|  |
| --- |
| **ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ** |
|  |
|  |
|  |
| **Практическое занятие №7.**  **Тема: Реализация шаблона класса** |
|  |
|  |
|  |

**Цель:** получение практических навыков разработки и использования шаблонов классов для хранения и обработки данных.

**Основные теоретические положения**

**Шаблон функции**

Обобщенная (родовая) функция определяет базовый набор операций, которые будут применяться к разным типам данных. Обобщенная функция (шаблон) оперирует с тем типом данных, который она получает в качестве параметра. С помощью этого механизма одна и та же процедура может применяться к самым разным данным. Как известно, многие алгоритмы логически одинаковы, независимо от того, для обработки каких типов данных они предназначены. Например, алгоритм быстрой сортировки одинаков как для массивов целых, так и для массивов действительных чисел. Это именно тот случай, когда сортируемые данные отличаются только по типам. Благодаря созданию родовой функции вы можете независимо от типа данных определить суть алгоритма. После того как это сделано, компилятор автоматически генерирует правильный код для фактически используемого при выполнении функции типа данных.

Обобщенная функция создается с помощью ключевого слова **template**. Обычное значение этого слова (т. е. шаблон) точно отражает его назначение в C++. Оно предназначено для создания шаблона (или каркаса), который описывает то, что будет делать функция, при этом компилятору остается дополнить каркас необходимыми деталями. Ниже представлена типовая форма определения функции-шаблона:

**template<class Фтип>возвр\_значение имя\_функции(список\_парам**)

**{**

**// тело функции**

**}**

Здесь вместо **Фтип** указывается тип используемых функцией данных. Это имя можно указывать внутри определения функции. *Однако это всего лишь* *фиктивное имя, которое компилятор автоматически заменит именем реального типа данных при создании конкретной версии функции.*

Хотя традиционно для задания обобщенного типа данных в объявлении шаблона указывается ключевое слово **class**, также можно использовать ключевое слово **typename**.

В следующей программе создается шаблон функции, которая меняет местами значения двух переменных, передаваемых ей в качестве параметров. Поскольку в своей основе процесс обмена двух значений не зависит от типа переменных, этот процесс удачно реализуется с помощью обобщенной функции.

// Пример обобщенной функции или шаблона

**#include <iostream>**

**using namespace std;**

// Это функция-шаблон

**template <class X> void swapargs(X &a, X &b)**

**{**

**X temp;**

**temp = a;**

**a = b;**

**b = temp;**

**}**

**int main()**

**{**

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

**float x = 10.1, у = 23.3;**

**cout<<"Исходные значения i,j равны:"<< i <<' '<<j<<endl;**

**cout<<"Исходные значения x,у равны:"<< x <<' '<<у<<endl;**

**swapargs (i, j); // обмен целых**

**swapargs (x, y) ; // обмен действительных**

**cout <<"Новые значения i,j равны:" << i <<' '<<j<<endl;**

**cout <<"Новые значения x,у равны:" << x <<' '<<у<<endl;**

**return 0;**

**}**

Ключевое слово template используется для определения обобщенной функции. Строка: **template <class X> void swapargs(X &а, X &b)** сообщает компилятору две вещи: во-первых, создается шаблон, и, во-вторых, начинается определение обобщенной функции. Здесь X — это обобщенный тип данных, используемый в качестве фиктивного имени. После строки с ключевым словом template функция swapargs() объявляется с именем X в качестве типа данных обмениваемых значений. В функции main( ) функция swapargs() вызывается с двумя разными типами данных: целыми и действительными. Поскольку функция swapargs() — это обобщенная функция, компилятор автоматически создает две ее версии: одну — для обмена целых значений, другую для обмена действительных значений.

С помощью инструкции **template** можно определить более одного родового типа данных, отделяя их друг от друга запятыми. Например, в данной программе создается родовая функция, в которой имеются два родовых типа данных:

**#include <iostream>**

**using namespace std;**

**template <class typel, class type2>**

**void myfunc (typel x, type2 y)**

**{**

**cout << x << ' ' << у << endl;**

**}**

**int main( )**

**{**

**myfunc (10, "hi") ;**

**myfunc(0.23, 10L) ;**

**return 0;**

**}**

В данном примере при генерации конкретных экземпляров функции

**myfunc()** фиктивные имена типов **typel** и **type2** заменяются компилятором на

типы данных **int** и **char\*** или **double** и **long** соответственно.

**Шаблон класса**

В дополнение к обобщенным функциям можно определить и обобщенные классы. При этом создается класс, в котором определены все необходимые алгоритмы, а фактические типы обрабатываемых данных задаются в качестве параметров позже, при создании объектов этого класса. Обобщенные классы полезны, когда класс содержит общую логику работы. Например, алгоритм, который реализует очередь целых, будет также работать и с очередью символов. Кроме того, механизм, который реализует связанный список почтовых адресов, будет также поддерживать связанный список запасных частей к автомобилям. С помощью обобщенного класса можно создать класс, реализующий очередь, связанный список и т. д. для любых типов данных. Компилятор будет автоматически генерировать правильный тип объекта на основе типа, заданного при создании объекта.

Ниже представлена основная форма объявления обобщенного класса:

**template <class Фтип> class имя\_класса**

**{**

**// поля, методы**

**};**

Здесь Фтип — это фиктивное имя типа, который будет задан при создании

экземпляра класса. При необходимости можно определить более одного обобщенного типа данных, разделяя их запятыми.

После создания обобщенного класса с помощью представленной выше формы можно создать конкретный экземпляр этого класса:

**имя\_класса <тип> объект;**

Здесь тип — это имя типа данных, с которым будет оперировать класс.

Чтобы научиться пользоваться библиотекой стандартных шаблонов с максимальной эффективностью, необходимо иметь твердые знания по классам-шаблонам и их синтаксису.

Функции-члены обобщенного класса сами автоматически становятся обобщенными. Для них не обязательно явно задавать ключевое слово template.

В C++ имеется встроенная библиотека классов-шаблонов, которая называется библиотекой стандартных шаблонов (Standard Template Library, STL). Эта библиотека предоставляет обобщенные версии классов для наиболее часто используемых алгоритмов и структур данных.

В следующем примере шаблона класса рассмотрим банковский счет. Банковский счет должен иметь номер, однако на момент написания класса может быть неизвестно, какой тип будет представлять номер счета - это может быть числовой номер 7833545, а может и быть просто набор символов в виде строки, например, "12348fdhRam". На первый взгляд, можно просто определить два класса. Но для того и применяются шаблоны чтобы создать «универсальный» класс, пригодный в обоих случаях.

**#include <iostream>**

**#include <string>**

**using namespace std;**

**template <typename T>**

**class Account {**

**private:**

**T id;**

**public:**

**Account(T ide) {id = ide;}**

**T getId() {**

**return id;**

**}**

**};**

**int main()**

**{**

**Account<string> acc1("12345Kostrov");**

**Account<int> acc2(1234566);**

**cout << "acc1: " << acc1.getId() << endl;**

**cout << "acc2: " << acc2.getId() << endl;**

**return 0;**

**}**

При использовании шаблона класса необходимо в угловых скобках после названия класса указать конкретный тип, который будет применяться вместо параметра T. Так, в первом случае вместо T применяется тип **string**, поэтому в конструктор класса можно передать строку:

**Account<string> acc1("12345Kostrov");**

Во втором случае применяется тип **int**, поэтому в конструктор передается число:

**Account<int> acc2(1234566);**

В следующем усложненном примере показан вариант программы, в которой реализация обобщенного класса производится данными из объектов другого класса и с ними производятся операции с использованием средств обобщенного класса.

// Шаблон класса

**template <typename T>**

**class Shablon {**

**private:**

**T id;**

**public:**

**Shablon(){}**

**Shablon(T ide) {id = ide;}**

**T getId() {return id;}**

**};**

// Класс для данных

**class Base{**

**int property;**

**string brand;**

**public:**

**Base (int property, string brand){**

**this->property=property;**

**this->brand=brand;**

**}**

**int getProperty() {return property;}**

**string getBrand() {return brand;}**

**};**

**int main()**

**{**

// Создание массива объектов класса Base

**Base base[] = {Base(16,"IBM"),**

**Base(8,"HP"),**

**Base(16,"Dell"),**

**Base(32,"Apple"),**

**Base(4,"HP")**

**};**

//Реализация шаблона класса для типа int

**Shablon<int>iOb[5];**

**for(int i=0; i<5; i++)**

**iOb[i]=base[i].getProperty();**

// Поиск минимального элемента

**int minimum=iOb[0].getId();**

**for(int i=1; i<5; i++){**

**if(iOb[i].getId()<minimum)**

**minimum=iOb[i].getId();**

**}**

**cout << "Minimum = " << minimum << endl;**

//Реализация шаблона класса для типа string

**Shablon<string>strOb[5];**

**for(int i=0; i<5; i++)**

**strOb[i]=base[i].getBrand();**

// Определение числа объектов с брендом "HP"

**int count=0;**

**for(int i=0; i<5; i++){**

**if(strOb[i].getId()=="HP")**

**count++;**

**}**

**cout << "Number of HP = " << count << endl;**

**return 0;**

**}**

**Практические задания**

*Выберите один из вариантов задания.*

**Вариант А.**

**Задание 1.** В соответствии с *индивидуальным заданием* (см. приложение) создать класс для предметной области.

**Задание 2.** Спроектировать шаблон класса с одним обобщенным типом.

**Задание 3.** Разработать реализацию шаблона класса для каждого из двух вычисляемых показателей.

**Вариант В.**

**Задание 1.** В соответствии с *индивидуальным заданием* (см. приложение) создать класс для предметной области.

**Задание 2.** Спроектировать шаблон класса с одним обобщенным типом.

**Задание 3.** Разработать реализацию шаблона класса для одного из двух вычисляемых показателей для разных типов данных.

**Вариант С.**

**Задание 1.** В соответствии с *индивидуальным заданием* (см. приложение) спроектировать шаблон функции, предназначенной для определения одного вычисляемого показателя.

**Задание 2.** Реализовать шаблон, передавая в функцию данные требуемого типа.

Отчет оформляется по общеустановленным правилам в *электронном виде* со следующим содержанием:

1. титульный лист,
2. тема и цель практического занятия,
3. задание на практическое занятие,
4. текст программы с комментариями,
5. результаты работы программы и
6. выводы по разработанным элементам программы.

Приложение

|  |  |  |
| --- | --- | --- |
| **№**  **п/п** | **Предметная область** | **Вычисляемые показатели** |
|  | Компьютер | Экземпляр с наибольшей оперативной памятью  Количество компьютеров заданной фирмы |
|  | Локальная сеть | Минимальная стоимость монтажа  Количество компьютеров в сети |
|  | Транспортное средство | Максимальная емкость бензобака  Количество транспортных средств заданного типа |
|  | Программный продукт | Последняя версия  Количество программных продуктов заданной фирмы |
|  | Документ | Документ, выданный раньше всех других  Количество документов определённого типа |
|  | Диета | Максимальное дневное количество белков  Средняя продолжительность |
|  | Периферийное устройство компьютера | Минимальная цена устройства  Количество устройств определённой фирмы |
|  | Строительный товар | Сумма покупки  Количество товаров заданного типа |
|  | Представитель университета | Количество студентов, обучающихся у конкретного преподавателя  Общее количество студентов |
|  | Предприятие малого бизнеса | Название предприятия с максимальным числом сотрудников  Число парикмахерских |
|  | СУБД | Количество СУБД заданного производителя  Последняя версия |
|  | Страховой полис | Количество полисов на заданную фамилию  Полис с наименьшим номером |
|  | Наушники | Тип с максимальным частотным диапазоном  Тип с минимальным весом |
|  | Недвижимость | Максимальная жилая площадь квартиры  Количество квартир заданной компании |
|  | Часы | Самый дорогой экземпляр  Количество часов определённого производителя |
|  | Телефон | Самая новая модель  Число телефонов одной фирмы |
|  | Бытовая техника | Количество товаров заданной фирмы  Экземпляр с максимальной ценой |
|  | Магнитная карта для проезда на транспорте | Количество карт без поездок  Количество карт заданного типа |
|  | Насекомое | Максимальный размер крыльев  Количество экземпляров одного цвета |
|  | Канцелярские товары | Количество товаров определённой фирмы  Общая стоимость |
|  | Товар | Количество товаров заданной фирмы  Общая сумма покупки |
|  | Учащийся | Учащийся с максимальным IQ (коэффициентом интеллекта)  Средний возраст |
|  | Путешествие (тур) | Самый дешевый тур на 7 и более дней  Число туров в выбранную страну |
|  | Мебель | Количество предметов из дерева  Тип с максимальной ценой |
|  | Программное обеспечение | Продукт с максимальным объемом памяти  Количество продуктов заданной фирмы |
|  | Представление | Минимальное число зрителей в зале  Средняя цена билета |
|  | Запоминающее устройство | Экземпляр с минимальным размером  Число устройств одной фирмы |
|  | Принтер | Представитель с наибольшей производительностью  Число принтеров одного производителя |
|  | Книга | Количество книг одного автора  Название книги с наибольшим числом страниц |
|  | Телефон | Самая новая модель  Экземпляр с минимальным весом |