# Лабораторная работа № 5

**Тема работы:** реализация шаблонов классов.

**Цель работы:** научиться использовать шаблоны функций для создания группы однотипных функций, а также шаблоны классов для создания группы связанных типов классов.

**Теоретические сведения:**

Шаблон семейства классов определяет способ построения отдельных классов подобно тому, как класс определяет правила построения и формат отдельных объектов. Этот класс можно рассматривать как некоторое описание множества классов, отличающихся только типами их данных. В С++ используется ключевое слово template для обеспечения параметрического полиморфизма. Шаблоны определения класса и шаблоны определения функции позволяют многократно использовать код, корректно по отношению к различным типам.

Формат шаблона класса имеет вид

template <список параметров>

class объявление класса

Список параметров шаблона класса представляет собой идентификатор типа, подставляемого в объявление данного класса при его генерации. Идентификатору типа предшествует ключевое слово **class** или typename.

Определение шаблона может быть только глобальным.

Далее будет приведен пример определения шаблона класса вектора (одномерного массива). Какой бы тип ни имели элементы массива (целый, вещественный, с двойной точностью и т. д.), в этом классе должны быть определены одни и те же базовые операции, например, доступ к элементу по индексу и т. д. Если тип элементов вектора задавать как параметр шаблона класса, то система будет формировать вектор нужного типа (и соответствующий класс) при каждом определении конкретного объекта.

В программе шаблон семейства классов с общим именем vector используется для формирования двух классов с массивами целого и символьного типов. В соответствии с требованием синтаксиса имя параметризованного класса, определенное в шаблоне (в примере − vector), используется в программе только с последующим конкретным фактическим параметром (аргументом), заключенным в угловые скобки. Параметром может быть имя стандартного или определенного пользователем типа. В данном примере использованы стандартные типы int и char. Использовать имя vector без указания фактического параметра шаблона нельзя, т. к. никакое умалчиваемое значение при этом не предусматривается.

В списке параметров шаблона могут присутствовать формальные параметры, не определяющие тип, точнее – это параметры, для которых тип фиксирован.

Рассмотрим пример использования шаблонного класса, реализующего одномерный вектор. Реализовать некоторые методы для работы с вектором, а также перегрузить операцию [].

#include <iostream>

#include <string.h>

#include <locale.h>

#include <iomanip>

#include <typeinfo.h>

using namespace std;

template <class T>

class vector

{

T \*ms; // указатель на вектор

int size,ind; // размер и индекс

public:

vector() : size(0),ind(0),ms(NULL) {}

vector(int);

~vector()

{

delete [] ms;

}

void set(const T&); // увеличение размера массива на один элемент

T\* get\_vect(); // возвращается указатель на массив

int get\_size(); // возвращается размер массива

T &operator[](int); // определение обычного метода

};

template <class T>

vector<T>::vector(int SIZE) : size(SIZE), ind(0)

{

ms=new T[size];

const type\_info & t=typeid(T); // получение ссылки t на

const char\* s=t.name(); // объект класса type\_info

for(int i=0;i<size;i++) // в зависимости от типа T

if(!strcmp(s,"char")) \*(ms+i)=' '; // заносим пустой символ

else \*(ms+i)=0; // или заносим цифру нуль

}

template <class T>

void vector<T>::set(const T &t) // увеличение размера массива на один элемент

{

T \*tmp = NULL;

if(++ind>=size)

{

tmp=ms;

ms=new T[size+1]; // ms−указатель на новый массив

}

if(tmp) memcpy(ms,tmp,sizeof(T)\*size); // перезапись tmp -> ms

ms[size++]=t; // добавление нового элемента

if(tmp) delete [] tmp; // удаление временного массива

}

template <class T>

T\* vector<T>::get\_vect()

{

return ms;

}

template <class T>

int vector<T>::get\_size()

{ return size;

}

template <class T>

T & vector<T>::operator[](int n) // определение обычного метода

{

if(n<0 || (n>=size)) n=0; // контроль за выходом из вектора

return ms[n];

}

int main()

{

vector <int> VectInt;

vector <char> VectChar;

VectInt.set(3);

VectInt.set(26);

VectInt.set(12); // получен int-вектор из трех атрибутов

VectChar.set('a');

VectChar.set('c'); // получен char-вектор из двух атрибутов

cout << VectInt[0]<< endl;

cout << VectChar[0] << endl;

VectInt[0]=1;

VectChar[1]='b';

int \*m\_i=VectInt.get\_vect();

for(int i=0; i<VectInt.get\_size(); i++ )

cout<<setw(3)<< \*(m\_i+i); cout<< endl; // вывод целочисленного вектора

char \*m\_c=VectChar.get\_vect();

for(int i=0; i<VectChar.get\_size(); i++ )

cout<<setw(3)<< \*(m\_c+i); cout<< endl; // вывод символьного вектора

return 0;

}

**Контрольные вопросы**

1. Для чего используются шаблоны классов? Что у них общего с шаблонами функций?
2. Как описываются шаблоны классов?
3. Как создать объект на основе класса, порожденного шаблоном?
4. Каких типов могут быть фактические параметры шаблонов классов?
5. Можно ли описывать в списке параметров шаблона параметры, не определяющие тип?

**Порядок выполнения работы**

1. Изучить краткие теоретические сведения.
2. Ознакомиться с материалами литературных источников.
3. Ответить на контрольные вопросы.
4. Разработать алгоритм программы.
5. Написать, отладить и выполнить программу.

**Варианты заданий**

1. Написать программу, в которой определяется шаблон для функции max(x,y), возвращающей большее из значений x и y. Написать перегруженную функцию max(char\*,char\*), возвращающую большую из передаваемых ей символьных строк. В каждой из функций предусмотреть вывод сообщения о том, что вызвана шаблонная или специализированная функция и вывод найденного большего.
2. Реализовать шаблон класса Tree, реализующий бинарное дерево. Для представления узлов дерева использовать шаблон класса TreeNode. Определить функции добавления узла, обхода всего дерева в нисходящем и в восходящем порядке, поиска по дереву. На основе созданного класса написать программу вычисления арифметических выражений.
3. Написать шаблон класса вектор, который принимает элементы любого типа. В классе реализовать методы, например, поиска, сортировки, удаления и т. д. При реализации методов использовать динамическую идентификацию типа элементов объекта.
4. Описать шаблонный класс для работы с односвязными списками. Для объектов класса определить операции проверки списка на пустоту, добавления элемента в начало списка, в конец списка, подсчет числа вхождений элемента в список, удаление элемента из списка. Продемонстрировать работу с шаблонным классом для списка с целыми элементами и с элементами-строками.
5. Для каждого варианта (табл. 2) разработать шаблон соответствующего класса, где поля могут иметь различные типы данных (некоторые поля могут быть статическими). Предусмотреть наличие в классе указанных методов и перегруженных операций.

Таблица 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер  варианта | Название класса | Поля | Методы | Перегружаемые операции |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Стек | Указатель на вершину, текущий размер | Добавить, извлечь элемент, проверить на пустоту, распечатать | += (поместить в стек), префиксный – – (извлечь из стека) |
| 2 | Элемент односвязного списка | Указатель на начало, информационное поле элемента, ссылка на следующий элемент | Добавить элемент в начало, удалить из начала, найти элемент | << (распечатать весь список), == (равны ли размеры двух списков?) |
| 3 | Элемент двусвязного списка | Указатели начала и конца, информационное поле элемента, ссылки на следующий и предыдущий элементы | Добавить элемент в конец, удалить из конца, распечатать список | << (распечатать весь список), == (равны ли размеры двух списков?) |
| 4 | Квадратное уравнение | Коэффициенты, дискриминант, корни | Вывести на экран уравнение, решить уравнение, вывести корни | Бинарный –, \*= (каждый коэффициент умножить на число) |
| Окончание таблицы 2 | | | | |
| 1 | 2 | 3 | 4 | 5 |
| 5 | Матрица | Две размерности, указатель на элементы | Вывести на экран, найти максимум и среднее значение элементов | + (сложение двух матриц), < (каждый элемент первой матрицы меньше соответствующего элемента второй) |
| 6 | Дата | Число, месяц, год | Изменить значения полей, определить, сколько дней осталось до нового года, вывести в формате «чч/мм/гггг» | != (проверка двух дат на неравенство), бинарный – (промежуток между двумя датами в днях, месяцах, годах) |
| 7 | Время | Часы, минуты, секунды | Изменить значения полей, вывести в формате «чч:мм:сс», определить, является ли заданное время «до полудня» или «после полудня» | >> (ввод новых значений полей), += (добавить заданное количество секунд) |
| 8 | Множество | Элементы множества, текущее количество элементов | Проверка на включение элемента в множество, вывод на экран, удаление элемента из множества | += (добавление элемента в множество), \* (пересечение множеств) |