МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

Факультет: Электротехнический

Кафедра: «Информационные технологии и автоматизированные системы» (ИТАС)

Направление: Разработка информационных систем (РИС)

**О Т Ч Ё Т**   
**о лабораторной работе №4**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Простое наследование. Принцип подстановки.»

Выполнил:

Студент группы РИС-24-3б

Суяргулов А.О.

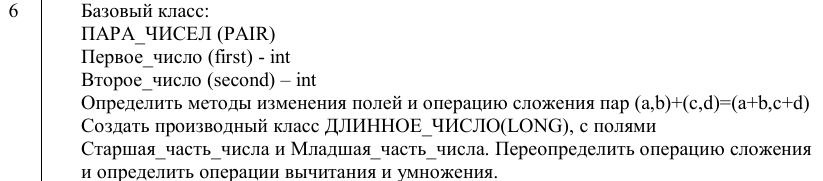
Проверил:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2025

ВВЕДЕНИЕ



ОСНОВНАЯ ЧАСТЬ

**Блок-схема:**

**Код:**

**Main:**

#include "PairLong.h"

#include <iostream>

#include <clocale>

using namespace std;

int main() {

// Для правильного отображения русского языка в консоли

setlocale(LC\_ALL, "Russian");

// Создаем пару чисел

Pair p1(1, 2);

// Выводим пару чисел на экран

cout << "Пара чисел: ";

p1.print();

cout << endl;

// Создаем длинное число

Long l1(3, 4, 5, 6);

// Выводим длинное число на экран

cout << "Длинное число: ";

l1.print();

cout << endl;

return 0;

}

**PairLong.h:**

#ifndef PAIRLONG\_H

#define PAIRLONG\_H

#include <iostream>

using namespace std;

// Базовый класс: Пара чисел

class Pair {

public:

// Поля класса

int first; // Первое число

int second; // Второе число

// Конструктор по умолчанию

Pair() {

first = 0;

second = 0;

}

// Конструктор с параметрами

Pair(int firstNum, int secondNum) {

first = firstNum;

second = secondNum;

}

// Метод для вывода пары чисел на экран

void print() {

cout << "(" << first << ", " << second << ")";

}

};

// Производный класс: Длинное число

class Long : public Pair {

public:

// Поля класса

int high; // Старшая часть числа

int low; // Младшая часть числа

// Конструктор с параметрами

Long(int firstNum, int secondNum, int highNum, int lowNum) : Pair(firstNum, secondNum) {

high = highNum;

low = lowNum;

}

// Метод для вывода длинного числа на экран

void print() {

cout << "Пара: ";

Pair::print(); // Вызываем метод print() базового класса

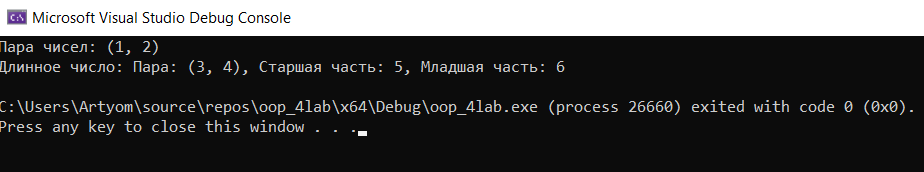
cout << ", Старшая часть: " << high << ", Младшая часть: " << low;

}

};

#endif // PAIRLONG\_H

**Результат работы кода:**

****

**Контрольные вопросы:**

1. Для чего используется механизм наследования?

Наследование позволяет создавать новые классы (производные классы) на основе существующих (базовых классов), переиспользуя и расширяя их функциональность. Это способствует повторному использованию кода, организации иерархии классов и реализации полиморфизма.

2. Каким образом наследуются компоненты класса, описанные со спецификатором public?

public компоненты базового класса наследуются как public в производном классе.

3. Каким образом наследуются компоненты класса, описанные со спецификатором private?

private компоненты базового класса не наследуются производным классом. Они доступны только внутри самого базового класса.

4. Каким образом наследуются компоненты класса, описанные со спецификатором protected?

protected компоненты базового класса наследуются как protected в производном классе. Они доступны внутри самого базового класса, производных классов и классов, находящихся в том же пакете (если речь идет о Java).

5. Каким образом описывается производный класс?

Используется ключевое слово class, имя производного класса, двоеточие, спецификатор доступа (например, public, protected или private) и имя базового класса:

c++

class DerivedClass : public BaseClass {

//

};

6. Наследуются ли конструкторы?

Нет, конструкторы не наследуются. Производный класс должен иметь свои собственные конструкторы. Однако производный класс может вызывать конструктор базового класса в своей инициализации.

7. Наследуются ли деструкторы?

Да, деструкторы наследуются. Важно, чтобы деструктор базового класса был объявлен как virtual, если есть вероятность полиморфного использования класса.

8. В каком порядке конструируются объекты производных классов?

1. Конструктор базового класса.

2. Члены-объекты производного класса (если есть).

3. Конструктор производного класса.

9. В каком порядке уничтожаются объекты производных классов?

1. Деструктор производного класса.

2. Деструкторы членов-объектов производного класса (в обратном порядке их объявления).

3. Деструктор базового класса.

10. Что представляют собой виртуальные функции и механизм позднего связывания?

Виртуальные функции — это функции, объявленные с ключевым словом virtual в базовом классе. Они позволяют переопределять поведение функции в производных классах. Позднее связывание (динамическое связывание) — это механизм, при котором вызов виртуальной функции определяется во время выполнения программы, основываясь на фактическом типе объекта, на который указывает указатель или ссылка.

11. Могут ли быть виртуальными конструкторы? Деструкторы?

Конструкторы не могут быть виртуальными. Деструкторы могут и должны быть виртуальными в базовых классах, если есть вероятность полиморфного использования класса, чтобы гарантировать правильное удаление объектов производных классов через указатель на базовый класс.

12. Наследуется ли спецификатор virtual?

Да, спецификатор virtual наследуется. Если функция объявлена как virtual в базовом классе и переопределена в производном классе, она остается виртуальной.

13. Какое отношение устанавливает между классами открытое наследование?

Открытое наследование (наследование с public) устанавливает отношение "is-a" (является). Производный класс является подтипом базового класса.

14. Какое отношение устанавливает между классами закрытое наследование?

Закрытое наследование (наследование с private) устанавливает отношение "implemented-in-terms-of" (реализован на основе). Производный класс использует функциональность базового класса, но не является его подтипом.

16. Имеется иерархия классов:

class Student

{

int age;

public:

string name;

};

class Employee : public Student

{

protected:

string post;

};

class Teacher : public Employee

{

protected: int stage;

};

Teacher x;

**Ссылка на GitHub:**

<https://github.com/Drazda1488/oop_4lab>