Тема: «Обобщения и шаблоны»

Цель: получить навыки создания обобщённых типов. Изучить шаблоны.

Время выполнения: 8 часов.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Универсальные шаблоны вводят в .NET концепцию параметров типов, что позволяет создавать классы и методы, которые откладывают спецификацию одного или нескольких типов до тех пор, пока класс или метод не будет объявлен и создан в клиентском коде. Как пример, ниже показан класс с параметром T универсального типа. Этот класс может использоваться в другом клиентском коде, не требуя ресурсов и не создавая рисков, связанных с операциями приведения и упаковки-преобразования в среде выполнения.

```
// Declare the generic class.
public class GenericList<T>
{
    public void Add(T input) { }
}
class TestGenericList
{
    private class ExampleClass { }
    static void Main()
    {
        // Declare a list of type int.
        GenericList<int> list1 = new GenericList<int>();
        list1.Add(1);

        // Declare a list of type string.
        GenericList<string> list2 = new GenericList<string>();
        list2.Add("");

        // Declare a list of type ExampleClass.
                    GenericList<ExampleClass>   list3   =   new
GenericList<ExampleClass>();
        list3.Add(new ExampleClass());
    }
}
```

Универсальные классы и методы сочетают такие характеристики, как возможность многократного использования, типобезопасность и

эффективность, которые не обеспечивают их не универсальные аналоги. Универсальные типы наиболее часто используются с коллекциями и методами, которые выполняют с ними операции. Пространство имен `System.Collections.Generic` содержит несколько универсальных классов коллекций. Не универсальные коллекции, например `ArrayList`, не рекомендуются и поддерживаются только для обеспечения совместимости.

Вы также можете создавать пользовательские универсальные типы и методы для предоставления собственных универсальных решений и шаблонов проектирования, которые являются типобезопасными и эффективными.

В следующем примере кода показан простой универсальный класс связанного списка для демонстрационных целей. (В большинстве случаев следует использовать класс `List<T>`, предоставляемый .NET, вместо создания собственного.) Параметр типа `T` используется в нескольких расположениях, где обычно используется конкретный тип для указания типа элемента, хранящегося в списке. Он используется в следующих случаях:

- в качестве типа параметра метода в методе `AddHead`;
- в качестве типа возвращаемого значения свойства `Data` во вложенном классе `Node`;
- в качестве типа закрытого члена `data` во вложенном классе.

`T` доступен для вложенного `Node` класса. Когда экземпляр `GenericList<T>` создается с конкретным типом, например `GenericList<int>`, каждое вхождение `T` будет заменено `int`.

```
// type parameter T in angle brackets
public class GenericList<T>
{
    // The nested class is also generic on T.
    private class Node
    {
        // T used in non-generic constructor.
        public Node(T t)
        {
            next = null;
            data = t;
        }

        private Node? next;
        public Node? Next
        {
            get { return next; }
```

```
            set { next = value; }
        }

        // T as private member data type.
        private T data;

        // T as return type of property.
        public T Data
        {
            get { return data; }
            set { data = value; }
        }
    }

    private Node? head;

    // constructor
    public GenericList()
    {
        head = null;
    }

    // T as method parameter type:
    public void AddHead(T t)
    {
        Node n = new Node(t);
        n.Next = head;
        head = n;
    }

    public IEnumerator<T> GetEnumerator()
    {
        Node? current = head;

        while (current != null)
        {
            yield return current.Data;
            current = current.Next;
        }
    }
}
```

В следующем примере кода показано, как клиентский код использует универсальный класс `GenericList<T>` для создания списка целых чисел.

Просто изменяя тип аргумента, следующий код можно легко изменить для создания списков строк или любого другого пользовательского типа:

```
class TestGenericList
{
    static void Main()
    {
        // int is the type argument
        GenericList<int> list = new GenericList<int>();

        for (int x = 0; x < 10; x++)
        {
            list.AddHead(x);
        }

        foreach (int i in list)
        {
            System.Console.Write(i + " ");
        }
        System.Console.WriteLine("\nDone");
    }
}
```

Общие сведения об универсальных шаблонах

- Используйте универсальные типы, чтобы получить максимально широкие возможности многократного использования кода, обеспечения безопасности типов и повышения производительности.
- Чаще всего универсальные шаблоны используются для создания классов коллекций.

Библиотека классов .NET содержит несколько универсальных классов коллекций в пространстве имен `System.Collections.Generic`. Универсальные коллекции следует по возможности использовать вместо классов, таких как `ArrayList` в `System.Collections` пространстве имен.

- Вы можете создавать собственные универсальные интерфейсы, классы, методы, события и делегаты.
- Универсальные классы можно ограничить, чтобы они разрешали доступ к методам только для определенных типов данных.
- Сведения о типах, используемых в универсальном типе данных, можно получить во время выполнения с помощью отражения.

Индивидуальные задания для лабораторной работы

Варианты:

1. Стек.
2. Очередь.
3. Список односвязный.
4. Список двусвязный.
6. Кольцо односвязное.
7. Кольцо двусвязное.
8. Матрица.

Каждый студент для своего варианта должен:

1. Построить шаблонный класс, который будет описывать элемент хранимых данных, доступ к ним, сравнение элементов и т.п. по необходимости.

2. Построить контейнерный шаблонный класс операций над элементами данных, включающий операции:

- добавления;
- удаления;
- поиска;
- просмотра;
- сортировки элементов;
- перестановки элементов в обратном порядке;
- замены всех подобных элементов по заданному ключу;
- поиска максимального элемента;
- остальные функции добавлять по необходимости.

3. Для данного контейнерного класса предусмотреть при формировании элемента задание режима уникальных элементов (т.е. проверку на дублирование значений элементов).

4. Для обработки всех ошибочных ситуаций использовать конструкцию try…catch().

4. Дополнительно к контейнеру рекомендуется реализовать класс-итератор.

5. В Main создать три экземпляра шаблонного класса-контейнера для разных типов данных. Работа с этими объектами должна демонстрироваться на следующих операциях: добавить – просмотреть – найти – удалить – найти – просмотреть.

6. Отладить и выполнить полученную программу. Проверить обработку исключительных ситуаций (например, чтение из пустого стека, дублирование объектов и т.п.).

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть лабораторной работы.

2. Реализовать индивидуальное задание согласно варианту, сделать скриншоты работающих программ. Написать комментарии.

3. Написать отчет, содержащий:

1. Титульный лист, на котором указывается:

а) полное наименование министерства образование и название учебного заведения;

б) название дисциплины;

в) номер практического занятия;

г) фамилия преподавателя, ведущего занятие;

д) фамилия, имя и номер группы студента;

е) год выполнения лабораторной работы.

2. Индивидуальное задание (листинг кода программы, скриншоты консоли с результатами работы).

3. Вывод о проделанной работе.