МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Череповецкий государственный университет»

**Лабораторная работа № 4**

**«ПРОГРАММИРОВАНИЕ ШАБЛОННЫХ КЛАССОВ И ФУНКЦИЙ»**

**Выполнил:**

студент гр. 1ИВТпб-01-21оп

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

Пышницкий К.М.  
Отметка о зачете:

Череповец

2017 год

**Цель работы:** изучить способы определения шаблонов, принцип и механизм создания шаблонных классов и шаблонов функций, получить практические навыки работы с шаблонными классами.

**Вариант**

A 16. Осветительные приборы.

**Задания**

1. Модифицируйте абстрактный тип данных, реализованный по заданию лабораторной работы 6, согласно варианту задания (см. раздел III УМП\_ООП\_1.doc), используя шаблоны определения класса и шаблоны определения функции.

2. Проверьте работоспособность АТД на тестовом наборе данных.

**Ход работы**

**Текст программы:**

**ClassLight.h**

template <class T> 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, T 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

T k; // размер цоколя

float d; // яркость света

};

**ClassLight.cpp**

#include "ClassLight.h"

#include <iostream>

#include <string>

template<typename T> void light<T>::input(char \*nm1, char nm2, int i, float j)

{

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

strcpy(name1, nm1);

name2 = nm2;

k = i;

d = j;

}

template<typename T> void light<T>::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;

}

template<typename T> void light<T>::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;

}

template<typename T> bool light<T>::sravn(light b)

{

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

}

//конструкторы

template<typename T> light<T>::light() {

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

strcpy(name1, "Unknown");

name2 = 'E';

k = 0;

d = 0;

}

template<typename T> light<T>::light(char \*nm1, char nm2, T i, float j) {

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

strcpy(name1, nm1);

name2 = nm2;

k = i;

d = j;

}

template<typename T> light<T>::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

template<typename T> light<T> light<T>::operator++() {

k++;

d++;

return (\*this);

}

template<typename T> light<T> operator-(light<T> a, light<T> b) {

light<T> h;

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

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

return(h);

}

template<typename T> light<T>& light<T>::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);

}

template class light<int>;

template class light<float>;

template class light<char>;

**Main.cpp**

#include <iostream>

#include "ClassLight.h"

using namespace std;

void main()

{

setlocale(LC\_ALL, "Russian");

light <int> t1("Светильник1", 'E', 25.5, 10.8);

t1.print();

light <float> t2("Светильник2", 'E', 25.5, 10.8);

t2.print();

light <char> t3("Светильник3", 'E', 71, 10.8);

t3.print();

system("pause");

}

**Результаты тестирования:**

*Входные данные 1 (шаблон класса light для k):*

light <int> t1("Светильник1", 'E', 25.5, 10.8);

light <float> t2("Светильник2", 'E', 25.5, 10.8);

light <char> t3("Светильник3", 'E', 71, 10.8);

*Результат 1 Рис.1:*

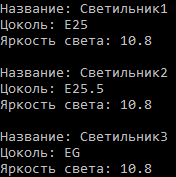


Рис. 1. Результат 1

*Входные данные 2:*

light <int> t1("Светильник1", 'E', 11.5, 10.8);

light <float> t2("Светильник2", 'E', 11.5, 10.8);

light <char> t3("Светильник3", 'E', 11.5, 10.8);

*Результат 2 Рис.2:*

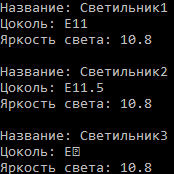


Рис. 2. Результат 2

*Входные данные 3:*

light <int> t1("Светильник1", 'E', 77, 10.8);

light <float> t2("Светильник2", 'E', 77, 10.8);

light <char> t3("Светильник3", 'E', 77, 10.8);

*Результат 3 Рис.3:*

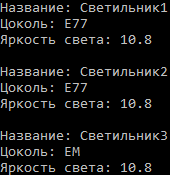


Рис. 3. Результат 3

**Ответы на контрольные вопросы:**

1. Для чего используется ключевое слово template?

Для реализации параметризованных типов.

2. В чем заключаются особенности параметров по умолчанию для шаблонов?

Параметры по умолчанию можно инстанцировать при объявлении аргументов, а можно опустить (используются значения по умолчанию). Параметрами шаблонов могут быть: параметры-типы, параметры обычных типов, параметры-шаблоны.

Для параметров любого типа можно указывать значения по умолчанию.

3. Для чего используются шаблоны функций?

Шаблоны определения класса и шаблоны определения функции позволяют многократно использовать код простым способом, безопасным по отношению к типу данных, который разрешает компилятору автоматически реализовать этот тип данных.

4. Назовите разновидности дружественных функций шаблонного класса.

Шаблонные классы могут содержать «друзей». Дружественные функции могут быть универсальными и могут не использовать спецификацию шаблона (для всех инстанцирований шаблонного класса имеется единственный экземпляр дружественной функции), а также могут быть специфичными и могут задействовать аргументы шаблона для каждого инстанцирования шаблонного класса.

Например:

template <class T>

class matrix {

public:

friend void foo\_bar(); //универсальна

friend vect<T> product(vect<T> v); //специфична

...

};

5. Дайте определение специализации шаблона.

Если шаблонный код является неудовлетворительным для конкретного типа аргумента, он может быть специализирован. Шаблонная функция может быть перегружена функцией, список аргументов и возвращаемый тип которой соответствуют объявлению шаблона. Такая перегружающая шаблон функция является специализацией шаблона. Если специализация соответствует вызову, то вызывается именно она, а не код, сгенерированный из шаблона. Например:

void maxelement<char\*>(char\*a[], char\* &max, int size);

// специализирована с использованием strcmp()

// для возвращения max string

Это – специализация объявленного ранее шаблона template <class T> maxelement().

Также возможна и специализация класса. Например:

class stack<foobar\_obj> {/\* специализация для foobar\_obj \*/};

**Вывод:** изучил способы определения шаблонов, принцип и механизм создания шаблонных классов и шаблонов функций, получил практические навыки работы с шаблонными классами. Для этого был разработан класс “Осветительные приборы”. Для созданного класса было проведено тестирование.