Основы программирования

Особые классы

Структуры

Определение

Структура – это набор разнотипных данных, имеющий одно имя и позволяющий доступ через это имя к составным частям структуры.

Структура очень похожа на класс.

Наряду с классами структуры представляют еще один способ создания собственных типов данных в C#.

Использование структур

Как правило, типы структуры используются для проектирования небольших ориентированных на данные типов, которые предоставляют минимум поведения или не предоставляют его вовсе.

Например, платформа .NET использует типы структуры для представления числа (как целого, так и вещественного), логического значения, символа Юникода, экземпляра времени.

Если для создаваемого типа более значимым является его поведении этого типа, рекомендуется определить класс.

Синтаксис

```
<атрибуты> <спецификаторы> struct <Имя> :
                                        <Интерфейсы>
{
   <тело структуры>;
}
```

Члены структур

- поля,
- методы,
- конструкторы,
- индексаторы,
- свойства,
- операторные методы,
- события.

Конструкторы

- Допускается определять конструкторы, но не деструкторы.
- Нельзя определить конструктор, используемый по умолчанию (т.е. конструктор без параметров). (определяется для всех структур автоматически и не подлежит изменению).
- Структуры не поддерживают наследование, Следовательно, их члены нельзя указывать как `abstract`, `virtual` или `protected`.

Создание экземпляра структуры

- Как и объект класса с помощью оператора `new` (вызывается конструктор, используемый по умолчанию).
- Не используя оператор `new` (инициализацию любых членов структуры придется выполнить вручную).

Пример. Задача

- Создадим структуру, которая описывает студента.
- Для сокращения кода структуры объявим ее поля публичными, то есть опустим свойства.
  Хотя, более правильно сделать поля закрытыми и описать для доступа к ним открытые свойства.

Пример. Структура

```
struct Student                                  {
  public int id;
  public string surname;

  public Student(int num, string fam)  {
    id = num;
    surname = fam;                              }

  public override string ToString()    {
    return ”Номер зачетки: ”+ id + ”, фамилия: ” +
  surname;                                      }
}
```

Пример. Проверка работы

```
…
Student Vasya = new Student(1, "Иванов");
Student Kolya = new Student(2, "Петров");

Student Petya=Vasya;
Petya.id = 3;

Console.WriteLine("Студенты:\n" + Vasya + "\n"
  + Kolya + "\n" + Petya);
…
```

Пример. Результат работы

```
Студенты:
Номер зачетки: 1, фамилия: Иванов
Номер зачетки: 2, фамилия: Петров
Номер зачетки: 3, фамилия: Иванов
```

Пример. Пояснения

- При присваивании структуры создается ее копия в стеке, поэтому ссылки `Petya` и `Vasya` ссылаются на РАЗНЫЕ объекты.
- При передаче в методы структуры также передаются по значению. Их можно передать в метод с помощью ключевых слов `ref` и `out`.
- Работа со структурами в стэке производится быстрее, чем работа с объектами в динамической памяти.

Пример. Вопрос

```
struct Student                          {
  public int id;
  public string surname;                }

static void Main() {
  Student sasha;
  stud1= sasha.surname
  // а можно так:
  // sasha.surname = "Александров";
  // sasha.id = 18;                     …
}
```

Особенности

- структура -- значимый тип, хранится в стэке и обрабатывается по правилам для значащих типов;
- структуры не участвуют в наследовании -- не могут порождать потомков, и сами являются прямыми наследниками класса `Object`;
- структуры могут реализовывать интерфейсы;
- в структуре могут быть описаны все элементы, присущие классам, кроме деструктора и конструктора без параметров;

Особенности

- структуры не могут иметь спецификатор `abstract`;
- спецификатор доступа `protected` не используется;
- как правило структуры имеют спецификатор `public`, спецификаторы `internal` и `private` используются для вложенных структур;
- для элементов структур также не используются спецификаторы `protected` и `protected internal`;

Особенности

- для полей структуры нельзя задавать значения по умолчанию -- а вот статическим полям значения по умолчанию задать можно;
- методы структур не могут быть помечены спецификаторами `abstract` или `virtual`, однако структуры, являясь потомком класса `Object`, могут переопределять его виртуальные методы с использованием спецификатора `override`.

Для чего нужны структуры?

- Типы классов являются ссылочными типами. То есть переменная типа класса содержит ссылку на экземпляр этого типа, а не сам экземпляр.
- Типы структур являются значимыми типами.
- Иногда полезно иметь прямой доступ к объектам как к значениям простых типов, например, ради повышения эффективности программы.
  Так как каждый доступ к объектам по ссылке связан с дополнительными издержками на расход вычислительных ресурсов и оперативной памяти.

Для чего все это?

Для повышении эффективности и производительности программ:

- для работы со структурой вообще не требуется переменная ссылочного типа;
- работа со структурой не приводит к ухудшению производительности, так как доступ к структуре осуществляется непосредственно.

Рекомендация: если нужно просто сохранить группу связанных вместе данных, не требующих наследования и обращения по ссылке, то с точки зрения производительности для них лучше выбрать структуру.

Полезные ссылки

C# и .NET | Структуры (metanit.com)

Структуры в языке C# — C# ~ Си шарп для начинающих (c-sharp.pro)

Структуры C# (programming-lessons.xyz)

Структуры в Си-шарп (mycsharp.ru)