Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.3**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: Классы и объекты. Перегрузка операций

Вариант 14

Выполнил:

студент группы РИС-20-2б

Вичугов Алексей Дмитриевич

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов.
2. Использование перегруженных операций в классах.

**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 (селекторы и модификаторы).
5. Перегрузить операцию присваивания.
6. Перегрузить операции ввода и вывода объектов с помощью потоков.
7. Задание варианта: Создать класс Pair с двумя полями типа int и double. Числа должны быть разделены двоеточием. Перегрузить операции сравнения. Реализовать вычитание константы из пары.
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

**Анализ задачи**

1. Определение класса:

class Para{

int a;

float b;

int \*u;

public:

Para();

Para(int c, float d);

Para(Para &p);

friend void Init(Para& p, int c, float d);

void Show();

int RetA();

float RetB();

Para& operator =(Para &p);

friend istream& operator >>(istream &ist, Para &p);

friend ostream& operator <<(ostream &os, Para &p);

bool operator !=(Para &p);

bool operator ==(Para &p);

Para& operator -(int value);

Para& operator -(float value);

~Para();

};

1. Определение компонентных функций:

Para::Para(){

a=0;

b=0.0F;

u=NULL;

}

Para::Para(int c, float d){

a=c;

b=d;

u=NULL;

}

Para::Para(Para &p){

a=p.RetA();

b=p.RetB();

u=NULL;

}

Para::~Para(){

delete u;

}

int Para::RetA() {return a;}

float Para::RetB() {return b;}

void Para::Show(){cout << "Numbers: " << a << ":" << b << endl;}

Para& Para::operator =(Para &p){

a=p.RetA();

b=p.RetB();

return \*this;

}

bool Para::operator !=(Para &p){

if (a!=p.RetA()&&b!=p.RetB()) return 1;

else return 0;

}

bool Para::operator ==(Para &p){

if (a==p.RetA()&&b==p.RetB()) return 1;

else return 0;

}

Para& Para::operator -(int value) {a-=value;return \*this;}

Para& Para::operator -(float value) {b-=value;return \*this;}

1. Определение глобальных функций:

void Init(Para& p, int c, float d){

p.a=c;

p.b=d;

}

ostream& operator <<(ostream &os, Para &p){

os << "Operator \"<<\": " << endl << "Numbers: " << p.RetA() << ":" << p.RetB() << endl;

}

istream& operator >>(istream& ist, Para &p){

cout << "Operator \'>>\': " << endl;

cout << "First (int): "; ist >> p.a;

while (ist.fail()){

ist.clear();

ist.ignore(10,'\n');

cout << "Incorrect input. Repeat first (int): ";

ist >> p.a;

}

ist.clear();

ist.ignore(10,'\n');

cout << "Second (float): "; ist >> p.b;

while (ist.fail()){

ist.clear();

ist.ignore(10,'\n');

cout << "Incorrect input. Repeat second (float): ";

ist >> p.b;

}

ist.clear();

ist.ignore(10,'\n');

}

1. Определение функции main:

main(){

cout.precision(5);

Para first(10, 50.53);

cout << "First: " << endl;

first.Show();

Init(first, 18, 46.52);

first.Show();

cout << endl << "Second: " << endl;

Para second;

second.Show();

if (second==first) cout << "Second == first. " << endl;

if (second!=first) cout << "Second != first. " << endl;

second=first;

cout<<second;

if (second==first) cout << "Second == first. " << endl;

if (second!=first) cout << "Second != first. " << endl;

cout << endl << "Third: " << endl;

Para third;

cout<<third;

cin>>third;

third.Show();

cout << endl << "Substraction: " << endl;

third=third-5;

third.Show();

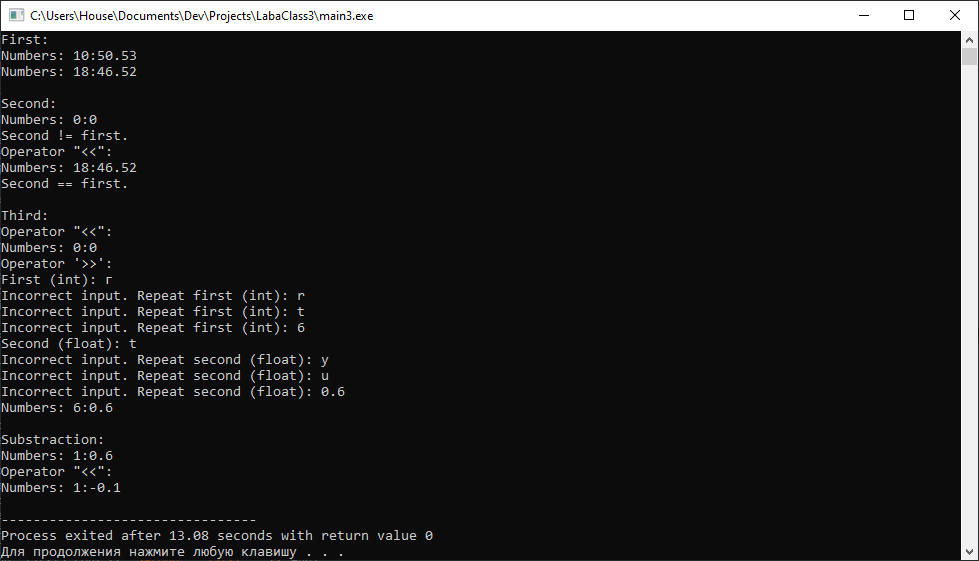
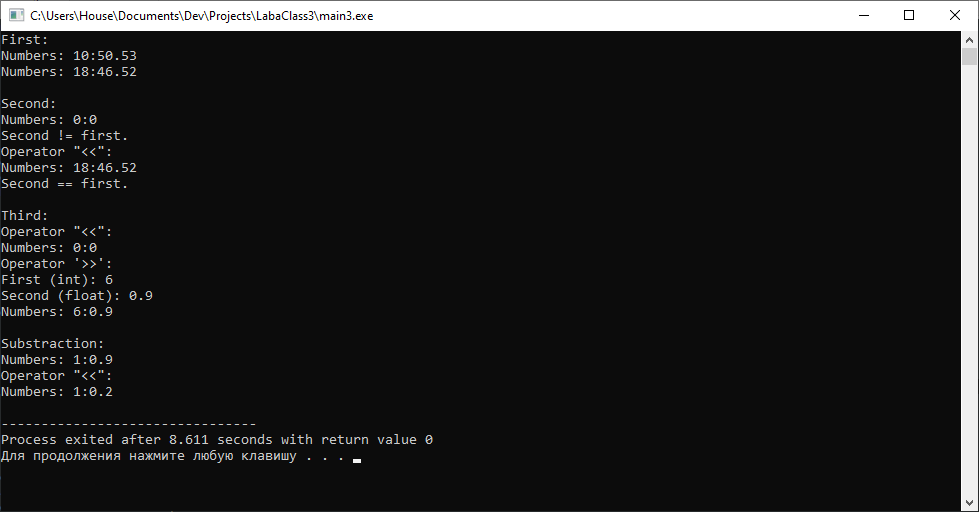
third=third-0.7F;

cout << third;

return 0;

}

**Результат работы программы**



**Ответы на вопросы**

1. Для получения возможность внешним функциям и классам иметь доступ к закрытым полям класса.
2. Дружественная функция объявляется в классе с помощью ключевого слово friend и определяется вне класса. Она имеет доступ ко всем полям класса.
3. С помощью ключевого слова operator.
4. Ноль.
5. Один.
6. Один.
7. Два.
8. Постфиксные операции должны иметь первый аргумент типа int.
9. С использованием ключевого слова operator и указанием параметра.
10. Объект, для которого перегружена операция.
11. Объявив их дружественными внутри класса и определив их вне класса с указанием параметров, один из которых является ссылкой на поток.
12. Будет выполнено тело этой функции-операции.
13. Будет выполнено тело этой функции-операции, определённой вне класса.
14. Он обратится к телу этой функции-операции, определённому в классе.
15. Он обратится к телу этой функции-операции, определённому вне класса.