[Введение в программирование на C# 2.0](http://www.intuit.ru/department/pl/csharp20/class/free/status/" \l "lecture_num1)

Программа – правильно построенная (не вызывающая возражений со стороны C#-компилятора) последовательность предложений, на основе которой формируется сборка.

В общем случае, программист создает файл, содержащий объявления классов, который подается на вход компилятору. Результат компиляции представляется транслятором в виде сборки. В принципе сборка может быть двух видов (здесь все зависит от замысла разработчика кода):

* Portable Executable File (PE-файл с расширением .exe), пригоден к непосредственному исполнению CLR.
* Dynamic Link Library File (DLL-файл с расширением .dll), предназначен для повторного использования как компонент в составе какого-либо приложения.

В любом случае на основе входного кода транслятор строит модуль на IL, манифест, и формирует сборку. В дальнейшем сборка либо может быть выполнена после JIT-компиляции, либо может быть использована в составе других программ.

**Пространство имен**

.NET Framework располагает большим набором полезных функций. Каждая из них является членом какого-либо класса. Классы группируются по пространствам имен. Это означает, что в общем случае имя класса может иметь сложную структуру — состоять из последовательности имен, разделенных между собой точками. Последнее имя в этой последовательности собственно и является именем класса. Классы, имена которых различаются лишь последними членами (собственно именами классов) последовательностей, считаются принадлежащими одному пространству имен.

Средством "навигации" по пространствам имен, а точнее, средством, которое позволяет сокращать имена классов, является оператор

using <ИмяПространстваИмен>;

В приложении может объявляться собственное пространство имен, а также могут использоваться ранее объявленные пространства.

В процессе построения сборки транслятор должен знать расположение сборок с заявленными для использования пространствами имен. Расположение части сборок известно изначально. Расположение всех остальных требуемых сборок указывается явно (непосредственно в Visual Studio при работе над проектом открыть окно Solution Explorer, выбрать пункт References, далее Add Reference... – там надо задать или выбрать соответствующий .DLL- или .EXE-файл).

В частности, сборка, которая содержит классы, сгруппированные в пространстве имен System, располагается в файле mscorlib.dll.

Наиболее часто используемое пространство имен – System. Расположение соответствующей сборки известно. Если не использовать оператор

using System;

корректное обращение к функции WriteLine(...) – члену класса Console выглядело бы следующим образом:

System.Console.WriteLine("Ha-Ha-Ha!"); // Полное квалифицированное

//имя функции – члена класса Console, отвечающей за вывод строки в окно приложения.

При компиляции модуля транслятор по полному имени функции (если используется оператор using – то по восстановленному на его основе) находит ее код, который и используется при выполнении сборки.

**Класс и Структура. Первое приближение**

Классы и структуры являются программно определяемыми типами, которые позволяют определять (создавать) новые типы, специально приспособленные для решения конкретных задач. В рамках объявления класса и структуры описывается множество переменных различных типов (набор данных — членов класса), правила порождения объектов — представителей структур и классов, их основные свойства и методы.

В программе класс объявляется с помощью специальной синтаксической конструкции, которая называется объявлением класса. Фактически, объявление структур и классов является основным элементом любой C# программы. В программе нет ничего, кроме объявлений и конструкций, облегчающих процедуру объявления.

С точки зрения синтаксиса, между объявлениями классов и структур существуют незначительные различия (ключевые слова struct и class, в структуре не допускается объявлений членов класса со спецификаторами доступа protected и protected internal, при объявлении структуры не допускается объявление конструктора без параметров), часть из которых будет рассмотрены далее.

Основное их различие состоит в том, что класс и структура принадлежат к двум различным категориям типов – типов-ссылок и типов-значений.

В этом разделе обсуждаются основные правила объявления классов.

Объявление класса состоит из нескольких элементов:

* Объявление атрибутов – необязательный элемент объявления.
* Модификаторы (в том числе модификаторы прав доступа) – необязательный элемент объявления.
* Partial (спецификатор разделения объявления класса) – необязательный элемент объявления.
* Class (struct для структуры).
* Имя класса.
* Имена предков (класса и интерфейсов) – необязательный элемент объявления.
* Тело класса (структуры).

Атрибуты – средство добавления ДЕКЛАРАТИВНОЙ (вспомогательной) информации к элементам программного кода. Назначение атрибутов: организация взаимодействия между программными модулями, дополнительная информация об условиях выполнения кода, управление сериализацией (правила сохранения информации), отладка и многое другое.

Модификаторы new, abstract, sealed, static обсуждаются дальше. Модификаторы прав доступа обеспечивают реализацию принципа инкапсуляции, используются при объявлении классов, структур и их составляющих компонентов. Представлены следующими значениями:

|  |  |
| --- | --- |
| public | Обозначение для общедоступных членов класса. К ним можно обратиться из любого метода любого класса программы |
| protected | Обозначение для членов класса, доступных в рамках объявляемого класса и из методов производных классов |
| internal | Обозначение для членов класса, которые доступны из методов классов, объявляемых в рамках сборки, где содержится объявление данного класса |
| protected internal | Обозначение для членов класса, доступных в рамках объявляемого класса, из методов производных классов, а также доступных из методов классов, которые объявлены в рамках сборки, содержащей объявление данного класса |
| private | Обозначение для членов класса, доступных в рамках объявляемого класса |

Спецификатор разделения объявления класса partial позволяет разбивать код объявления класса на несколько частей, каждая из которых размещается в собственном файле. Если объявление класса занимает большое количество строк, его размещение по нескольким файлам может существенно облегчить работу над программным кодом, его документирование и модификацию. Транслятор способен восстановить полное объявление класса. Спецификатор partial может быть использован при объявлении классов, структур и интерфейсов.

Сочетание ключевого слова class ( struct, interface ) и имени объявляемого класса (структуры или интерфейса) задает имя типа.

Конструкции

:имя класса

(при объявлении класса)

:список имен интерфейсов

(при объявлении структуры или класса)

:имя класса, список имен интерфейсов

(при объявлении класса)

с обязательным разделителем ' :' обеспечивают реализацию принципа наследования и будут обсуждаться позже.

Тело класса в объявлении ограничивается парой разделителей ' { ', ' } ', между которыми располагаются объявления данных — членов и методов класса.

Следующий пример демонстрирует использование основных элементов объявления структуры. При объявлении структуры допускается лишь один явный спецификатор – public (здесь он опущен). Отсутствие спецификаторов доступа в объявлениях членов структуры (класса) эквивалентно явному указанию спецификаторов private.

// Указание на используемые пространства имен.

using System;

using System.Drawing;

namespace qwe // Объявление собственного пространства имен. Начало.

{

// Начало объявления структуры

struct S1

{// Тело структуры – НАЧАЛО

// Объявление данных-членов.

private Point p;

// protected int qwe; // Спецификатор protected в объявлении членов

// структуры недопустим. Структура не имеет развитого механизма

//наследования.

// Структура не может иметь конструктора без параметров.

public S1(int x, int y)

{

p = new Point(10,10);

}

// Объявление методов.

// Статический метод. Точка входа.

static void Main(string[] args)

{

// Тело метода. Здесь обычно располагается программный код,

// определяющий функциональность класса.

}

}// Тело структуры – КОНЕЦ

} // Объявление собственного пространства имен. Конец.