ЛАБОРАТОРНАЯ РАБОТА № 7
ИССЛЕДОВАНИЕ ШАБЛОНОВ ФУНКЦИЙ»

7.1 Цель работы

Исследование назначения и способа описания шаблонов функций, применение их при написании объектно-ориентированных программ.

7.2 Краткие теоретические сведения

7.2.1 Шаблоны и их применение

Шаблоны позволяют давать обобщенные, в смысле произвольности используемых типов данных, определения функций и классов. Поэтому их часто называют параметризованными функциями и параметризованными классами. Часто также используются термины «шаблонные функции» и «шаблонные классы».

Шаблон функции представляет собой обобщенное определение функции, из которого компилятор автоматически создает представитель функции для заданного пользователем типа (или типов) данных. Когда компилятор создает по шаблону функции конкретную ее реализацию, то говорят, что он создал **порожденную функцию**. Синтаксис объявления шаблона функции имеет следующий вид:

```
template <class T1|T1 идент1, class T2|T2 идент2, …,
          class Tn|Tn идентn>
 возвр_тип имя_функции (параметры){
       /*тело функции*/
  }
```

За ключевым словом **template** следуют один или несколько параметров, заключенных в угловые скобки, разделенные между собой запятыми. Каждый параметр является либо ключевым словом **class**, за которым следует имя типа, либо именем типа, за которым следует идентификатор.

Для задания параметризованных типов данных вместо ключевого слова class может также использоваться ключевое слово **typename**. Параметры шаблона, следующие за ключевыми словами **class** или **typename**, называют **параметризованными типами**. Они информируют компилятор о том, что некоторый (пока что неизвестный) тип данных используется в шаблоне в качестве параметра (в момент вызова шаблона на место такого параметризованного типа станет тип, заданный программистом). Параметры шаблона, состоящие из имени типа и следующего за ним имени идентификатора, информируют компилятор о том, что параметром шаблона является константа указанного типа. Шаблонную функцию можно вызывать как обычную, никакого специального синтаксиса не требуется.

Рассмотрим пример определения и вызова шаблонных функций:

```
#include <iostream>
using namespace std;
template <class T>    // Шаблон функции, вычисляющей квадрат
// введенного значения
T Sqr(T x)   {        // здесь T — некий общий тип, который будет
    return x*x;       // задан при вызове функции в теле основной
}                     // программы

template < class T>   // Шаблон функции обмена двух элементов
                      // в массиве по значению их индексов
T* Swap(T* t, int ind1, int ind2) {
    T tmp = t[ind1];        // Собственно обмен
    t[ind1] = t[ind2];
    t[ind2] = tmp;
    return t;
}
int main() {
    int n=10,               // Для возведения в квадрат
        i = 2, j = 5        // Индексы в массиве
    double d = 10.21        // Для возведения в квадрат
    char* str = "Шаблон";       // Массив букв
    cout << "Значение n = " << n << endl
         << "Его квадрат = " << Sqr (n) << endl
                            // Вызов шаблона для возведения
         << "Значение d = " << d << endl
                            // в квадрат целого числа
         <<"Его квадрат = " << Sqr(d) << endl
                            // Вызов шаблона для возведения
         << "Исходная строка = " << str << endl
                            // в квадрат веществ. числа
                            // Преобразование массива
  << "Преобразованная строка = " << Swap(str, i, j) << endl;
  return 0;
}
```

При выполнении программа выводит на экран следующее:

```
Значение n = 10
Его квадрат = 100
Значение d = 10.21
Его квадрат = 104.2441
Исходная строка = Шаблон
Преобразованная строка = Шанлоб
```

Обратите внимание на вызовы шаблонных функций, сделанные в предыдущем примере: **Sqr(n)** и **Sqr(d).** Каждый такой вызов приводит к

построению конкретной **порожденной функции**. В данном случае первый вызов приводит к построению компилятором функции **`Sqr()`** для целого типа данных, во втором — для типа double. В связи с этим процесс построения порожденной функции компилятором называют **конкретизацией шаблонной функции**.

Как и для обычных функций, можно создать прототип шаблона функции в виде его предварительного объявления. Например:

```
template < class T>
T* Swap(T* t, int ind1, int ind2);
```

Каждый типовой параметр шаблона должен появиться в списке формальных параметров функции. Например, следующее объявление приведет к ошибке во время компиляции:

```
template < class T, class U>
T FuncName(U);
```

7.2.3 Недостатки шаблонов

Шаблоны предоставляют определенные выгоды при программировании, связанные с широкой применимостью кода и легким его сопровождением. В общем, этот механизм позволяет решать те же задачи, для которых используется полиморфизм. С другой стороны, в отличие от макросов, они позволяют обеспечить безопасное использование типов данных. Однако с их использованием связаны и некоторые недостатки:

- программа содержит полный код для всех порожденных представителей шаблонного класса или функций;
- не для всех типов данных предусмотренная реализация класса или функции оптимальна.

Преодоление второго недостатка возможно с помощью специализации шаблонов.

7.3 Порядок выполнения лабораторной работы

7.3.1 В ходе самостоятельной подготовки изучить основы работы с шаблонами функций и классов.

2 Разработать программу на языке C++, которая обрабатывает данные разных типов (**`int, char`**, и др.). Функция обработки данных должна быть реализована как шаблон.

3 Разработать тестовые примеры.

4 Выполнить отладку программы.

5 Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы.

6 Оформить отчет по проделанной работе.

7.4 Варианты заданий

Описать функцию-шаблон, заданную по варианту. Проиллюстрировать ее корректную работу на различных по типу наборах данных (не менее трех: **`int,`**

char и др.). А также написать функцию-шаблон для заполнения массива и функцию-шаблон для вывода элементов массива на экран.

Вариант 1

Написать функцию-шаблон, выполняющую транспонирование матрицы.

Вариант 2

Написать функцию-шаблон, меняющую местами i-ю строку и j-й столбец в квадратной матрице.

Вариант 3

Написать функцию-шаблон сортировки массива по возрастанию методом пузырька.

Вариант 4

Написать функцию-шаблон, определяющую, существуют ли в матрице повторяющиеся строки.

Вариант 5

Написать функцию-шаблон, определяющую элемент, который встречается в массиве максимальное число раз.

Вариант 6

Написать функцию-шаблон, определяющую, существуют ли в матрице повторяющиеся столбцы.

Вариант 7

Написать функцию-шаблон, вычисляющую максимальное значение элемента в массиве.

Вариант 8

Написать функцию-шаблон, переставляющую i-ю и j-ю строки в матрице.

Вариант 9

Написать функцию-шаблон, записывающую массив в обратном порядке.

Вариант 10

Написать функцию-шаблон, меняющую диагонали матрицы местами.

Вариант 11

Написать функцию-шаблон последовательного поиска в массиве по ключу. Функция возвращает индекс первого найденного элемента в массиве, равного ключу.

Вариант 12

Написать функцию-шаблон, заменяющую в матрице все элементы A на элемент B (значения A и B передаются параметрами в функцию-шаблон).

Вариант 13

Написать функцию-шаблон, циклически сдвигающую одномерный массив элементов n раз (1-й элемент становится 2-м, 2-й – 3-м …n-й элемент становится первым).

Вариант 14

Написать функцию-шаблон, меняющую в одномерном массиве соседние элементы (поменять элементы с четными индексами на элементы с нечетными индексами).

Вариант 15

Написать функцию-шаблон, проверяющую матрицу на симметричность.

Вариант 16

Написать функцию-шаблон, проверяющую: верно, что все элементы матрицы одинаковые.

Вариант 17

Написать функцию-шаблон, проверяющую: верно, что элементы в одномерном массиве расположены по убыванию.

Вариант 18

Написать функцию-шаблон, проверяющую: верно, что максимальный и минимальный элемент матрицы расположены в одной строке.

Вариант 19

Написать функцию-шаблон, проверяющую: верно, что в одномерном массиве есть хотя бы два одинаковых элемента.

Вариант 20

Написать функцию-шаблон, проверяющую на равенство две строки матрицы (номера строк вводит пользователь).

Вариант 21

Написать функцию-шаблон, меняющую местами минимальный и максимальный элементы матрицы.

Вариант 22

Написать функцию-шаблон, проверяющую на равенство значений двух элементов массива, индексы которых задает пользователь.

Вариант 23

Написать функцию-шаблон, подсчитывающую количество нулевых элементов.

Вариант 24

Написать функцию-шаблон, сдвигающую элементы одномерного массива вправо на n позиций (значение n вводит пользователь).

Вариант 25

Написать функцию-шаблон, проверяющую: верно, что элементы на главной и вспомогательной диагоналях в квадратной матрице равны.

7.5 Содержание отчета о выполнении лабораторной работы

Титульный лист, цель работы, вариант задания, текст программы с комментариями, описание тестовых примеров и выводы по проделанной работе.

7.6 Контрольные вопросы

7.6.1 Дайте определение понятию «шаблон», приведите пример.
Опишите случаи в которых выгодно применение шаблонов.

7.6.3 Перечислить особенности работы шаблонов.

7.6.4 Дайте определение понятию «порожденная функция».

7.6.5 Перечислить достоинства и недостатки шаблонов
7.6.6 Дайте определение понятию «конкретизация шаблонной функции».
7.6.7 Дайте определение понятию «параметризованные типы».