Министерство образования РФ

Пермский национальный исследовательский политехнический университет

Кафедра ИТАС

ЛАБОРАТОРНАЯ РАБОТА № 4

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 №4 "Простое наследование. Принцип подстановки"

**Вариант №15**

**Цель:** 1) Создание консольного приложения, состоящего из нескольких файлов в системе

программирования Visual Studio.

2) Создание иерархии классов с использованием простого наследования.

3) Изучение принципа подстановки.

**Задача:** 1. Определить пользовательский класс.

2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.

3. Определить в классе деструктор.

4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).

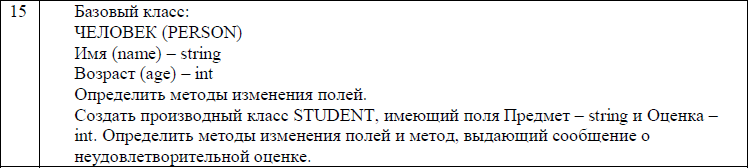
5. Перегрузить операцию присваивания.

6. Перегрузить операции ввода и вывода объектов с помощью потоков.

7. Определить производный класс.

8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.



**Решение задания:**

Файл Person.h:

#pragma once

#include <string>

#include <iostream>

using namespace std;

struct Person {

Person();

virtual ~Person();

Person(string, int);

// Конструктор копирования

Person(const Person&);

// Селекторы

string Get\_name() { return name; }

int Get\_age() { return age; }

// Модификаторы

void Set\_name(string);

void Set\_age(int);

// Перегрузка операции присваивания

Person& operator=(const Person&);

// Глобальные операторы-функции ввода-вывода

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

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

protected:

string name;

int age;

};

Файл Person.cpp:

#include <iostream>

#include "Person.h"

// Конструктор по умолчанию

Person::Person() { name = ""; age = 0; }

// 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; }

// Деструктор

Person::~Person() {}

// Модификаторы

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

void Person::Set\_name(string a) { name = 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 << "Введите имя и возраст для Person\n";

cout << "Имя: ";

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;

}

Файл Student.h:

#include "Person.h"

struct Student : public Person {

Student();

~Student();

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);

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

protected:

string subject;

int grade;

};

Файл Student.cpp:

#include <iostream>

#include "Student.h"

// Конструктор по умолчанию

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

// 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; }

// Деструктор

Student::~Student(void) {}

// 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 << "\n\nВведите имя, возраст, предмет и оценку для Student";

cout << "\nИмя: ";

in >> l.name;

cout << "\nВозраст: ";

in >> l.age;

cout << "\nПредмет: ";

in >> l.subject;

cout << "\nОценка: ";

in >> l.grade;

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

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

else if (l.grade < 2 || l.grade > 5)

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

else

{ cout << "\nОценка: " << l.grade; }

return in;

}

// Операция вывода

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

out << "\nИмя: " << l.name;

out << "\nВозраст: " << l.age;

out << "\nПредмет: " << l.subject;

out << "\n";

return out;

}

Файл Принцип подстановки.cpp

#include <iostream>

#include <Windows.h>

#include "Person.h"

#include "Student.h"

using namespace std;

// Функция, принимающая объект базового класса как параметр

void f1(Person& c) { c.Set\_name("Alexander"); c.Set\_age(20); cout << c; }

// Функция, возвращающая объект базового класса как результат

Person f2() {

Student l("Alexander", 21, "Math", 4);

return l;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Person a;

cin >> a;

cout << a;

Person b("Иван", 25);

cout << "\n\nКонструктор с параметрами:";

cout << b;

Student c;

cin >> c;

cout << c;

f1(c);

a = f2();

return 0;

}

**Результат:**

Введите имя и возраст для Person

Имя: Alex

Возраст: 20

Имя: Alex

Возраст: 20

Конструктор с параметрами:

Имя: Иван

Возраст: 25

Введите имя, возраст, предмет и оценку для Student

Имя: John

Возраст: 21

Предмет: maths

Оценка: 5

Оценка: 5

Имя: John

Возраст: 21

Предмет: maths

