Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Лабораторная работа №3»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

Пермь 2023

**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. Перегрузить операции указанные в варианте.
8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

**Вариант 8:**

Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа int для копеек. Дробная часть числа при выводе на экран должна быть отделена от целой части запятой. Реализовать:

- сложение денежных сумм,

- вычитание денежных сумм.

**Исходный код программы:**

**Файл “Money.h”:**

#include <iostream>

using namespace std;

class Money {

long int rub;

int kop;

public:

Money() {

rub = 0;

kop = 0;

}

Money(int rub, int kop) {

this->rub = rub;

this->kop = kop;

}

Money(const Money &other) {

rub = other.rub;

kop = other.kop;

}

~Money() {};

int get\_rub() {

return rub;

}

int get\_kop() {

return kop;

}

void set\_rub(int rub) {

this->rub = rub;

}

void set\_kop(int kop) {

this->kop = kop;

}

Money& operator=(const Money&);

Money operator+(const Money&);

Money operator-(const Money&);

friend istream& operator>>(istream&in, Money&t);

friend ostream& operator<<(ostream&out, const Money&t);

};

**Файл “Money.cpp”:**

#include "Money.h"

#include <iostream>

using namespace std;

Money&Money::operator=(const Money&t) {

if (&t == this)

return \*this;

rub = t.rub;

kop = t.kop;

return \*this;

}

Money Money::operator+(const Money&t) {

int temp1 = rub \* 100 + kop;

int temp2 = t.rub \* 100 + t.kop;

Money p;

p.rub = (temp1 + temp2) / 100;

p.kop = (temp1 + temp2) % 100;

return p;

}

Money Money::operator-(const Money&t) {

int temp1 = rub \* 100 + kop;

int temp2 = t.rub \* 100 + t.kop;

Money p;

p.rub = (temp1 - temp2) / 100;

p.kop = (temp1 - temp2) % 100;

return p;

}

istream&operator>>(istream&in, Money&t) {

cout << "Rubles: ";

in >> t.rub;

cout << "Kopecks: ";

in >> t.kop;

return in;

}

ostream&operator<<(ostream&out, const Money&t) {

return (out << t.rub << "." << t.kop << " rub.");

}

**Файл “Lab\_3.cpp”:**

#include "Money.cpp"

#include <iostream>

using namespace std;

int main() {

Money a;

Money b;

cin >> a;

cin >> b;

cout << "a = " << a << endl;

cout << "b = " << b << endl;

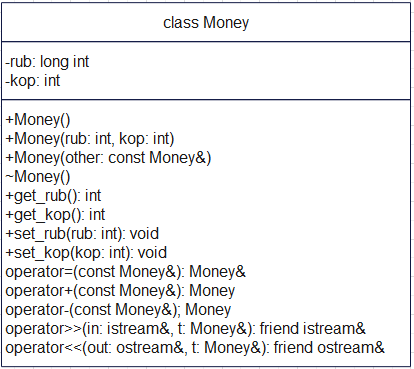
cout << a << " + " << b << " = " << a + b << endl;

cout << a << " - " << b << " = " << a - b << endl;

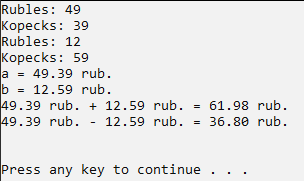
return 0;

}

**UML диаграмма:**



**Скриншот результата выполнения программы**



**Ответы на вопросы**

1. Для чего используются дружественные функции и классы?

**Ответ:** Если есть необходимость иметь доступ извне к скрытым полям класса, то есть расширить интерфейс класса.

class first\_class {

...

void show\_something(second\_class &item);

...

};

class second\_class {

...

friend void first\_class::show\_something(second\_class&);

...

};

1. Сформулировать правила описания и особенности дружественных функций.

**Ответ:**

Дружественная функция:

- Имеет доступ к скрытым полям класса.

- Не является компонентой функцией класса.

- Может быть глобальной функцией.

- Может быть компонентной функцией другого ранее определённого класса.

Правила описания:

- Описывается в теле класса ключевым словом **friend.**

- В параметры дружественной функции передаётся ссылка на объект класса.

1. Каким образом можно перегрузить унарные операции?

**Ответ:** Унарную операцию можно перегрузить:

- Как компонентную функцию класса.

- Как внешнюю функцию.

1. Сколько операндов должна иметь унарная функция-операция, определяемая внутри класса?

**Ответ:** Унарная фукнция-операция, определённая внутри класса не содержит операндов.

class my\_class {

int a, b;

public:

my\_class& operator++() {

++a;

return \*this;

}

};

1. Сколько операндов должна иметь унарная функция-операция, определяемая вне класса?

**Ответ:** Унарная функция-операция, определённая вне класса, содержит один параметр типа класса:

class my\_class {

int a, b;

public:

friend my\_class & operator++(my\_class&);

};

my\_class & operator ++(my\_class& p) {

++p.a;

return p;

}

1. Сколько операндов должна иметь бинарная функция-операция, определяемая внутри класса?

**Ответ:** Бинарная функция-операция, определяемая внутри класса, должна содержать операнд:

class my\_class {

int a, b;

public:

my\_class & operator +(int x) {

a += x;

reutrn \*this;

}

};

1. Сколько операндов должна иметь бинарная функция-операция, определяемая вне класса?

**Ответ:** Бинарная функция-операция, определяемая вне класса, должна содержать два параметра типа класса:

class my\_class {

int a, b;

public:

friend my\_class & operator +(my\_class& p, int x);

};

my\_class & operator +(my\_class &p, int x) {

p.a += x;

return p;

}

1. Чем отличается перегрузка префиксных и постфиксных унарных операций?

**Ответ:** Операции постфиксного инкремента и декремента должны иметь первый параметр типа int. Он используется для того, чтобы отличить их от префиксной формы.

1. Каким образом можно перегрузить операцию присваивания?

**Ответ:**

class my\_class {

int a, b;

public:

my\_class& operator =(const my\_class&);

};

my\_class& my\_class::operator = (const& my\_class p) {

if (&p = this)

return \*this;

a = p.a;

b = p.b;

return \*this;

}

1. Что должна возвращать операция присваивания?

**Ответ:** Операция-функция должна возвращать ссылку на объект, для которого она вызвана.

1. Каким образом можно перегрузить операции ввода-вывода?

**Ответ:**

class my\_class {

int a, b;

public:

friend istream& operator>>(istream&in, my\_class&p);

friend ostream& operator<<(ostream&out, my\_class&p);

};

istream&operator>>(istream&in, my\_class &p) {

cout << “A = “;

cin >> p.a;

cout << “B = “;

cin >> p.b;

return in;

};

ostream&operator<<(ostream&out, my\_class &p) {

return (out << p.a << “,” << p.b);

}

1. В программе описан класс

class Student {

...

Student& operator++();

...

и определён объект этого класса

Student s;

Выполняется операция

++s;

Каким образом, компилятор будет воспринимать вызов функции-операции?

**Ответ:** Компилятор будет воспринимать вызов функции-операции как префиксный инкремент.

1. В программе описан класс

class Student {

...

friend Student& operator ++(Student&);

...

};

и определён объект этого класса

Student s;

Каким образом, компилятор будет воспринимать вызов функций-операций?

**Ответ:** Компилятор будет воспринимать вызов функции-операции как префиксный инкремент.

1. В программе описан класс

class Student {

...

bool operator<(Student &P);

...

};

и определены объекты этого класса

Student a, b;

Выполняется операция

cout << a < b;

Каким образом, компилятор будет воспринимать вызов функции-операции?

**Ответ:** Компилятор будет воспринимать вызов функции-операции как бинарную операцию сравнения меньше.

1. В программе описан класс

class Student {

...

friend bool operator >(const Person&, Person&)

...

};

и определены объекты этого класса

Student a, b;

Выполняется операция

cout << a > b;

Каким образом, компилятор будет воспринимать вызов функции-операции?

**Ответ:** Компилятор будет воспринимать вызов функции-операции как бинарную операцию сравнения больше.