Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 5

ПО ИНФОРМАТИКЕ ЗА II СЕМЕСТР

Вариант 15

Выполнил студент:

Сташков Арсений Дмитриевич

Группа РИС-20-1бз

Шифр 20-ЭТФ-644

Проверила:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

ПЕРМЬ 2022

# Лабораторная работа №5 "Наследование. Виртуальные функции. Полиморфизм"

**Вариант №15**

**Цель:** 1) Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.

2) Создание иерархии классов с использованием простого наследования и

абстрактного класса.

3) Изучение полиморфизма и виртуальных методов.

**Задача:** 1. Определить абстрактный класс.

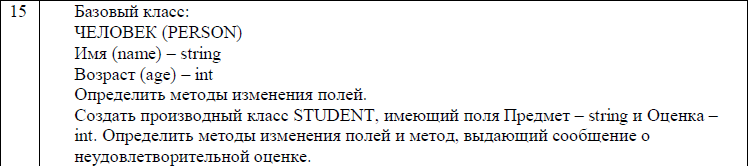
2. Определить иерархию классов, в основе которой будет находиться абстрактный класс (см. лабораторную работу №4).

3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.

4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.

5. В основной функции продемонстрировать перегруженные операции и

полиморфизм Вектора.



**Решение задания:**

Файл Vector.h:

#pragma once

#include "Obj.h"

#include <string>

#include <iostream>

using namespace std;

struct Vector {

Vector(void);

Vector(int);

~Vector(void);

void Add(Obj\*);

friend ostream& operator << (ostream& out, const Vector&);

private:

Obj\*\* beg;

int size;

int cur;

};

Файл Vector.cpp:

#include "Vector.h"

Vector::Vector(void) { beg = 0; size = 0; cur = 0; }

Vector::~Vector(void) { if (beg != 0) delete[] beg; beg = 0; }

Vector::Vector(int n) { beg = new Obj \* [n]; cur = 0; size = n; }

void Vector::Add(Obj\* p) {

if (cur < size) { beg[cur] = p; cur++; }

}

ostream& operator << (ostream& out, const Vector& v) {

if (v.size == 0) { out << "Empty\n"; }

Obj\*\* p = v.beg;

for (int i = 0; i < v.cur; i++)

{ (\*p)->Show(); p++; }

return out;

}

Файл Person.h:

#pragma once

#include <string>

#include <iostream>

#include "Obj.h"

using namespace std;

struct Person : public Obj {

Person(void);

virtual ~Person(void);

void Show();

Person(string, int);

Person(const Person&);

string Get\_name() { return name; }

int Get\_age() { return age; }

void Set\_name(string);

void Set\_age(int);

Person& operator = (const Person&);

friend istream& operator >> (istream& in, Person& c);

friend ostream& operator << (ostream& out, const Person& c);

protected:

string name;

int age;

};

Файл Person.cpp:

#include "Person.h"

Person::Person(void)

{ name = ""; age = 0; }

// Деструктор

Person::~Person(void) {}

// Конструктор с параметрами

Person::Person(string N, int A) { name = N; age = A; }

// Конструктор копирования

Person::Person(const Person& person) { name = person.name; age = person.age; }

// Модификаторы

void Person::Set\_name(string N) { name = N; }

void Person::Set\_age(int A) { age = A; }

// Перегрузка операции присваивания

Person& Person::operator = (const Person& c) {

if (&c == this) return \*this;

name = c.name;

age = c.age;

return \*this;

}

// Глобальная функция для ввода

istream& operator >> (istream & in, Person & c) {

cout << "\nИмя: ";

in >> c.name;

cout << "Возраст: ";

in >> c.age;

return in;

}

// Глобальная функция для вывода

ostream& operator << (ostream & out, const Person & c) {

out << "\n\nИмя: " << c.name;

out << "\nВозраст: " << c.age;

return out;

}

void Person::Show() {

cout << "\n\nИмя: " << name;

cout << "\nВозраст: " << age;

cout << "\n";

}

Файл Student.h:

#pragma once

#include "Person.h"

struct Student : public Person {

Student(void);

~Student(void);

void Show();

Student(string, int, string, int);

Student(const Student&);

string Get\_subject() { return subject; }

int Get\_grade() { return grade; }

void Set\_Subject(string);

void Set\_Grade(int);

Student& operator = (const Student&);

friend istream& operator >> (istream& in, Student& l);

friend ostream& operator << (ostream& out, const Student& l);

protected:

string subject;

int grade;

};

Файл Student.cpp:

#include "Student.h"

// Конструктор без параметров

Student::Student(void) : Person() { subject = ""; grade = 0; }

// Деструктор

Student::~Student(void) {}

// Конструктор с параметрами

Student::Student(string N, int A, string S, int G) :Person(N, A) { subject = S; grade = G; }

// Конструктор копирования

Student::Student(const Student& L) { name = L.name; age = L.age; }

// Модификаторы

void Student::Set\_Subject(string S) { subject = S; }

void Student::Set\_Grade(int G) { grade = G; }

// Операция присваивания

Student& Student::operator = (const Student& l) {

if (&l == this) return \*this;

name = l.name;

age = l.age;

subject = l.subject;

grade = l.grade;

return \*this;

}

// Операция ввода

istream& operator >> (istream & in, Student & l) {

cout << "\nИмя: ";

in >> l.name;

cout << "Возраст: ";

in >> l.age;

cout << "Предмет: ";

in >> l.subject;

cout << "Оценка: ";

in >> l.grade;

if (l.grade < 3 && l.grade > 0)

{ cout << "\nОценка неуд!: " << l.grade; }

else if (l.grade < 0 || l.grade > 5)

{ cout << "\nНекорректно введена оценка"; }

else

{ return in; }

}

// Операция вывода

ostream& operator << (ostream & out, const Student & l) {

out << "\nИмя: " << l.name;

out << "\nВозраст: " << l.age;

out << "\nПредмет: " << l.subject;

out << "\nОценка: " << l.grade;

out << "\n";

return out;

}

void Student::Show() {

cout << "\nИмя: " << name;

cout << "\nВозраст: " << age;

cout << "\nПредмет: " << subject;

cout << "\nОценка: " << grade;

cout << "\n";

}

Файл Obj.h:

#pragma once

struct Obj {

Obj() {}

~Obj() {}

// Чисто виртуальная функция

virtual void Show() = 0;

};

Файл Полиморфизм.cpp

#include <iostream>

#include <string>

#include <Windows.h>

#include "Person.h"

#include "Student.h"

#include "Obj.h"

#include "Vector.h"

using namespace std;

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Vector v(5);

Person a;

cout << "Заполните данные Person:\n";

cin >> a;

Student b;

cout << "\nЗаполните данные Student:\n";

cin >> b;

Obj\* p = &a;

v.Add(p);

p = &b;

v.Add(p);

cout << v;

}

**Результат:**

Заполните данные Person:

Имя: Alexander

Возраст: 23

Заполните данные Student:

Имя: Michael

Возраст: 21

Предмет: maths

Оценка: 4

Имя: Alexander

Возраст: 23

Имя: Michael

Возраст: 21

Предмет: maths

Оценка: 4

