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: 09.03.04 - «Программная инженерия»

**«Лабораторная работа №3**

**"Перегрузка операций"**

Выполнил студент гр. РИС-24-3б Носков Егор Михайлович

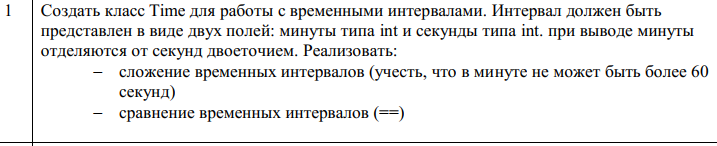
Проверил: Доцент кафедры ИТАС Ольга Андреевна Полякова

|  |  |
| --- | --- |
|  | |
| (оценка) (подпись) | |
|  |
| (дата) |

Г. Пермь, 2024

**Постановка задачи**

**Вариант 1:**

****

**Программа на языке C++**

Source.cpp:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Time {

int min;

int sec;

public:

Time() {

this->min = 0;

this->sec = 0;

};

Time(const int& min, const int& sec) {

this->min = min;

this->sec = sec;

};

Time operator+(const Time& A) const {

if (this->sec < 60 && A.sec < 60) {

if (this->sec + A.sec < 60) {

return Time(this->min + A.min, this->sec + A.sec);

}

else {

return Time(this->min + A.min+ (this->sec + A.sec) / 60, (this->sec+A.sec)%60);

}

}

else {

cerr << "В переменной не может содержаться больше 59 секунд";

return Time(0, 0);

};

};

bool operator==(const Time& A) const {

if (this->min == A.min && this->sec == A.sec) {

return true;

}

else return false;

};

friend std::ostream& operator<<(ostream& stream, const Time& A) {

stream << '(' << A.min << ':' << A.sec << ')';

return stream;

};

};

int main() {

setlocale(LC\_ALL, "RU");

Time One(4, 8);

Time Two(3, 59);

cout << "Первый временной интервал: " << One << endl;

cout << "Второй временной интервал: " << Two << endl;

cout<< "Сумма 1 и 2 временных интервалов" << One + Two<<endl;

if (One == Two) {

cout << "Интервалы 1 и 2 равны" << endl;

}

else cout<< "Интервалы 1 и 2 не равны" << endl;

Time Three(4, 20);

Time Four(4, 20);

cout<<"Третий временной интервал: " << Three << endl;

cout << "Четвертый временной интервал: " << Four << endl;

cout << "Сумма 1 и 2 временных интервалов" << Three + Four<<endl;

if (Three == Four) {

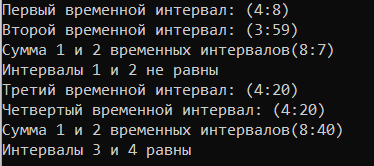
cout << "Интервалы 3 и 4 равны" << endl;

}

else cout << "Интервалы 3 и 4 не равны" << endl;

}

Вывод программы:



**Контрольные вопросы**

**1. Для чего используются дружественные функции и классы?**

Дружественные функции и классы предоставляют доступ к **private** и **protected** членам класса, даже если они не являются его методами. Используются для:

* Перегрузки операторов, работающих с приватными данными (например, operator<< для вывода).
* Реализации функций, которым нужен доступ к скрытым полям класса, но которые логически не принадлежат классу.

**2. Правила описания и особенности дружественных функций:**

* Объявляются внутри класса с ключевым словом friend.
* Не являются методами класса (не имеют this).
* Могут быть определены вне класса без использования оператора разрешения ::.
* Не наследуются в производных классах.

**3. Как перегрузить унарные операции?**

Унарные операции (например, ++, --) можно перегрузить:

* **Как метод класса** (1 аргумент — неявный this).
* **Как дружественную функцию** (1 явный аргумент).

Пример для ++:

class Student {

public:

Student& operator++(); *// Префиксный ++ (метод класса)*

friend Student& operator++(Student&); *// Дружественная функция*

};

**4. Количество операндов у унарной функции-операции внутри класса:**

1 операнд (неявный this). Например:

Student& operator++() { ... } *// this указывает на текущий объект.*

**5. Количество операндов у унарной функции-операции вне класса:**

1 операнд (явный аргумент). Например:

Student& operator++(Student& s) { ... } *// s — явный параметр.*

**6. Количество операндов у бинарной функции-операции внутри класса:**

1 операнд (второй — неявный this). Например:

bool operator<(const Student& other) { ... } *// this и other.*

**7. Количество операндов у бинарной функции-операции вне класса:**

2 операнда (оба явные). Например:

bool operator>(const Student& a, const Student& b) { ... }

**8. Отличие перегрузки префиксных и постфиксных унарных операций:**

* **Префиксная** (например, ++a):

Student& operator++(); *// Метод класса.*

* **Постфиксная** (например, a++):

Student operator++(int); *// Параметр int — фиктивный, для различия.*

**9. Как перегрузить операцию присваивания (**=**)?**

Перегружается только **как метод класса** (не может быть дружественной функцией):

Student& operator=(const Student& other) {

if (this != &other) { *// Проверка на самоприсваивание.*

*// Копирование данных из other в this.*

}

return \*this; *// Возврат ссылки на текущий объект.*

}

**10. Что должна возвращать операция присваивания?**

**Ссылку на текущий объект** (Student&), чтобы поддерживать цепочку присваиваний:

a = b = c; *// Корректно, если operator= возвращает ссылку.*

**11. Как перегрузить операции ввода-вывода (**>>**,**<<**)?**

Только **как дружественные функции**, так как левый операнд — поток (istream/ostream), а не объект класса:

friend ostream& operator<<(ostream& os, const Student& s) {

os << s.name; *// Пример вывода.*

return os;

}

friend istream& operator>>(istream& is, Student& s) {

is >> s.name; *// Пример ввода.*

return is;

}

**12. Вызов**++s**для**Student& operator++();**:**

Компилятор воспримет это как вызов метода класса:

s.operator++(); *// Префиксный инкремент.*

**13. Вызов**++s**для**friend Student& operator++(Student&);**:**

Компилятор вызовет дружественную функцию:

operator++(s); *// Передача s как аргумента.*

**14. Вызов**cout << a < b;**для**bool operator<(Student &P);**:**

Из-за приоритета операторов (<< выше <) код эквивалентен:

(cout << a) < b; *// Ошибка, так как cout << a возвращает ostream&.*

Правильная перегрузка для сравнения:

if (a < b) cout << "a < b";

**15. Вызов**cout << a > b;**для**friend bool operator>(const Student&, Student&);**:**

Аналогично, компилятор интерпретирует это как:

(cout << a) > b; *// Ошибка: нет operator> для ostream& и Student.*

Правильный вариант:

cpp

Copy

Download

cout << (a > b); *// Вывод результата сравнения.*

**Ссылка на репозиторий с отчетом Github**

****