

Обобщенное программирование

Обобщенное программирование

- парадигма программирования, заключающаяся в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание.

Основана на использовании абстрактных описаний требований (расширении понятия **абстрактного типа данных**)

Обобщенное программирование

- Вместо описания отдельного типа применяется описание семейства типов, имеющих общий интерфейс и семантическое поведение. Набор требований, описывающий интерфейс и семантическое поведение, называется *концепцией*
- Тип *моделирует* концепцию (является моделью концепции), если он удовлетворяет её требованиям

Этапы решения задачи

1. Найти полезный и эффективный алгоритм
2. Определить обобщённое представление (параметризовать алгоритм, минимизировав требования к обрабатываемым данным)
3. Описать набор (минимальных) требований, удовлетворяя которые всё ещё можно получить эффективные алгоритмы
4. Создать каркас на основе классифицированных требований

Реализация в языках программирования

C++	Шаблоны (template)
Java	Обобщения (дженерики, generics)
.NET (в т.ч. C#)	Обобщения
Object Pascal, Delphi	Обобщенные классы

Обобщения в C#

Альтернативные варианты

- Различные варианты одного и того же алгоритма (перегрузка методов)
- Использование базового класса (в пределе – **object**) с явным приведением типов
- Обобщения: единое решение, не зависящее от конкретного типа данных, применяемое к обработке данных разных типов

Понятие обобщения

- Обобщение – параметризированный тип
- Обобщения позволяют создавать классы, структуры, интерфейсы, методы и делегаты для обработки **разнотипных данных** с соблюдением **типовой безопасности**
- Класс, структура, интерфейс, метод или делегат, оперирующий параметризированным типом данных, называется *обобщенным*

Простой пример

```
class Gen<T> {  
    T ob;  
    public Gen(T o) { ob = o; }  
    public T GetOb() { return ob; }  
    public void ShowType() { Console.WriteLine("Тип T:" +  
        typeof(T)); }  
}  
  
...  
static void Main() {  
    Gen<int> iOb = new Gen<int>(102);  
    iOb.ShowType();  
    int v = iOb.GetOb(); Console.WriteLine("Значение: " + v);  
    Gen<string> strOb = new Gen<string>("Это строка");  
    strOb.ShowType();  
    string s = strOb.GetOb(); Console.WriteLine("Значение: " + s);  
}
```

Необобщенный аналог

```
class NonGen {
    object ob;
    public NonGen(object o) { ob = o; }
    public object GetOb() { return ob; }
    public void ShowType() { Console.WriteLine("Тип T:" +
        ob.GetType()); }
}

...

static void Main() {
    NonGen iOb = new NonGen(102);
    iOb.ShowType();
    int v = (int) iOb.GetOb(); Console.WriteLine("Значение: " + v);
    NonGen strOb = new NonGen("Это строка");
    strOb.ShowType();
    string s = (string) strOb.GetOb(); Console.WriteLine("Значение:
        " + s);}
    iOb = strOb; //логическая ошибка!
```

Терминология

- **Gen**<int> – закрыто сконструированный тип
- **Gen**<T> – открыто сконструированный тип
- **int** – закрытый тип
- **T** – открытый тип (включает параметр)
- Если все аргументы типа – закрытые, то такой тип – закрыто сконструированный; если хотя бы 1 открытый – открыто сконструированный

Общий синтаксис

Объявление обобщенного класса:

`class`

`имя_класса`<`список_параметров_типа`> {

Объявление объекта (ссылки на обобщенный класс):

`имя_класса`<`список_аргументов_типа`>

`имя_переменной` = `new`

`имя_класса`<`список_аргументов_типа`>

(`аргументы_конструктора`);

Ограниченные типы: ограничение на базовый класс

```
class A {  
public void Hello () { Console.WriteLine("Hello"); } }  
class B : A { }  
class C { }  
class Test<T> where T : A  
{ T obj ;  
public Test(T o) { obj = o; }  
public void SayHello() { obj.Hello (); }  
}
```

...

```
static void Main() {  
A a = new A(); B b = new B(); C c = new C();  
Test<A> t1 = new Test<A>(a); t1.SayHello();  
Test<B> t2 = new Test<B>(b); t2.SayHello();  
// Test<C> t3 = new Test<C>(c); t3.SayHello(); // Ошибка!  
}
```

Ограниченные типы: **другие ограничения**

1. Ограничение на интерфейс

```
class имя<T> where T : имя_интерфейса
```

2. Ограничение на конструктор new() – позволяет получать экземпляр объекта обобщенного типа

```
class имя<T> where T : new()
```

3. Ограничения ссылочного типа и типа значения

```
class имя<T> where T : struct
```

```
class имя<T> where T : class
```

Абстрактные типы данных

- это тип данных, который предоставляет для работы с элементами этого типа определённый набор функций, а также возможность создавать элементы этого типа при помощи специальных функций
 - Список
 - Стек
 - Очередь
 - Ассоциативный массив

Обобщенные коллекции в C#

- Dictionary<Tkey, TValue>
- HashSet<T>
- LinkedList<T>
- List<T>
- Queue<T>
- SortedSet<T>
- Stack<T>
- ...

Пример использования

```
class Stud
{
    public string name;
    public int mark;
    public Stud(string n, int m) { name = n; mark = m; }
}
class Program
{
    static void Main()
    {
        List<Stud> l = new List<Stud>();
        l.Add(new Stud("A", 1)); l.Add(new Stud("B", 9));
        l.Add(new Stud("C", 5));
        l.Sort((s1, s2) => s1.mark - s2.mark);
        foreach (Stud s in l) Console.WriteLine(s.name);
    }
}
```