Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Кафедра **«**Информационные технологии и автоматизированные системы**»**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

Дисциплина: «Основы алгоритмизации и программирования»

Семестр 2

Тема: Простое наследование. Принцип подстановки.

Вариант 6.

Выполнил работу

Студент группы РИС-22-1Б

Поважный В. Е.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

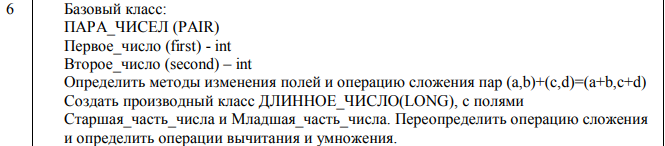
г. Пермь-2023

**Введение**

Для выполнения лабораторной работы требуется сделать отчет программы. Создание программ – отличный способ практики программирования.

**Постановка задачи**

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.



**Код программы**

**ооп 4.cpp**

#include <iostream>

#include "Header.h"

#include "Header1.h"

using namespace std;

Pair::Pair(void)

{

first = 0;

second = 0;

}

// деструктор

Pair::~Pair(void)

{

}

// конструктор с параметрами

Pair::Pair(int C, int P)

{

first = C;

second = P;

}

// конструктор копирования

Pair::Pair(const Pair& Pair)

{

first = Pair.first;

second = Pair.second;

}

// модификаторы

void Pair::Set\_first(int C)

{

first = C;

}

void Pair::Set\_second(int P)

{

second = P;

}

// перегрузка операции присваивания

Pair& Pair::operator=(const Pair& c)

{

if (&c == this)

return \*this;

second = c.second;

first = c.first;

return \*this;

}

// глобальная функция для ввода

istream& operator>>(istream& in, Pair& c)

{

cout << "\nsecond:";

in >> c.second;

cout << "\nfirst:";

in >> c.first;

return in;

}

// глобальная функция для вывода

ostream& operator<<(ostream& out, const Pair& c)

{

out << "\nfirst : " << c.first;

out << "\nsecond : " << c.second;

out << "\n";

return out;

}

\_Long::\_Long(void) : Pair()

{

gruz = 0;

}

// дестрктор

\_Long::~\_Long(void)

{

}

// конструктор с параметрами

\_Long::\_Long(int C, int P, double G) : Pair(C, P)

{

gruz = G;

}

// конструктор копирования

\_Long::\_Long(const \_Long& L)

{

first = L.first;

second = L.second;

gruz = L.gruz;

}

// модификатор

void \_Long::Set\_Gruz(double G)

{

gruz = G;

}

void \_Long::Long(int K, int L)

{

gruz = K + L;

}

// оперция присваивания

\_Long& \_Long::operator=(const \_Long& l)

{

if (&l == this)

return \*this;

second = l.second;

first = l.first;

gruz = l.gruz;

return \*this;

}

// операция ввода

istream& operator>>(istream& in, \_Long& l)

{

cout << "\nsecond:";

in >> l.second;

cout << "\nfirst:";

in >> l.first;

return in;

}

// операция вывода

ostream& operator<<(ostream& out, const \_Long& l)

{

out << "\nfirst : " << l.first;

out << "\nsecond : " << l.second;

out << "\nsumm : " << l.gruz;

out << "\n";

return out;

}

void f1(\_Long& c)

{

int first = c.first;

int second = c.second;

c.Long(first, second);

}

int main()

{

Pair a;

cin >> a;

cout << a;

Pair b(4, 115);

cout << b;

a = b;

cout << a;

\_Long c;

cin >> c;

f1(c);

cout << c;

system("pause");

return 0;

}

**Header.h**

**#include** <string>

#include <iostream>

using namespace std;

class Pair

{

public:

Pair(void);

public:

virtual ~Pair(void);

Pair(int, int);

Pair(const Pair&);

int Get\_first() { return first; }

int Get\_second() { return second; }

void Set\_first(int);

void Set\_second(int);

Pair& operator=(const Pair&);

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

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

int first;

int second;

};

**Header1.h**

**#include** "Header.h"

class \_Long : public Pair

{

public:

\_Long(void);

public:

~\_Long(void);

\_Long(int, int, double);

\_Long(const \_Long&);

int Get\_gruz() { return gruz; }

void Set\_Gruz(double);

void Long(int, int);

\_Long& operator=(const \_Long&);

friend istream& operator>>(istream& in, \_Long& l);

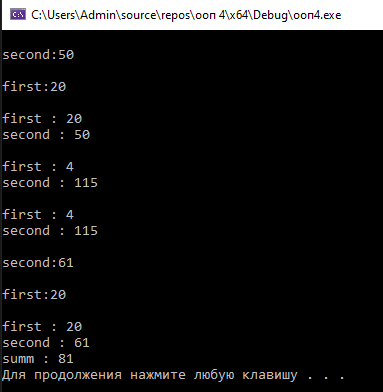
friend ostream& operator<<(ostream& out, const \_Long& l);

protected:

int gruz;

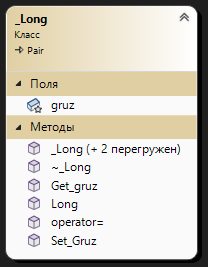
};

**Вывод программы**

****

***Рисунок 1 – вывод программы***

**UML диаграмма**

****

***Рисунок 2 – UML диаграмма***

**Вывод**

Программа выполняет свою задачу.

**Заключение**

Для решения задачи потребовались знания языка программирования, а конкретнее C++. Программа выполняет те условия, что были указаны в постановке задачи и работает без проблем.