Лабораторная работа № 5

**Применение шаблонов: Обобщенные функции.**

**Цель работы:** Формирование навыка создания шаблонов функций.

**Теория**

Шаблоны – это мощный инструмент языка С++, позволяющий упростить программирование и достичь одну из самых трудных целей программирования –создание многократно используемого кода. Шаблонов не было в первоначальной версии, но впоследствии они стали незаменимой частью языка, сейчас механизмы шаблонов поддерживаются всеми известными компиляторами С++.

С помощью шаблонов можно создавать обобщённые функции и классы, которые работают с типом данных как с параметром. Таким образом одну и ту же функцию (или класс) можно использовать различных типов данных не предоставляя явным образом конкретные версии для каждого типа данных.

Очень многие алгоритмы носят универсальный характер, например сортировка и поиск минимального элемента массива, помещение и выборка элемента очереди и тому подобное, во всех этих случаях можно использовать шаблоны функций.

Написав и отладив шаблонный класс, программист получает законченный программный компонент, который избавляет от необходимости создавать отдельные реализации для различных типов данных.

**Обобщенные функции(шаблоны функций)**

Шаблон позволяет определять семейство функций. Это семейство характеризуется общим алгоритмом, применяемым к данным различных типов.

В определении функции задают универсальный набор операций, реализующий алгоритм работы, тип же данных, с которыми работает функция, передается в качестве параметра. После этого компилятор автоматически сгенерирует код для типа данных, который реально используется при вызове этой функции. По сути, создавая обобщённую функцию, мы получаем функцию, которая автоматически перегружает саму себя.

Для создания обобщённой функции (или класса) используется ключевое слово template(шаблон). Шаблон описывает действия, выполняемые функцией, компилятору же остаётся дополнить недостающие детали в соответствии с заданным значением параметра (типом данных).

Приведем пример обобщенной функции, меняющей местами две переменные, так как алгоритм перестановки не зависит от типа переменных, его можно описать с помощью шаблона функции.

Пример: переставить две переменные

#include"stdafx.h"

#include<iostream>

usingnamespace **std;**

template**<**class **X>** // заголовок

void **swapvar (X &a, X &b)**

{ X temp; // тело

**temp=a;**

**a = b;**

**b = temp;**

}

int **\_tmain(**int **argc, \_TCHAR\* argv[])**

{ int **i=10, j=20;**

double **x=11.1, y=22.2;**

char **a=**'a',b='z';

cout<<"До перестановки :"<<endl;

cout<<i<<' '<<j<<endl;

cout<<x<<' '<<y<<endl;

cout<<a<<' '<<b<<endl;

**swapvar (i , j);**

**swapvar(x , y);**

**swapvar(a , b);**

cout<<"После перестановки :";

cout<<endl;

cout<<i<<' '<<j<<endl;

cout<<x<<' '<<y<<endl;

cout<<a<<' '<<b<<endl;

system ("pause");

return **0;**

}

**Результат выполнения программы:**

**До перестановки :**

**10 20**

**11.1 22.2**

**a z**

**После перестановки :**

**20 10**

**11.1 22.2**

**z a**

Ключевое слово templateозначает «шаблон», параметр X задает обобщенный тип данных для функции (формальные параметры и переменные функции имеют тип X). Компилятор автоматически создаст три версии swapvar(): для целых, действительных и символьных данных(количество версий определяют вызовы функций).

Конкретная версия функции, является экземпляром обобщенной функции и называется генерируемой функцией (в некоторых источниках–cпециализацией)

**Пример:**функция с двумя обобщенными типами.

template**<**class **type1,** class **type2>**

void **display (type1 a, type2 b)**

{ cout<<a<<' '<<b<<endl;

}

int **\_tmain(**int **argc, \_TCHAR\* argv[])**

{

int **s=1;**

display(s," кГяблок ");

display("Для продолжения введите",'Y');

}

**Результат выполнения программы:**

**1 кГ яблок**

**Для продолжения введите Y**

Генерация экземпляров функции display():

* Типы параметров **int , char\***
* Типы параметров **char\*, char**

Компилятор будет генерировать столько различных вариантов шаблонной функции, сколько существует способов её вызова.

Шаблонная функция перегружает сама себя, так как автоматически создаются несколько функций с одним и тем же именем, но с разными типами параметров.

**Явная перегрузка обобщенной функции.**

Когда программиста не устраивает автоматическая реализация функции для отдельного типа данных, он может перегрузить её явно.

Явно перегруженная функция «маскирует» (или замещает) обобщенную функцию, для данного типа данных.

Пример:

#include"stdafx.h"

#include"stdafx.h"

#include<iostream>

usingnamespace **std;**

template**<**class **X>**

void **swapvar (X &a, X &b)**

{ X temp;

**temp=a;**

**a = b;**

**b = temp;**

cout<<" работает обобщенная функция"**<<endl;**

}

// явная перегрузка

void **swapvar (**int**&a,** int**&b)**

{ int **temp;**

**temp=a;**

**a = b;**

**b = temp;**

**cout<<**" работает перегруженная функция"<<endl;

}

int **\_tmain(**int **argc, \_TCHAR\* argv[])**

{int **i=10, j=20;**

double **x=11.1, y=22.2;**

cout<<"c параметрами i , j"<<endl;

**swapvar (i , j);**

cout<<"с параметрами х , y"<<endl;;

**swapvar (x , y);**

**system(**"pause");

return **0;**

}

**Результат выполнения программы:**

**С параметрами i , j работает перегруженная функция**

**С параметрами х , y работает обобщенная функция**

**Стандартные параметры в шаблонах функции.**

В шаблонных функциях можно совмещать стандартные и обобщенные параметры.

Пример: Повторение параметра

#include"stdafx.h"

#include<iostream>

usingnamespace **std;**

template**<**class **X>**

void **repeat (X &a,** int **n)**

{ int **i;**

for **(i=0; i<n; i++)**

cout<<a<<' ';

// cout<<a<<' ';

cout <<endl;

}

int **\_tmain(**int **argc, \_TCHAR\* argv[])**

{int **i = 9;**

char **s =** '\*';

char **str[]=**"Ї« бвЁ­Єг § Ґ«®!\n";

**repeat (i , 10);**

**repeat (s , 5);**

repeat (str , 3);

system ("pause");

return **0;**

}

**Результатвыполненияпрограммы:**

9 9 9 9 9 9 9 9 9

\* \* \* \* \*

Пластинкузаело! Пластинку заело! Пластинку заело!

Подведем итоги : шаблонные функции подобны перегружаемым функциям, но имеют некоторые ограничения в применении.

К алгоритму перегружаемой функции не предъявляется никаких требований, обычно в различных реализациях функции задаются различные действия.

К обобщенной функции такие требования предъявляется. Обобщенная функция должна выполнять одно и то же действие для всех своих возможных версий, отличие между версиями состоит только в типе обрабатываемых данных.

**Контрольные вопросы**

1. Для чего нужны шаблоны?
2. Дайте понятия: шаблона, обобщенной функции, обобщенного типа
3. Синтаксис создания шаблона функции (обобщенной функции).
4. Применение ключевых слов class и template (приведите пример)
5. В каких случаях требуется явная перегрузка обобщенной функции
6. Применение стандартных параметров в обобщенных функциях.

**Общие правила и требования**

1. Написать функции для заполнения массивов случайными числами, ввода данных с клавиатуры, вывода на экран
2. Использовать способы адресации элементов массива с помощью индекса или через указатель (последнее приветствуется и добавляет Вам баллы)
3. Оформить задание в виде шаблона функции и продемонстрировать работу с различными типами данных

|  |  |
| --- | --- |
| Вариант задания | Задание |
| 1, 13, 25 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации (с клавиатуры) * для вывода информации на экран по n чисел в строке (n задать в качестве параметра)   Явно перегрузить функцию инициализации для типа char, контролировать ошибки ввода (не допускать ввод цифр)  Продемонстрировать работу с типами данных: int, double, char |
| 2,14,26 | Создать массив, размерность указать в качестве параметра (инициализация случайными числами)  Создать шаблонные функции:   * для поиска максимального элемента и его местоположения * для вывода всех элементов после максимального   Явно перегрузить функцию вывода для типа double - вывести все элементы, расположенные до максимального  Продемонстрировать работу с типами данных: int, double |
| 3,15,27 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации (ввод с клавиатуры) * для вычисления количества нулевых элементов.   Явно перегрузить функцию инициализации для типа double – вводить числа большие 10, иначе выводить сообщение «числа должны быть больше 10».  Продемонстрировать работу с типами данных: int, double |

|  |  |
| --- | --- |
| 4,16,28 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации случайными числами * с клавиатуры ввести число K. Создать массив из индексов элементов, значение которых не превосходит K   Явно перегрузить функцию инициализации для типа char – инициализацию проводить буквами латинского алфавита.  Продемонстрировать работу с типами данных: double, char |
| 5,17,29 | Создать массив, размерность указать в качестве параметра (инициализация случайными числами). Создать шаблонные функции:   * для сортировки массива по убыванию * для вывода информации на экран по n чисел в строке (n задать в качестве параметра)   Явно перегрузить функцию сортировки для типа char, сортировку не проводить, вывести сообщение «сортировка не нужна»  Продемонстрировать работу с типами данных: int, double, char |
| 6,18,30 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации, ввод с клавиатуры * для вычисления суммы элементов, меньших n (n задать в качестве параметра)   Явно перегрузить функцию инициализации для типа double, использовать датчик случайных чисел, обеспечить генерацию как положительных, так и отрицательных чисел  Явно перегрузить функцию вычисления суммы элементов для типа char – вместо вычислений, вывести сообщение «суммирование не проводится»  Продемонстрировать работу с типами данных: char, int, double |
| 7,19 | Создать массив, размерность указать в качестве параметра (инициализация случайными числами). Создать шаблонные функции:   * для сортировки массива по возрастанию * для вывода информации на экран.   Явно перегрузить функцию сортировки для типа double – сортировать по убыванию. Продемонстрировать работу с типами данных:int, double, char |
| 8,20 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации случайными числами * для поиска минимального элемента и его местоположения   Явно перегрузить функцию инициализации для типа char , коды символов должны лежать в заданном диапазоне (диапазон задается параметрами).  Продемонстрировать работу с типами данных: int, double, char |

|  |  |
| --- | --- |
| 9,21 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации случайными числами * определение индексов элементов, значение которых не превосходит n   (n задать в качестве параметра)  Явно перегрузить функцию инициализации для типа char (данные вводить с клавиатуры)  Явно перегрузить функцию определение индексов элементов для типа char – индексы не определять, вывести сообщение «для данных типа char индексы не определяются»  Продемонстрировать работу с типами данных: int, double, char |
| 10,22 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации случайными числами * вставить в исходный массив число. С клавиатуры ввести само число и номер элемента, после которого выполняется вставка.   Явно перегрузить функцию вставки для типа char- с клавиатуры ввести символ, вставлять в начало массива  Продемонстрировать работу с типами данных: int, double, char |
| 11,23 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации, вводить как положительные, так и отрицательные значения (ввод с клавиатуры) * найти сумму фрагмента массива. С клавиатуры ввести два положительных числа (индексы элементов-границ фрагмента ).   Явно перегрузить функцию суммирования для типа int – суммировать только положительные числа.  Продемонстрировать работу с типами данных: int, double |
| 12,24 | Создать массив, размерность указать в качестве параметра.  Создать шаблонные функции:   * для инициализации случайными числами * определить минимальный и максимальный элементы и сумму фрагмента массива между ними.   Явно перегрузить функцию суммирования для типа double – суммировать от начала массива до минимального элемента  Продемонстрировать работу с типами данных: int, double |