МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Череповецкий государственный университет»

**Лабораторная работа № 3**

**«ПЕРЕГРУЗКА ОПЕРАТОРОВ»**

**Выполнил:**

студент гр. 1ИВТпб-01-21оп

Климов А.Г.  
**Проверил:** преподаватель

Пышницкий К.М.  
Отметка о зачете:

Череповец

2017 год

**Цель работы:** изучить понятие перегрузки, приобрести практические навыки перегрузки отдельных операторов.

**Вариант**

A 16. Осветительные приборы.

**Задания**

1. Модифицируйте абстрактный тип данных, реализованный по заданию раздела V, согласно варианту задания (см. раздел III УМП\_ООП\_1.doc), перегрузив все доступные операторы.

2. Проверьте работоспособность АТД на тестовом наборе данных.

**Ход работы**

**Текст программы:**

**ClassLight.h**

class light

{

public:// набор функций

light& operator=(const light &m);

friend light operator-(light a, light b);

light operator++();

//

light(); //конструктор по умолчанию

light(char \*nm1, char nm2, int i, float j); //конструктор с параметрами

light(const light &t); //конструктор с const, меняющий содержимое, к которому получает доступ (копирующий конструктор)

~light() { delete[]name1; } //деструктор

//

void input(char \*nm1, char nm2, int i, float j);

void print();

void prisv(light b);

bool sravn(light b);

void destr() { delete[]name1; } // уничтожение объектов

private:

char \*name1; // название осветительного прибора

char name2; // тип лампочки

int k; // размер цоколя

float d; // яркость света

};

**ClassLight.cpp**

#include "ClassLight.h"

#include <iostream>

#include <string>

void light::input(char \*nm1, char nm2, int i, float j)

{

name1 = new char[strlen(nm1) + 1];

strcpy(name1, nm1);

name2 = nm2;

k = i;

d = j;

}

void light::print()

{

std::cout << "Название: " << name1 << std::endl;

std::cout << "Цоколь: " << name2 << k << std::endl;

std::cout << "Яркость света: " << d << std::endl;

std::cout << std::endl;

}

void light::prisv(light b)

{

delete[]name1;

name1 = new char[strlen(b.name1) + 1];

strcpy(name1, b.name1);

name2 = b.name2;

k = b.k;

d = b.d;

}

bool light::sravn(light b)

{

return ((strcmp(name1, b.name1) == 0) && (name2 == b.name2) && (k == b.k) && (d == b.d)) ? true : false;

}

//конструкторы

light::light() {

name1 = new char[strlen("Unknown") + 1];

strcpy(name1, "Unknown");

name2 = 'E';

k = 0;

d = 0;

}

light::light(char \*nm1, char nm2, int i, float j) {

name1 = new char[strlen(nm1) + 1];

strcpy(name1, nm1);

name2 = nm2;

k = i;

d = j;

}

light::light(const light &t) {

name1 = new char[strlen(t.name1) + 1];

strcpy(name1, t.name1);

name2 = t.name2;

k = t.k;

d = t.d;

}

//перегрузка операторов

light light::operator++() {

k++;

d++;

return (\*this);

}

light operator-(light a, light b) { //должно быть без h

light h;

h.k = a.k - b.k;

h.d = a.d - b.d;

return(h);

}

light& light::operator=(const light &m){

k = m.k;

d = m.d;

delete[]name1;

name1 = new char[strlen(m.name1) + 1];

strcpy(name1, m.name1);

return(\*this);

}

**Main.cpp**

#include <iostream>

#include "ClassLight.h"

using namespace std;

void main()

{

setlocale(LC\_ALL, "Russian");

//

//лр 1

light x, y, z;

x.input("Светильник1", 'E', 40, 70.5);

y.input("Светильник2", 'E', 27, 90.5);

z.input("Светильник3", 'E', 14, 50.5);

x.print();

y.print();

z.print();

cout << endl;

y.prisv(z);

x.print();

y.print();

z.print();

//y.destr();

y.input("Светильник35", 'E', 14, 50.5);

cout << y.sravn(z) << endl;

/\*x.destr(); // освобождение памяти

y.destr();

z.destr();\*/

//

//лр 2

light t1, t2("Светильник4", 'E', 40, 22.3), t3(t2);

t1.print();

t2.print();

t3.print();

cout << "-------------------" << endl;

//лр 3

light t4("Светильник1", 'E', 40, 22.3),t5("Светильник2", 'E', 25, 10.8);

operator-(t4, t5).print();

t4.operator++();

t4.print();

t5 = t4;

t5.print();

system("pause");

}

**Результаты тестирования:**

*Входные данные 1(перегрузка операторов: =,++,-):*

light t4("Светильник1", 'E', 40, 22.3), t5("Светильник2", 'E', 25, 10.8);

operator-(t4, t5).print();

t4.operator++();

t4.print();

t5 = t4;

t5.print();

*Результат 1 Рис.1:*

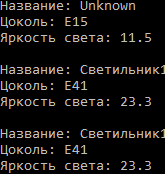


Рис. 1. Результат 1

*Входные данные 2 (перегрузка операторов: =,++,-):*

light t4("Светильник1", 'E', 40, 22.3), t5("Светильник2", 'E', 25, 10.8);

operator-(t4, t4).print();

t4.operator++();

t4.print();

t4 = t5;

t4.print();

*Результат 2 Рис.2:*

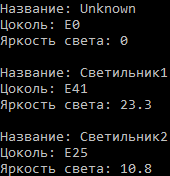


Рис. 2. Результат 2

**Ответы на контрольные вопросы:**

1. Назовите алгоритм соответствия для каждого аргумента.

Алгоритм соответствия для каждого параметра следующий:

1. Использовать точное соответствие, если оно найдено.

2. Проверить поддержку стандартных типов.

3. Проверить стандартные преобразования типов.

4. Проверить преобразования, определяемые пользователем.

5. Использовать соответствие для аргументов, если оно найдено.

2. С какой целью применяется перегрузка операторов?

Термин «перегрузка» относится к использованию одного и того же имени для множества значений функции. Перегрузка операторов позволяет абстрактному типу данных использовать синтаксис выражений С++, что существенно облегчает написание программ.

3. Для каких операторов нельзя использовать перегрузку?

Для перегрузки доступны все арифметические, логические операторы, операторы сравнения, равенство, присвоение, префиксные и постфиксные формы операторов приращения и декремента, а также оператор индексации «[]» и оператор обращения к функции «()». Однако нельзя перегружать троичный условный оператор выражения «?:», оператор разрешения области видимости «:: », оператор членства «.» и оператор селектора члена объекта «.\*».

4. В каком случае используется функция-член, имеющая пустой список аргументов?

Для перегрузки унарного оператора используется функция-член, имеющая пустой список аргументов, так как единственный аргумент оператора является неявным аргументом. Для бинарных операторов функция-член перегрузки оператора имеет в качестве первого аргумента неявно передаваемую переменную класса, а в качестве второго аргумента - единственный аргумент списка параметров. В дружественных и обычных функциях отсутствует неявный аргумент и они получают оба аргумента из списка параметров.

5. Что произойдет, если перегруженная операция [ ] будет иметь тип возвращаемого значения int, а не int&?

Когда мы передаем объект не по ссылке, то по сути создается новый объект (копия того, который передается в метод), что может нести за собой определенные издержки как по времени выполнения программы, так и по потребляемой ей памяти. — Применительно к большим объектам.

Передавая объект по ссылке, не происходит выделения памяти под сам объект (предположим, 128 байт) и операции копирования, память выделяется лишь под указатель на ячейку памяти, с которой мы работаем, а это около 4 — 8 байт. Таким образом, получается работа с объектом напрямую.

Но, если мы передаем объект по ссылке, то он становится изменяемым. То есть ничто не помешает нам при операции присваивания (v1 = v2) изменять не только значение v1, но еще и v2!

Можно создать функцию-оператор operator[]() таким образом, чтобы оператор [] можно было использовать как с левой, так и с правой стороны оператора присваивания. Для этого достаточно в качестве возвращаемой величины для operator[]() задать ссылку.

**Вывод:** изучил понятие перегрузки, приобрёл практические навыки перегрузки отдельных операторов. Для этого был разработан класс “Осветительные приборы”. Для созданного класса было проведено тестирование.