Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

направление подготовки: 09.03.01 - «Информатика и вычислительная техника»

**Отчет по лабораторной работе “Классы” №3**

**по дисциплине**

**«Теория алгоритмов и структуры данных»**

Выполнил студент гр. ИВТ-21-1б

Ахунов Руслан Булатович

Проверил:

Ст. Преподаватель кафедры ИТАС

Яруллин Денис Владимирович

(оценка) (подпись)

г. Пермь - 2022

**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 2:

Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. при выводе минуты отделяются от секунд двоеточием. Реализовать:

-   вычитание временных интервалов (учесть, что в минуте не может быть более 60 секунд)

-   сравнение временных интервалов (!=)

**Анализ задачи:**

class Time

{

private:

    int min;

    int sec;

public:

    Time()

    {

        min = 0;

        sec = 0;

    }

    Time(int m, int s)

    {

        min = m;

        sec = s;

    }

    Time(const Time& t)

    {

        min = t.min;

        sec = t.sec;

    }

    int getMin()

    {

        return min;

    }

    int getSec()

    {

        return sec;

    }

    void setMin(int m)

    {

        min = m;

    }

    void setSec(int s)

    {

        sec = s;

    }

    Time& operator=(const Time&t)

    {

        if (&t == this) {

            return \*this;

        }

        min = t.min;

        sec = t.sec;

        return \*this;

    }

    Time operator-(const Time& tmp);

    friend ostream& operator<<(ostream& out, const Time& tmp);

    friend void compar(const Time& tmp1, const Time& tmp2);

    ~Time()

    {

        cout << "Сработал деструктор..." << endl;

    }

};

Time Time::operator-(const Time& tmp)

{

    int tm1 = min \* 60 + sec;

    int tm2 = tmp.min \* 60 + tmp.sec;

    Time p;

    p.min = abs(tm1 - tm2) / 60;

    p.sec = abs(tm1 - tm2) % 60;

    return p;

}

ostream& operator<<(ostream& out, const Time& tmp)

{

    return (out << tmp.min << " : " << tmp.sec);

}

Класс Time, определены поля под минуты и секунды, min и sec, в спецификаторе доступа Private. Определены конструктор без параметров (Все значения полей равны 0), конструктор с параметрами (Значения присваиваются некоторые значения из других переменных), конструктор копирования.

«Гетеры» для минут и секунд, getMin и getSec. И так же «сетеры» для минут и секунд, setMin и setSec. Перегружен оператор присваивания и вычитания, так же перегружен оператор потокового вывода, и объявлена дружественная фукнция compar для сравнения временных промежутках.

int main()

{

    setlocale(LC\_ALL, "Russian");

    Time a(45, 13);

    Time b(5, 23);

    cout << "Первый промежуток - " << a << endl;

    cout << "Второй промежуток - " << b << endl;

    cout << "Разница между промежутками - " << a - b << endl << endl;

    cout << "/////" << endl << "Сравнение промежуктов:" << endl;

    compar(a, b);

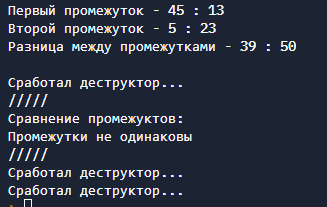
    cout << "/////" << endl;

    return 0;

}

В функции main, инициализируются две переменные типа Time, при этом используется конструктор с параметрами. Выводятся в консоль значения этих двух переменных, а так же разницу между ними.

**Результат работы программы**

****

При вычитании промежутков сработал деструктор, так как был закончен оператор -, и переменная p, была удалена из памяти.

Остальные два деструктора были запущены после окончания функции main.

**Код программы**

#include <iostream>

using namespace std;

class Time

{

private:

    int min;

    int sec;

public:

    Time()

    {

        min = 0;

        sec = 0;

    }

    Time(int m, int s)

    {

        min = m;

        sec = s;

    }

    Time(const Time& t)

    {

        min = t.min;

        sec = t.sec;

    }

    int getMin()

    {

        return min;

    }

    int getSec()

    {

        return sec;

    }

    void setMin(int m)

    {

        min = m;

    }

    void setSec(int s)

    {

        sec = s;

    }

    Time& operator=(const Time&t)

    {

        if (&t == this) {

            return \*this;

        }

        min = t.min;

        sec = t.sec;

        return \*this;

    }

    Time operator-(const Time& tmp);

    friend ostream& operator<<(ostream& out, const Time& tmp);

    friend void compar(const Time& tmp1, const Time& tmp2);

    ~Time()

    {

        cout << "Сработал деструктор..." << endl;

    }

};

Time Time::operator-(const Time& tmp)

{

    int tm1 = min \* 60 + sec;

    int tm2 = tmp.min \* 60 + tmp.sec;

    Time p;

    p.min = abs(tm1 - tm2) / 60;

    p.sec = abs(tm1 - tm2) % 60;

    return p;

}

ostream& operator<<(ostream& out, const Time& tmp)

{

    return (out << tmp.min << " : " << tmp.sec);

}

void compar(const Time& tmp1, const Time& tmp2)

{

    int tm1 = tmp1.min \* 60 + tmp1.sec;

    int tm2 = tmp2.min \* 60 + tmp2.sec;

    if (tm1 == tm2)

    {

        cout << "Промежутки одинаковы" << endl;

    }

    else

    {

        cout << "Промежутки не одинаковы" << endl;

    }

}

int main()

{

    setlocale(LC\_ALL, "Russian");

    Time a(45, 13);

    Time b(5, 23);

    cout << "Первый промежуток - " << a << endl;

    cout << "Второй промежуток - " << b << endl;

    cout << "Разница между промежутками - " << a - b << endl << endl;

    cout << "/////" << endl << "Сравнение промежуктов:" << endl;

    compar(a, b);

    cout << "/////" << endl;

    return 0;

}

**Вопросы**

1. Для чего используются дружественные функции и классы?

Дружественные функции применяются для доступа к скрытым полям класса и представляют собой альтернативу методам.

2. Сформулировать правила описания и особенности дружественных функций.

- Дружественная функция объявляется внутри класса*,* к элементам которого ей нужен доступ, с ключевым словом friend.

- Дружественная функция может быть обычной функцией или методом другого ранее определенного класса.

- Одна функция может быть дружественной сразу нескольким классам.

3. Каким образом можно перегрузить унарные операции?

* + - Как компонентную функцию класса
    - Как внешнюю (глобальную) функцию

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

Представлена с помощью нестатического метода без параметров, при этом операндом является вызвавший ее объект

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

Она должна иметь один параметр типа класса

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

Один(но два), но при этом вызвавший ее объект считается первым операндом.

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

Если функция определяется вне класса, она должна иметь два параметра типа класса.

8. Чем отличается перегрузка префиксных и постфиксных унарных операций?

Префиксные возвращают объект после его инкрементирования или декрементирования, когда постфиксные же возвращают объект до того как с ним будут выполнены данные действия.

9. Каким образом можно перегрузить операцию присваивания?

Пишется тип класса с ссылкой. Также должно быть написано ключевое слово “operator” и вслед за ним оператор присваивания. Операция-функция должна возвращать ссылку на объект, для которого она вызвана, и принимать в качестве параметра единственный аргумент — ссылку на присваиваемый объект.

10. Что должна возвращать операция присваивания?

Должна возвращать ссылку на объект, для которого она вызвана

11. Каким образом можно перегрузить операции ввода-вывода?

В классе нужно определить потоки ввода и вывода как дружественные функции, написать ключевое слово «operator», потом написать «<<» или «>>» в зависимости от того вывод это или ввод, в качестве параметров указываем ссылку на переменную ввода, вывода, и ссылку на объект класса

12. В программе описан класс class Student

{

…

Student& operator++();

….

};

и определен объект этого класса

Student s;

Выполняется операция

++s;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Сперва выполнится данный оператор, а после идут дальнейшие действия.

13. В программе описан класс class Student

{

…

friend Student& operator ++( Student&);

….

};

и определен объект этого класса Student s;

Выполняется операция

++s;

Каким образом, компилятор будет воспринимать вызов функции-операции?

Вернется значение до инкремента и с ним будут выполнятся действия, и только потом оно увеличится на 1.

14. 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?

Сперва идет действие с оператором «<», так он bool типа, то вернет либо true, либо false.

15. 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?

Сперва идет действие с оператором «>», так он bool типа, то вернет либо true, либо false.