Institute()

{

name = "no\_name";

Human human;

director = human;

studentsCount = 0;

workersCount = 0;

}

void Institute::Print()

{

cout << "\n\t\t---> " << name << " <---\n";

cout << "\t\tДиректор: " << director.fullName << "\n";

cout << "\t\tКоличество обучающихся: " << studentsCount << "\n";

for (auto to: studentsList)

{

cout << "\t\tНомер зачётной книжки: " << to.first << ". " << to.second.fullName << " \t[гр. "

<< to.second.groupNumber << "]\n";

}

cout << "\n\t\tКоличество сотрудников: " << workersCount << "\n";

for (auto to : workersList)

{

cout << "\t\tПредмет: " << to.subject << ". " << to.fullName << '\n';

}

if (assistList.size() > 0)

{

cout << "\n\t\tСписок лаборантов:\n";

for (auto to : assistList)

{

cout << "\t\tПредмет: " << to.second.subject << ". \n\t\t\tГлавный преподаватель: "

<< to.first << ". \n\t\t\tАссистент: " << to.second.fullName << ".\n";

}

}

}

int Institute::CountStudents()

{

return studentsCount;

}

int Institute::CountWorkers()

{

return workersCount;

}

void Institute::MakeDirector(Human newDirector)

{

director = newDirector;

}

void Institute::MakeAssistant(Teacher boss, Assistant lab)

{

lab.teacher = boss;

assistList[boss.fullName] = lab;

};

Student Institute::Enroll(Human student, int ID, int group)

{

Student newStudent(student, group, ID, 0);

studentsList[ID] = newStudent;

newStudent.groupNumber = group;

studentsCount++;

cout << "\t\tСтудент " << newStudent.ID << " успешно зачислен в группу \t" << newStudent.groupNumber << ".\n";

return newStudent;

}

void Institute::Expell(int ID)

{

auto it = studentsList.find(ID);

if (it != studentsList.end())

{

cout << "\n\n\t\tСтудент " << (\*it).second.ID << " отчислен.\n";

studentsList.erase(it);

studentsCount--;

}

}

Student GetStudent(map<int, Student> studentsList, int ID)

{

Student error;

for (auto to : studentsList)

{

if (to.second.ID == ID)

{

auto it = find(studentsList.begin(), studentsList.end(), to);

return (\*it).second;

}

}

return error;

}

int GetTeacherIndex(vector <Teacher> workersList, string name, string subject)

{

for (size\_t i = 0; i < workersList.size(); i++)

{

if (workersList[i].fullName == name && workersList[i].subject == subject)

{

return i;

}

}

return -1;

}

void Institute::Dismiss(string name)

{

for (auto to : assistList) //если увольняемый является лаборантом

{

if (to.second.fullName == name)

{

auto it = find(assistList.begin(), assistList.end(), to);

cout << "\n\t\tЛаборант " << (\*it).second.fullName << " снят.\n";

assistList.erase(it);

break;

}

}

for (auto to : workersList)

{

if (to.fullName == name)

{

auto it = find(workersList.begin(), workersList.end(), to);

cout << "\n\t\tПреподаватель " << (\*it).fullName << " уволен.\n";

workersList.erase(it);

workersCount--;

break;

}

}

}

Teacher Institute::Employ(Human human, int salarySize, int workYears, string subjectName)

{

Teacher newTeacher;

newTeacher.fullName = human.fullName;

newTeacher.age = human.age;

newTeacher.gender = human.gender;

newTeacher.birthDate = human.birthDate;

newTeacher.workExperience = workYears;

newTeacher.salary = salarySize;

newTeacher.subject = subjectName;

workersCount++;

workersList.push\_back(newTeacher);

return newTeacher;

}

## Содержимое файла InheritanceClasses.cpp

#include <Windows.h>

#include <iostream>

#include <vector>

#include "Institute.h"

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Human DirectorIPMKN("Сычугов Алексей Алексеевич", 45, 'М', "04.06.1977");

Institute IPMKN("Институт прикладной математики и компьютерных наук", DirectorIPMKN);

Human Olga("Горбунова Ольга Юрьевна", 32, 'Ж', "17.01.1984");

Teacher ITTeacher = IPMKN.Employ(Olga, 20000, 8, "Физика");

Human Valentina("Плюшкина Валентина Сергеевна", 35, 'Ж', "12.01.1981");

Teacher MathTeacher = IPMKN.Employ(Valentina, 25000, 8, "Линейная алгебра");

Human Inna("Снежная Инна Олеговна", 28, 'Ж', "30.01.1994");

Assistant ITLaborant(IPMKN.Employ(Inna, 13000, 6, "Физика"), ITTeacher);

IPMKN.MakeAssistant(ITTeacher, ITLaborant);

Human Denis("Котиков Денис Сергеевич", 19, 'М', "22.05.2003");

Student studentA = IPMKN.Enroll(Denis, IPMKN.CountStudents() + 1, 230711);

Human Anna("Булочкина Анна Андреевна", 18, 'Ж', "02.02.2004");

Student studentB = IPMKN.Enroll(Anna, IPMKN.CountStudents() + 1, 230721);

Human German("Герман Риттер Оттович", 20, 'М', "15.12.2002");

Student studentC = IPMKN.Enroll(German, IPMKN.CountStudents() + 1, 230711);

IPMKN.Print();

if (ITLaborant.teacher == ITTeacher)

cout << "\t\tПреподаватель " << ITLaborant.fullName << " стал ассистентом преподавателя "

<< ITTeacher.fullName << ".\n";

IPMKN.Expell(studentC.ID);

IPMKN.Dismiss(Inna.fullName);

ITTeacher.PrintIncomeInfo();

ITTeacher.SetSalary(22000);

IPMKN.Print();

return 0;

}

## РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

Пример работы с разработанными классами приведён на рисунках 2-4. Имеется сущность «Институт», у которой есть директор, сотрудники и обучающиеся, выраженные сущностями «Преподаватель», а также «Ассистент-лаборант», наследуемый от него, и «Студент». Все они, в свою очередь, являются сущностями «Человек», как указанно в диаграмме классов на рисунке 1.

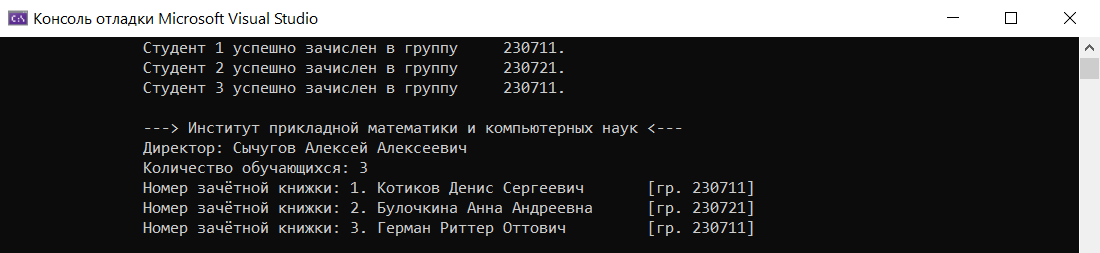


Рисунок 2 – Пример работы с сущностью «Студент» и «Институт»

На следующем рисунке показано закрепление лаборанта и добавление преподавателей в список сотрудников. Лаборант является Преподавателем.

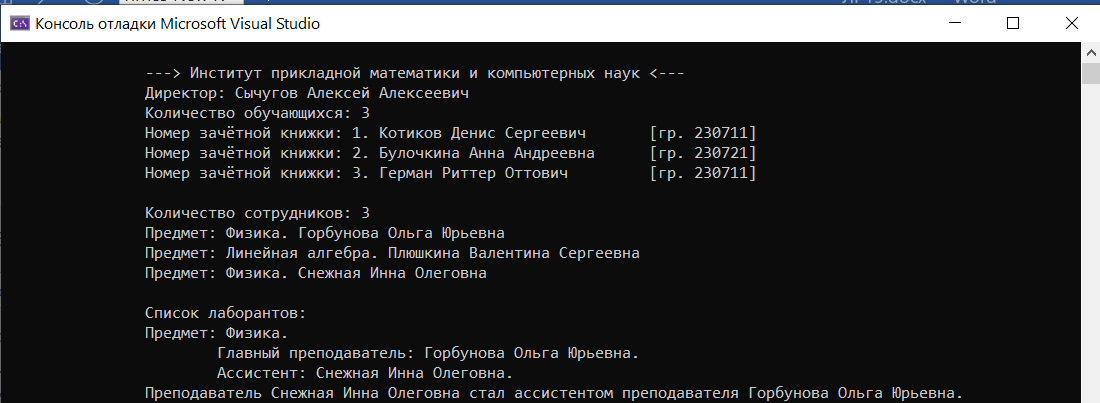


Рисунок 3 – Пример работы с сущностью «Преподаватель», «Институт» и «Лаборант»

На следующем рисунке приведён пример изменения зарплаты у сотрудника, а также отчисление студентов и увольнение преподавателя.

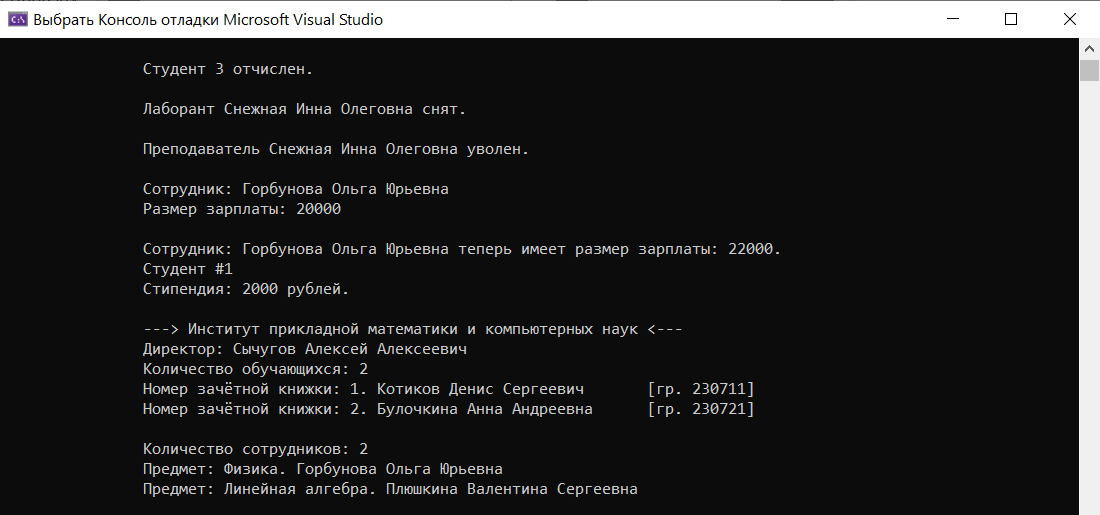


Рисунок 4 – Пример работы с сущностью «Преподаватель» и «Студент»

# ВЫВОД

В ходе выполнения лабораторной работы были изучены основные принципы наследования классов в языке С++. Для практического применения изученных понятия было разработано приложение по варианту.