Основные понятия языка программирования C#

Комментарий к слайду

В данной лекции будут рассмотрены вопросы, относящиеся к понятийному аппарату, истории развития, выразительным возможностям синтаксиса и особенностям реализации языка программирования C# в сравнении с другими объектно-ориентированными языками программирования.

Содержание лекции

Комментарий к слайду

Коротко о содержании лекции.

Прежде всего, необходимо напомнить слушателям динамику эволюционного процесса, который привел к появлению языка программирования C#.

Далее, естественно сосредоточиться на функциональных особенностях и расширенных (по сравнению с традиционными языками ООП) возможностях этого языка программирования. Таким образом, будет определено место языка программирования C# в семействе подобных ему языков и общей классификации.

Отметим, что исследование языка программирования C# будет производиться в сравнении с аналогами, прежде всего, с такими хорошо известными языками программирования, как Java и Visual Basic. Первый представляет собой потенциально независимое от программной платформы (и операционной системы) средство создания программного обеспечения, а второй – относительно простой в освоении объектно-ориентированный язык программирования производства компании Microsoft.

Исследование основных выразительных возможностей синтаксиса языка программирования C# будет иллюстрироваться примерами фрагментов программ с необходимыми комментариями. Особое внимание будет уделено общей структуре программы на языке C#.

Сравнительный анализ языка C# в семействе современных языков ООП завершится выводами относительно преимуществ и недостатков принципов реализации и синтаксиса языка.

Наконец, для желающих глубже исследовать предмет будут представлены ссылки на важнейшие работы теоретического и практического плана по теме лекции.

История развития языка программирования C#

Язык B: 1963, К.Томпсон (Ken Thompson, MIT)

- Подобен языку PL/I, однако менее громоздок, применялся для разработки операционной системы UNIX

Язык C: 1972, Д.Ритчи, К.Томпсон (Dennis Ritchie & Ken Thompson, AT&T Bell Telephone Laboratories)

- Язык B, расширенный типами, структурами и новыми операциями

Язык C++: 1984, Б.Страуструп (Bjarne Stroustrup , Bell Labs)

- Введены классы как объекты данных
- Название C++ предложил Р.Маскитти (Rics Mascitti, Bell Labs)

Язык C#:2000, Microsoft

- «Всякая сущность есть объект»; компонентный; безопасный

Комментарий к слайду

Прежде, чем перейти непосредственно к исследованию конструктивных особенностей языка программирования C#, рассмотрим историю его развития.

Исторические корни возникновения основной ветви языков программирования, которая привела к появлению C#, ведут к 60-м г.г., а именно, ко времени возникновения языка B. Последний является типичным представителем ранних императивных языков программирования. Язык B был придуман в 1963 году творческим коллективом, основным создателем языка принято считать К. Томпсона из Технологического университета Массачусетса (Ken Thompson, MIT). Основной целью разработки языка была реализация операционной системы UNIX. Уже существовавший язык PL/I, применявшийся в то время для мейнфреймов производства компании IBM, был достаточно громоздким и менее подходил для поставленной задачи, чем новое, оригинальное решение ученых-практиков.

Следующим шагом в «алфавите» языков программирования, ведущем к языку C#, стал язык C, который был изобретен на основе языка B в 1972 году. Авторами нового языка программирования стали уже известный нам Кен Томпсон, а также Д. Ритчи (Dennis Ritchie), которые работали в исследовательской лаборатории компании AT&T (AT&T Bell Telephone Laboratories). В варианте C язык B расширился за счет явного использования типов (напомним, что языки и теории с типами имеют существенные преимущества перед бестиповыми аналогами), структур и ряда новых операций.

Дальнейшее развитие языка происходило в стенах той же организации. И снова примерно через 10 лет, в 1984 году, Б. Страуструп (Bjarne Stroustrup, Bell Labs) выступил с проектом языка C++ - ООП-расширения языка C, в котором вводится понятие класса как объекта данных. Заметим, что название C++ для нового языка предложил Р.Маскитти (Rics Mascitti, Bell Labs), представляющий ту же организацию.

Наконец, уже в 2000 году, более чем через 15 лет, корпорация Microsoft выпустила в свет C++ нового поколения под названием C# («Си шарп»), основным постулатом которого является высказывание: «всякая сущность есть объект». Язык основан на строгой

компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

Основные возможности C#

Подобен языкам Java, C++ и VB, однако является компонентно-ориентированным и более безопасным

Добавлен ряд новых черт (делегаты, индексаторы, механизм (un)boxing и др.)

Сходство с Java:

- объектно-ориентированный (единственное наследование)
- интерфейсы
- исключения
- нити (threads)
- пространства имен
- сильная (строгая) типизация
- сборка мусора
- отражение (reflection)
- динамическая загрузка кода

Сходство с C++:

- «перегруженные» операторы
- арифметические операции с плавающей точкой относятся к небезопасному коду
- некоторые особенности синтаксиса

Комментарий к слайду

Как уже отмечалось в ходе лекции, язык программирования C# вобрал лучшие черты целого ряда своих предшественников. Кроме упомянутой ранее ветви языков B-C-C++, необходимо указать еще несколько знаковых для настоящего времени языков программирования, а именно, Java и Visual Basic.

Несмотря на весьма существенные различия между компонентной объектной моделью COM (основного стандарта Microsoft для компонентного проектирования и реализации программного обеспечения) и моделью Java Beans, базовым стандарт Sun Microsystems для компонент (зависимой от языка реализации), язык программирования C# имеет довольно много общего с языком Java. Естественно, немало черт язык программирования C# унаследовал и от своего предшественника, созданного корпорацией Microsoft, языка Visual Basic.

Как уже отмечалось, язык программирования C# основан на строгой компонентной архитектуре и реализует передовые механизмы обеспечения безопасности кода.

Перечислим наиболее характерные черты сходства языков программирования C# и Java. Прежде всего, оба эти языка относятся к категории объектно-ориентированных и предполагают единственность наследования. Другими важными особенностями, которые сближают языки программирования C# и Java, являются механизмы интерфейсов, обработки исключительных ситуаций, нитей (threads). Сборка мусора и пространства имен реализованы в этих двух языках сходным образом. Оба языка программирования характеризуются сильной (строгой) типизацией и динамической загрузкой кода при выполнении программы.

От своего прямого предшественника, языка программирования C++, языком C# унаследованы следующие механизмы: «перегруженные» операторы, небезопасные арифметические операции с плавающей точкой, а также ряд других особенностей синтаксиса.

Краткий список основных возможностей C#

- Компонентно-ориентированное программирование
  - Свойства
  - События
- Унифицированная система типизации (UTS)
- Делегаты (Delegates)
- Индексаторы (Indexers)
- Перегруженные операторы
- Оператор `foreach`
- Механизмы boxing и unboxing
- Атрибуты
- Прямоугольные массивы

Комментарий к слайду

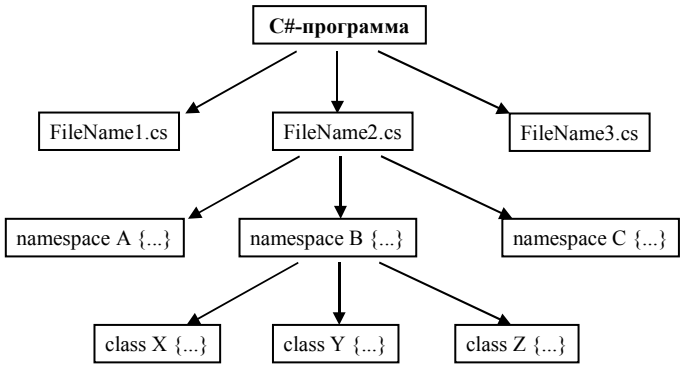
Несмотря на то, что целый ряд конструктивных синтаксических механизмов и особенностей реализации унаследован языком программирования C# от прародителей (C++, Visual Basic и Java), возможности этого нового языка программирования не ограничиваются суммой возможностей его исторических предшественников. На данном этапе ограничимся перечислением основных возможностей, которыми характеризуется язык программирования C#.

К числу принципиально важных решений, которые реализованы корпорацией Microsoft в языке программирования C#, можно отнести следующие:

- компонентно-ориентированный подход к программированию (который характерен и для идеологии Microsoft .NET в целом);
- свойства как средство инкапсуляции данных (характерно также в целом для ООП);
- обработка событий (имеются расширения, в том числе в части обработки исключений, в частности, оператор try);
- унифицированная система типизации (соответствует идеологии Microsoft .NET в целом);
- делегаты (delegate – развитие указателя на функцию в языках C и C++);
- индексаторы (indexer – операторы индекса для обращения к элементам класса-контейнера);
- перегруженные операторы (развитие ООП);
- оператор foreach (обработка всех элементов классов-коллекций, аналог Visual Basic);
- механизмы boxing и unboxing для преобразования типов;
- атрибуты (средство оперирования метаданными в COM-модели);
- прямоугольные массивы (набор элементов с доступом по номеру индекса и одинаковым количеством столбцов и строк).

Особенности языка программирования C#, которые в большей степени отвечают целям настоящего учебного курса, будут рассмотрены более подробно в рамках настоящей лекции, другие – в течение последующих лекций.

Структура программы на языке C#



Комментарий к слайду

Прежде всего, рассмотрим обобщенную структуру программы на языке программирования C#. Проиллюстрируем структуру программы показательным примером (см. схему).

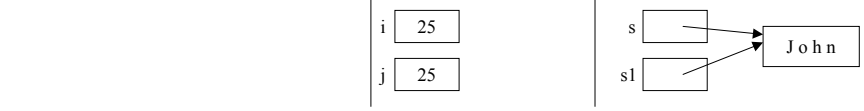
Заметим, что программа на C# может состоять как из одного, так и из нескольких файлов, содержащих исходный текст на языке программирования C#. Каждый такой файл имеет расширение .CS (в нашем примере файлы названы FileName1.cs, FileName2.cs и FileName1.cs).

Любой файл с исходным текстом на языке программирования C# может как содержать пространства имен, так и не содержать их (в нашем примере файл названы FileName2.cs содержит три пространства имен (A, B и C), а FileName1.cs и FileName3.cs не содержат пространств имен).

Наконец, каждое пространство имен может как содержать описание (одного или нескольких) классов, так и не содержать их (в нашем примере пространство имен B содержит три описания трех классов (X, Y и Z), а пространство имен A и C не содержат ни одного описания классов).

Сопоставление ссылочных типов и типов-значений

	Типы-значения	Ссылочные типы
Переменная содержит	значение	ссылку на значение
Переменная хранится	в стеке	в куче
Значение по умолчанию	0, false, '\0'	null
Оператор присваивания	копирует значение	копирует ссылку
Пример	int i = 25; int j = i;	string s = “John"; string s1 = s;



Комментарий к слайду

Ранее в ходе изложения настоящего курса уже обсуждались механизмы boxing и unboxing, реализованные в различных средах разработки программного обеспечения Microsoft .NET. Напомним, что эти механизмы используются для преобразования типов выражений языков программирования из ссылочных в типы-значения и обратно.

Рассмотрим более подробно реализацию двух основных семейств типов данных, а именно, ссылочных типов и типов-значений, применительно к языку программирования C#. Для определенности рассмотрим случай одного из простейших объектов языка программирования C#, а именно, переменной.

В соответствии с названиями, переменная в случае использования типов-значений содержит собственно значение, а при использовании ссылочных типов – не само значение, а лишь ссылку (указатель) на него.

Местом хранения переменной, определенной как тип-значение, является стек, а определенной как ссылочный тип – «куча» (последнее необходимо для динамического выделения и освобождения памяти для хранения переменной произвольным образом).

Значением, которым переменная инициализируется по умолчанию (необходимость выполнения этого требования диктуется идеологией безопасности Microsoft .NET) в случае определения посредством типа-значения является 0 (для целого или вещественного типа данных), false (для логического типа данных), '\0' (для строкового типа данных), а в случае определения посредством ссылочного типа – значение пустой ссылки null.

При выполнении оператора присваивания в случае переменной-значения копируется значение, а в случае переменной-ссылки – ссылка.

Приведенный далее пример иллюстрирует различия в реализации типов-ссылок и значений.

Современные языки программирования и .NET: II семестр
Лекция 3: Основные понятия языка программирования C#

Элементарные типы языка C# (в сопоставлении с языком SML)

C#	CTS	SML	Диапазон
sbyte	System.SByte	---	-128 .. 127
byte	System.Byte	byte	0 .. 255
short	System.Int16	int	-32768 .. 32767
ushort	System.UInt16	word	0 .. 65535
long	System.Int64	---	-2^{63} .. $2^{63}-1$
float	System.Single	real	±1.5E-45 .. ±3.4E38 (32 Bit)
double	System.Double	---	±5E-324 .. ±1.7E308 (64 Bit)
decimal	System.Decimal	---	±1E-28 .. ±7.9E28 (128 Bit)
bool	System.Boolean	bool	true, false
char	System.Char	char	Символ (в коде unicode)

Комментарий к слайду

Перейдем к более подробному рассмотрению основных типов, входящих в состав Common Type System (CTS) Microsoft .NET. При этом прикладное исследование языков программирования произведем в форме сопоставления отображений фрагментов систем типизации языков программирования SML и C# в систему типов CTS.

При этом для обеспечения большей наглядности сопоставления сведем отображения в следующую таблицу:

C#	CTS	SML	Диапазон
sbyte	System.SByte	---	-128 .. 127
byte	System.Byte	byte	0 .. 255
short	System.Int16	int	-32768 .. 32767
ushort	System.UInt16	word	0 .. 65535
long	System.Int64	---	-263 .. 263-1
float	System.Single	real	±1.5E-45 .. ±3.4E38 (32 Bit)
double	System.Double	---	±5E-324 .. ±1.7E308 (64 Bit)
decimal	System.Decimal	---	±1E-28 .. ±7.9E28 (128 Bit)
bool	System.Boolean	bool	true, false
char	System.Char	char	Символ (в коде unicode)

Даже из предварительного анализа таблицы видно, что система типизации языка программирования C# значительно богаче по сравнению с языком программирования SML. Можно заметить, что всякому типу языка программирования SML соответствует некоторый тип языка программирования C#, и их названия зачастую совпадают или являются подобными друг другу.

Наконец, отметим, что все без исключения типы обоих языков программирования однозначно отображаются в систему типизации Microsoft .NET, верхним иерархическим элементом которой является пространство имен System.

Оператор перечисления

Список поименованных констант

Описание (непосредственно в пространстве имен):

```
enum Color {red, blue, green}  // значения: 0, 1, 2
enum Access {personal=1, group=2, all=4}
enum Access1 : byte {personal=1, group=2, all=4}
```

Применение:

```
Color c = Color.blue;
    // для перечислимых констант должно быть указано полное имя
Access a = Access.personal | Access.group;
if ((Access.personal & a) != 0)
Console.WriteLine("access granted");
```


Комментарий к слайду

Кроме понятия переменной, которое уже было исследовано ранее в связи с типами-значениями и ссылочными типами, интересно провести исследование объектов-констант в языке программирования C#.

Напомним, что константные объекты в комбинаторной логике можно моделировать посредством комбинатора-канцелятора K с характеристикой K x y = x.

Рассмотрим ряд примеров описания констант посредством оператора перечисления (заметим, что применяемый в C# способ описания констант изначально ориентирован на написание безопасного кода, поскольку содержит обязательную инициализацию).

Оператор перечисления `enum` языка C# представляет собой список поименованных констант. Описание производится непосредственно в пространстве имен с областью видимости, которую необходимо обеспечить для константы:

```
enum Color {red, blue, green}
enum Access {personal=1, group=2, all=4}
enum Access1 : byte {personal=1, group=2, all=4}
```

Пример использования константы после ее описания имеет следующий вид:

```
Color c = Color.blue;
```

Заметим, что для перечислимых констант должно быть указано полное имя:

```
Access a = Access.personal | Access.group;
if((Access.personal & a) != 0)
      Console.WriteLine("access granted");
```

Операторы typeof и sizeof в языке C#

typeof

- Возвращает для данного типа дескриптор типа
  Дескриптор типа для объекта o возвращается операцией o.GetType().

```
Type t = typeof(int);
Console.WriteLine(t.Name);   // ➔ тип данного выражения: Int32
```

sizeof

- Возвращает размер элемента данного типа (в байтах).
- Применим только к типам-значениям.
- Используется только в блоках небезопасного кода (размер структур может изменяться в зависимости от среды реализации).
  Необходима компиляция посредством команды csc /unsafe xxx.cs

```
unsafe {
  Console.WriteLine(sizeof(int));
  Console.WriteLine(sizeof(MyEnumType));
  Console.WriteLine(sizeof(MyStructType));
}
```


Комментарий к слайду

Подобно языку программирования SML в языке C# реализована возможность определения типа для того или иного языкового объекта. Для реализации этой возможности используется оператор `typeof`, который для любого заданного типа возвращает дескриптор этого типа.

Заметим, что дескриптор типа для объекта `o` возвращается операцией `o.GetType()`. Рассмотрим пример, иллюстрирующий использование оператора `typeof`:

```
Type t = typeof(int);
Console.WriteLine(t.Name);
```

При выполнении данного примера тип рассматриваемого выражения будет определен системой как Int32, что вполне соответствует данным из сравнительной таблицы отображения типов языковых объектов C# и SML в систему типизации CTS Microsoft .NET.

Еще одной практически важной конструкцией C#, оперирующей типами, является оператор `sizeof`, который возвращает размер элемента данного типа (в байтах). Данный оператор применим только к типам-значениям и используется лишь в блоках небезопасного кода (т.к. размер структур может изменяться в зависимости от среды реализации); необходима компиляция кода командой `csc /unsafe xxx.cs`.

Приведем пример использования оператора `sizeof` (заметим, что блоки небезопасного C#-кода выделяются посредством ключевого слова `unsafe`):

```
unsafe {
   Console.WriteLine(sizeof(int));
   Console.WriteLine(sizeof(MyEnumType));
   Console.WriteLine(sizeof(MyStructType)); }
```

Оператор struct языка C#

Описание:

```
struct Point {
  public int x, y;              // поля
  public Point (int x, int y) { this.x = x;
this.y = y; }                   // конструктор
  public void MoveTo (int a,int b) {x=a; y=b;}
                               // методы
}
```

Применение:

```
Point p = new Point(3, 4);
// конструктор инициализирует объекты стека
p.MoveTo(10, 20);              // вызов метода
```


Комментарий к слайду

Одним из важнейших видов основных объектов языка программирования C# являются структуры (аналоги структур имеются и языке SML; в языках функционального программирования с примитивной системой типизации аналогами структур могут служить списки).

Рассмотрим пример описания структуры `Point`, моделирующей точку на плоскости:

```
struct Point {
     public int x, y;
     public Point (int x, int y) { this.x = x; this.y = y; }
     public void MoveTo (int a,int b) {x=a; y=b;}
}
```

Заметим, что координаты точки `x, y` (в простейшем случае целочисленные), которые используются для определения ее положения, являются атрибутами объекта или полями структуры.

Операция `Point` является функцией инициализации объекта и называется конструктором. Заметим, что унаследованный от C++ указатель `this` является ссылкой на текущий объект.

Наконец, операция (или, иначе, метод `MoveTo`) изменяет текущее местоположение точки на пару целочисленных координат `(a,b)`.

Для использования объекта-структуры необходимо проинициализировать объект-значение в стеке посредством конструктора. Затем можно вызвать метод `MoveTo`:

```
Point p = new Point(3, 4);
p.MoveTo(10, 20);
```

Область описания

Часть программы, к которой относится описание

Объекты языка могут быть описаны:

- в пространстве имен(классы, интерфейсы, структуры, перечисления, делегаты);
- в классе, интерфейсе, структуре (поля, методы, свойства, события, индексаторы и т.д.);
- в перечислении (перечисляемые константы); в блоке(локальные переменные)

Контекстные соглашения:

- недопустимо двукратное описание в пределах данной области описания;
- последовательность описаний произвольна;
- исключение: локальные переменные необходимо описать до использования

Соглашения об областях видимости:

- идентификатор виден лишь из своей области описания;
- область видимости можно изменить посредством модификаторов (`private, protected`)

Комментарий к слайду

Мы познакомились с одним из основных видов объектов языка программирования C#, а именно, со структурами.

В предыдущем примере фрагмента программы на C# дважды фигурировал идентификатор `Point`. В связи с этим уместно вспомнить о проблеме коллизий переменных и возможных путях ее преодоления (в предыдущей части курса для решения этой задачи использовался формализм, известный под названием чисел де Брейна). В языке программирования SML переменные могут быть описаны либо глобально (это описание распространяется на весь текст программы), либо локально в теле функции (это описание распространяется на весь текст функции).

В языке программирования C#, где объекты имеют существенно более сложную структуру, вводится понятие области описания, под которой понимают фрагмент программы, к которому относится данное описание.

Объекты языка программирования C# могут быть описаны:
• в пространстве имен (классы, интерфейсы, структуры, перечисления, делегаты);
• в классе, интерфейсе, структуре (поля, методы, свойства, события, индексаторы и т.д.);
• в перечислении (перечисляемые константы);
• в блоке (локальные переменные).

При этом принимаются следующие контекстные соглашения. Во-первых, недопустимо двукратное описание в пределах данной области описания. Во-вторых, последовательность описаний является произвольной. Исключение составляют локальные переменные, которые необходимо описать до первого использования.

Кроме того, принимаются следующие соглашения об областях видимости. Во-первых, любой идентификатор виден лишь из своей области описания. Во-вторых, область видимости можно изменить посредством модификаторов `(private,protected)`.

Пространства имен

Файл X.cs

```
namespace A {
   ... class C ...
   ... interface I...
   ... struct S...
   ... enum e ...
   ... delegate d ...

  namespace B {
     // полное имя: A.B
      ...
  }
}
```

Файл Y.cs

```
namespace A {
   ...
  namespace B {...}
  namespace C {...}
}
```

- Пространства имен из разных файлов, имеющие один и тот же идентификатор, составляют единую область описания.
- Вложенные пространства имен составляют собственную область описания.

Комментарий к слайду

В процессе изучения структуры программы на языке C# неоднократно употреблялся термин «пространство имен». В силу существования более значимых понятий объектно-ориентированного подхода к программированию в целом и языка программирования C# в частности приводилось лишь общее обсуждение данного термина. Рассмотрим пространства имен языка программирования C# более подробно.

Рассмотрим два файла X.cs и Y.cs, содержащих исходные тексты программ на языке C#.

Файл X.cs

```
namespace A {
    ... class C ...
    ... interface I...
    ... struct S...
    ... enum e ...
    ... delegate d ...
  namespace B {
          // полное имя: A.B
       ...
  }
}
```

Файл Y.cs

```
namespace A {
    ...
  namespace B {...}
  namespace C {...}
}
```

Заметим, что в файле X.cs содержатся описания пространств имен `A` и `B`, а в файле Y.cs – `A`, `B` и `C`, причем в обоих случаях последующие пространства имен вложены в `A`. При обращении к вложенному пространству имен нужно указывать его полное имя, например, `A.B`.

Необходимо отметить, что пространства имен из разных файлов, имеющие один и тот же идентификатор, составляют единую область описания. Заметим также, что вложенные пространства имен составляют собственную (отдельную) область описания.

Оператор блока в C#

В C# существуют различные виды блоков:

```
void foo (int x) {     // блок методов
      ... локальные переменные ...
   {                   // вложенный блок
      ... локальные переменные ...
   }

   for (int i = 0; ...) {
     // блок структурированных операторов
      ... локальные переменные ...
   }
}
```


Комментарий к слайду

Несмотря на то, что язык программирования C# реализует объектно-ориентированный подход к программированию, целый ряд характерных особенностей унаследован им еще от ранних процедурных прародителей, прежде всего, языка C.

Одной из черт, унаследованных C# от языка программирования C, является блок операторов – последовательность операторов разделенных символами «`;`», ограниченная фигурными скобками «`{`» и «`}`».

Аналогом оператора блока в языках функционального программирования является функция.

В язык программирования C# выделяют различные виды блоков, которые наглядно демонстрируются следующим примером:

```
void foo (int x) {
// блок методов
... локальные переменные ...
    { // вложенный блок
     ... локальные переменные ...
    }
    for (int i = 0; ...) {
    // блок структурированных операторов
     ... локальные переменные ...
    }
}
```

Как видно из примера, блоки объединяют (структурированные) операторы и методы. Допускается использование блоков, вложенных один в другой. Для удобочитаемости текста программы глубину вложенности блока лучше сохранять в зависимости от величины табуляции.

Описание локальной переменной в языке C#

```
void foo(int a) {
  int b;
  if (...) {
    int b;    // ошибка: переменная b уже описана в другом блоке
    int c;    // пока все в порядке, однако ...
    int d;
    ...
  } else {
    int a;    // ошибка: переменная a уже описана во внешнем блоке
    int d;    // конфликтов с переменной d из предыдущего блока нет
  }
  for (int i=0;...){...}
  for (int i=0;...){...} // ok: нет конфликтов с переменной i из
                         //предыдущего цикла
  int c;      // ошибка: c  уже описана в данном пространстве имен
}
```


Комментарий к слайду

Рассмотрим подробнее особенности описания локальных переменных в языке программирования C# на следующем примере программы:

```
void foo(int a) {
        int b;
        if (...) {
           int b;        // ошибка: переменная b уже описана в другом блоке
           int c;        // пока все в порядке, однако ...
           int d;
                         ...
        } else {
           int a;        // ошибка: переменная a уже описана во внешнем блоке
           int d;        // конфликтов с переменной d из предыдущего блока нет
        }
           for (int i=0;...){...}
           for (int i=0;...){...}  // все в порядке: нет конфликтов с
                                   // переменной i из предыдущего цикла
           int c;        // ошибка: c  уже описана в данном пространстве имен
}
```

Рассмотрим функцию `foo` с одним целочисленным аргументом `a`, которая не возвращает значения. Как видно из данного примера, коллизии описаний переменных возникают в случаях множественного описания внутри одного и того же блока или пространства имен.

Заметим попутно, что условный оператор `if...else` языка C# весьма схож с подобным оператором языка SML, а оператор `for` является оператором цикла.

Пример использования других пространств имен

Color.cs

```
namespace Util
{
  public enum
    Color {...}
}
```

Figures.cs

```
namespace
Util.Figures {
  public class
    Rect {...}
  public class
    Circle {...}
}
```

Triangle.cs

```
namespace
Util.Figures {
  public class
    Triangle {...}
}
```

```
using Util.Figures;
class Test {
  Rect r;            // без указания полного имени
                     //(т.к. используем Util.Figures)
  Triangle t;
  Util.Color c;    // с указанием полного имени
}
```


Комментарий к слайду

Рассмотрим более сложный пример использования пространств имен в языке программирования C#.

Пусть программный проект на языке программирования C# содержит три файла с описаниями структур данных, оформленными в виде отдельных пространств имен:

Color.cs

```
namespace Util
{
  public enum
    Color {...}
}
```

Figures.cs

```
namespace Util.Figures {
  public class
    Rect {...}
  public class
    Circle {...}
}
```

Triangle.cs

```
namespace
  Util.Figures {
   public class
     Triangle {...}
}
```

В данном случае при использовании полного имени пространства имен (`Util.Figures`) возможно обойтись без конкретизации классов (`Rect`), описанных внутри этого пространства имен. Однако, в случае обращения к классам вне данного пространства имен необходимо использовать полное квалификационное имя объекта (`Util.Color c`):

```
using Util.Figures;
class Test {
  Rect r;                  // без указания полного имени
                //(т.к. используем Util.Figures)
  Triangle t;
  Util.Color c;     // с указанием полного имени
```

Преимущества языка программирования C#

1. Подлинная объектная ориентированность (всякая языковая сущность претендует на то, чтобы быть объектом)
2. Компонентно-ориентированное программирование
3. Безопасный (по сравнению с языками C и C++) код
4. Унифицированная система типизации
5. Поддержка событийно-ориентированного программирования
6. «Родной» язык для создания приложений в среде .NET
7. Объединение лучших идей современных языков программирования: Java, C++, Visual Basic и др.

Комментарий к слайду

Проанализировав основные особенности языка программирования C#, а также исследовав структуру и принципы построения программ на этом языке, сформулируем наиболее заметные преимущества изучаемого языка программирования.

Прежде всего, необходимо отметить то обстоятельство, что язык программирования C# претендует на подлинную объектную ориентированность (а всякая языковая сущность претендует на то, чтобы быть объектом).

Кроме того, язык программирования C# призван практически реализовать компонентно-ориентированный подход к программированию, который способствует меньшей машинно-архитектурной зависимости результирующего программного кода, большей гибкости, переносимости и легкости повторного использования (фрагментов) программ.

Принципиально важным отличием от предшественников является изначальная ориентация на безопасность кода (что особенно заметно в сравнении с языками C и C++).

Унифицированная, максимально близкая по масштабу и гибкости к Common Type System, принятой в Microsoft .NET, система типизации является важным преимуществом языка C#.

Расширенная поддержка событийно-ориентированного программирования выгодно отличает язык программирования C# от целого ряда предшественников.

Язык программирования C# является «родным» для создания приложений в среде Microsoft .NET, поскольку наиболее тесно и эффективно интегрирован с ней.

Объединение лучших идей современных языков программирования (Java, C++, Visual Basic и др.) делает язык C# не просто суммой их достоинств, а языком программирования нового поколения.

Недостатки языка программирования C#

1. Довольно сложный синтаксис (75% из Java, 10% из C++, 5% из Visual Basic)
2. Относительно немного свежих концептуальных идей (вероятно, менее чем 10% конструкций языка)
3. Утверждение о чистой объектности C# спорно (проф. K. Гуткнехт (K.Gutknecht, ETH, Цюрих), предложил «активную» объектную модель для языка Zonnon)
4. Относительно невысокая производительность (~ в 100 раз медленнее, чем язык C, хотя и сравнимая с языком Java)
5. Не кросс-платформенный язык (в смысле ОС, хотя определенный прогресс в этом направлении очевиден)

Комментарий к слайду

Несмотря на значительное количество принципиальных преимуществ по сравнению с существующими аналогами, язык программирования C# не лишен и отдельных недостатков, которые, весьма вероятно, носят субъективный, локальный, временный характер.

Прежде всего, необходимо отметить то обстоятельство, что язык программирования C# имеет довольно сложный синтаксис (можно утверждать, что примерно 75% его синтаксических возможностей аналогичны языку программирования Java, 10% – подобны языку программирования C++, а 5% – заимствованы из языка программирования Visual Basic). Объем действительно свежих концептуальных идей в языке C# относительно невысок (по мнению некоторых исследователей, он, составляет около 10% от общего объема конструкций языка).

Утверждение том, что язык программирования C# является чисто объектным, допускает неоднозначную интерпретацию. Так, например, профессор K. Гуткнехт (K.Gutknecht) из института ETH (г. Цюрих, Швейцария) предложил альтернативную (так называемую «активную») объектную модель для разработанного им языка программирования Zonnon.

До настоящего времени компилятор и среда разработки программного обеспечения, поддерживающие язык C#, обладают относительно невысокой производительностью (т.е. код программы на языке C# компилируется и выполняется примерно в 100 раз медленнее, чем тот же код на языке C). Справедливости ради нужно отметить, что производительность программ на C# вполне сравнима с тем же показателем для языка Java.

На сегодня программы, написанные на языке C#, не работоспособны под управлением альтернативных операционных систем (ведутся работы по обеспечению совместимости с операционными системами Linux и FreeBSD семейства UNIX).

Библиография

1. Visual C#. NET Step by Step, Microsoft Press, 2003. ISBN: 0-7356-1909-3
2. Liberty J. Programming C#, 3d edition. O'Reilly & Associates, 2003, 710 pages. ISBN: 0596004893
3. Pratt T.W., Zelkovitz M.V. Programming languages, design and implementation (4th ed.).- Prentice Hall, 2000
4. Appleby D., VandeKopple J.J. Programming languages, paradigm and practice (2nd ed.).- McGraw-Hill, 1997
5. Gilmore S. Programming in Standard ML '97: a tutorial introduction. http:/www.dcs.ed.ac.uk/home/stg

Комментарий к слайду

К сожалению, в рамках времени, отведенных на одну лекцию, можно лишь в общих чертах охарактеризовать концепцию и возможности такого объемного и многоаспектного языка программирования как C#. Для более детального ознакомления с особенностями, достижениями и проблемами в области концептуальных инноваций, синтаксиса и расширенных возможностей языка программирования C# рекомендуется следующий список литературы:

1.Visual C#. NET Step by Step, Microsoft Press, 2003. ISBN: 0-7356-1909-3.

2.Liberty J. Programming C#, 3d edition. O'Reilly & Associates, 2003, 710 pages. ISBN: 0596004893.

3.Pratt T.W., Zelkovitz M.V. Programming languages, design and implementation (4th ed.).- Prentice Hall, 2000.

4.Appleby D., VandeKopple J.J. Programming languages, paradigm and practice (2nd ed.).- McGraw-Hill, 1997.

5.Gilmore S. Programming in Standard ML '97: a tutorial introduction. http:/www.dcs.ed.ac.uk/home/stg.

Кратко остановимся на источниках. Работа [1] является энциклопедией проектирования и реализации программного обеспечения на языке программирования C# под управлением Microsoft Visual Studio .NET. В работе [2] рассматриваются вопросы, связанные с разработкой программ на языке C#. Работы [3-4] посвящены теории и практике программирования на различных языках и в рамках различных подходов. Работа [1] содержит развернутое описание синтаксиса языка функционального программирования SML.