C#. Базовый курс

Урок 1

Введение. Базовые типы данных. Консоль. Классы и методы

Что такое .NET Framework? Консольное приложение. Выводим и вводим данные. Типы данных и их преобразование. Первое знакомство с методами и классами.

Для чего мы тут все собрались?

- Задача этого курса кратко и доступно изложить основы языка программирования C# - одного из самых перспективных современных языков программирования

- **Первоочередная** задача помочь проскочить "нулевой уровень".

Что будем изучать на курсе?

- Будет показано как использовать C# для решения классических задач программирования

- Особое внимание уделено как правильно использовать ООП, сравнивая процедурное и ООП программирование

Почему мы изучаем именно C#?

C# идеально заточен под программирование для Windows.

Но это не означает что на C# нельзя писать под Linux, ведь есть Mono.

C# язык для платформы .NET. На эту платформу переведено около 200 других языков

Для чего Вы учите C#?

Поумнеть
Стать лучше
Получить профессию
Открыть собственное дело

Для чего вы учите C#?

Программирование дает возможность к бесконечному самосовершенствованию в области ИТ

Сложно только то, что не понятно. Но если это понятно другим, то почему Вы не сможете это понять? И следовательно сделать это не сложным.

Каких результатов мы добьемся?

При успешном обучении Вы сможете писать программы на C#
Поймете что такое ООП
Получите навыки для дальнейшего обучения профессиональному программированию
При успешно обучении в итоге получится программа, которая продемонстрирует ваше владение языком C# и технологией .NET Framework

План

- .Net Framework
- Visual Studio
- Комментарии, регионы
- Структура консольного приложения
- Вывод на экран консольного приложения
- Типы данных
- Переменные и выражения
- Логический тип данных
- Булевская логика
- Преобразование типов

- Ввод данных
- Форматированный вывод на экран
- Класс. Статические методы
- Создание собственных методов.
- Перегрузка методов
- Практикум
- Выдача домашнего задания
- Дополнительно

Ключевые слова

C#, .Net Framework, CLR, Visual Studio, комментарии, класс, метод, Console, Write, WriteLine, ReadLine, Main, переменные, типы данных, преобразование типов, статический, перегрузка, пространство имен

.NET Framework

Платформа Microsoft .NET Framework состоит из набора базовых классов и CLR

C# и .NET Framework

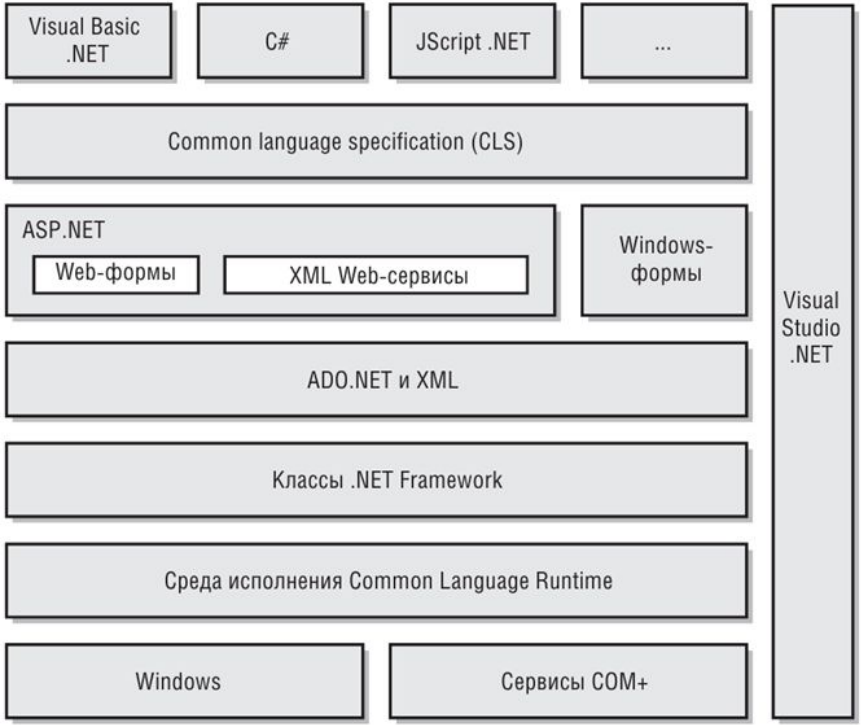
Помните, что C# и .Net Framework неразрывно связаны с друг другом. .Net Framework – это технология разработанная Microsoft для упрощения написания программ для ее продуктов(операционных систем, мобильных устройств, сайтов и др.) C# является специально разработанным языком поддержки .Net Framework. Хотя под .Net Framework можно программировать и на других языках в C# реализована полная поддержка этой технологии.

CLR, MSIL, управляемый код

Необходимо понимать, что при написании программы на C# по умолчанию программа компилируется в так называемый управляемый код MSIL(промежуточный язык), который выполняется с помощью CLR(общеязыковой средой выполнения). Это позволяет обеспечить переносимость программы с одной платформы на другую, дополнительную защиту от ошибок и ряд других преимуществ. Правда с не большой потерей в производительности.

Управляемый код – это код, который выполняется в CLR. В C# есть возможность выйти за рамки управляемого кода, если важны критерии производительности или есть другие потребности при написании программы

Управляющий код и среда Common Language Runtime



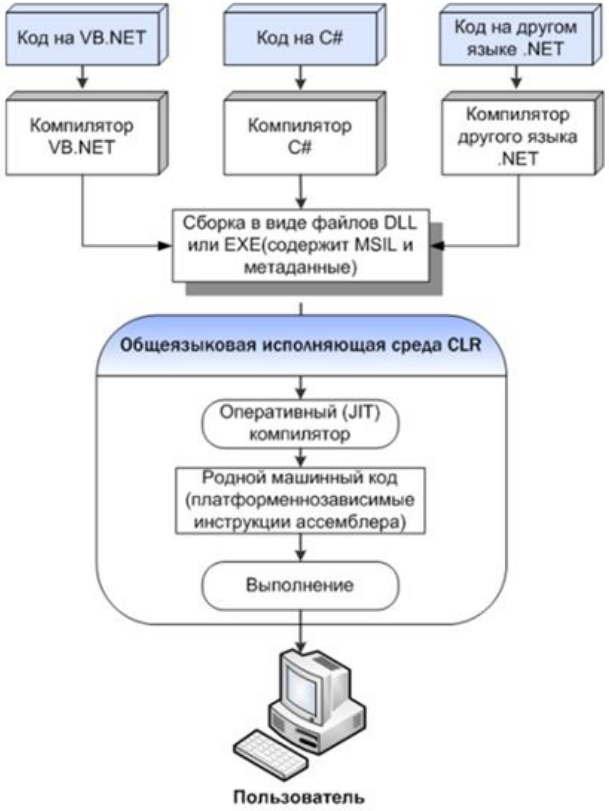
Источник:
Рейли Д.
Создание
приложений
ASP.NET

Common Language Runtime

Общеязыковая среда выполнения.

Программа с языка C# транслируется в промежуточный код MSIL

Схема компиляции .NET приложения



Столбовский Д.Н.
Разработка Web-приложений ASP .NET
с использованием Visual Studio .NET

MSIL – MS Intermediate language

```
using System;

namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

"Hello, world!" на C#

"Hello, world!" на MSIL

```
.method public static void  Main() cil managed
{
  .entrypoint

.custom instance void [mscorlib]System.STAThreadAttribute::.ctor() =
    ( 01 00 00 00 )
  // Code size    11 (0xb)
  .maxstack  8
  IL_0000:  ldstr"Hello .NET World!"
  IL_0005:  callvoid [mscorlib]System.Console::WriteLine(string)
  IL_000a:  ret
} // end of method modmain::Main
```

Visual Studio

Как создается консольный проект
Обозреватель решений
Текущий запускаемый проект

Visual Studio

Visual Studio(VS) это интегрированная среда разработки(IDE), которая существенно облегчает жизнь программистам.

Хотя с первого взгляда может показаться, что VS это одна программа, на самом деле VS состоит из множества программ. Например, компилятор, отладчик, утилиты для работы с базами данных и др.

IntelliSence

IntelliSence – Технология автодополнения в VisualStudio

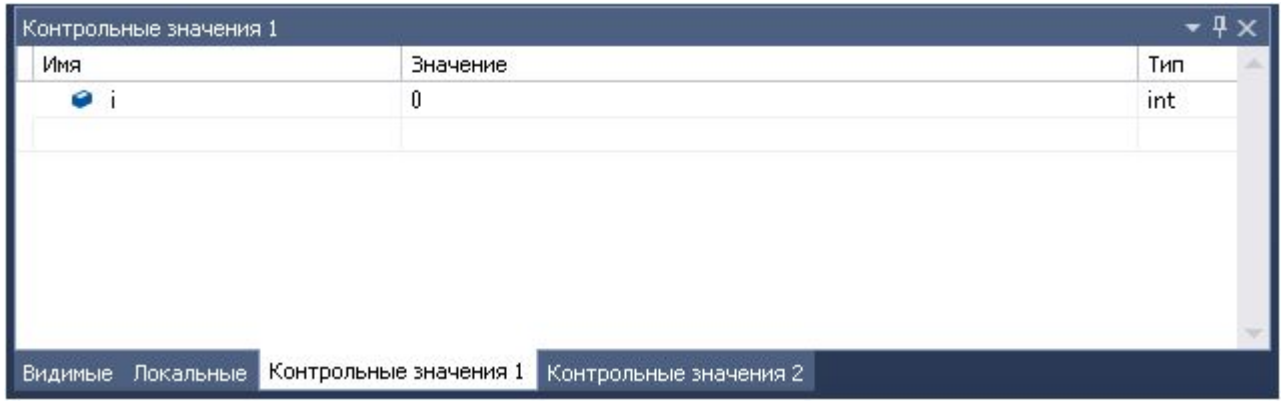
- Окончание операторов и имен
- Подсвечивание ошибок и предупреждений
- Форматирование кода
- ...

Отладка программ

Для отладки можно(и нужно ☺) использовать окна Отладка и Стек вызовов. Если вдруг их нет на экране включите их в меню Вид-Панели инструментов.

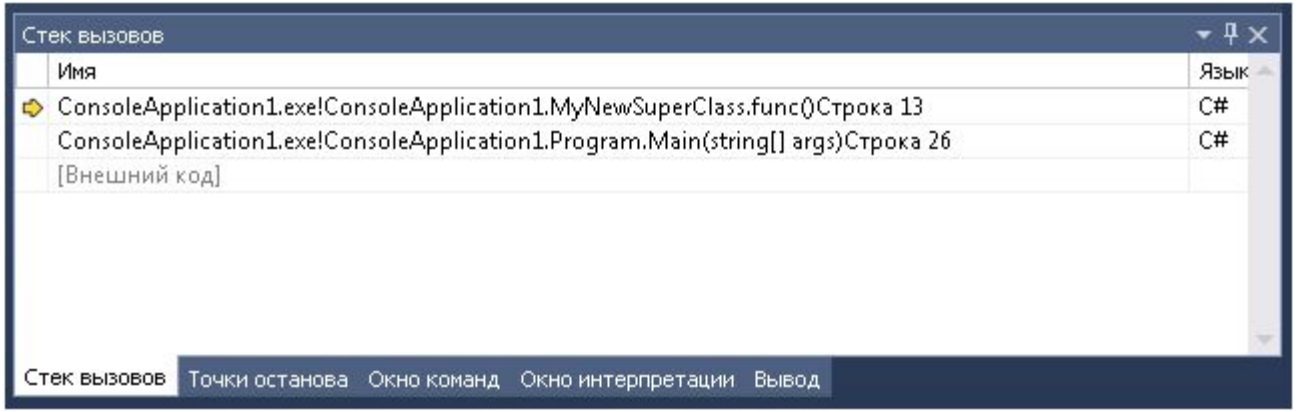
Debugging(Отладка)

Видимые
Локальные
Контрольные значения



Calls stack(Стек вызовов)

Список вызовов функций



Отладка программ

Для отладки программ существует отладчик, который позволяет вам управлять выполнением программы и смотреть, как изменяются переменные. Наиболее часто используемые клавиши для взаимодействия с отладчиком и редактором VS на следующем слайде

Удобные клавиатурные команды

Клавиша	Команда
<F9>	Добавление/снятие точки останова
<F12>	Переход к определению, объекта или метода
<Ctrl>+<M>	Развертывание и свертывание структуры кода в редакторе
<Ctrl>+<K>+<C> <Ctrl>+<K>+<U>	Закомментировать строки кода Раскомментировать строки кода
<F5>	Запуск с отладкой
<Ctlr>+<F5>	Запуск без отладки
<F10>	Трассировка с обходом
<F11>	Трассировка со входом

Outlining(учимся прятать код)

#region описание_региона
#endregion

Комментарии

Однострочный

```
//    Игнорируется текст до конца строки
```

Многострочный

```
/*
  Игнорируется всё, что заключено
  между  /* и */
*/
```

Простая программа

Любая программа состоит из каких-то обязательных элементов. В C# это класс и фигурные скобки ограничивающие некоторую область программы.

Программа должна содержать метода. Хотя программа может быть и без функций, но это уже некоторый специфический случай, который вы поймете позже.

Функция Main – является точкой входа в программу. Функция Main может отсутствовать, только в случае, если ваша программа не должна выполняться, а используется другой программой. Например, вы пишите библиотеку функций.

Элементы простой программы

- Пространства имен – контейнеры для классов
- Класс – логическая единица программы на C# в которой содержатся методы и другие элементы
- Метод или функция – подпрограмма программы в которой содержится реализация алгоритма

Главная функция Main

Функция Main – это как заглавный сайт. Только на заглавный сайт вы можете зайти из разных мест, а программа начинается всегда с главной метода (главной страницы сайта). Поэтому ее еще называют **точкой входа**

Простая программа на C#

```
//Подключение пространства имен System
using System;

namespace ConsoleApplication1//Название пространства имен
{//начало пространства имен
    class Program  //название класса
    {//начало класса
        static void Main(string[] args)
        {//начало программы
            Console.WriteLine("Hello, world!");//Тело программы
        } //конец программы
    } //конец класса
} //конец пространства имен
```

Простейшая программа на C# (консольная)

```
using System;
class MyClass
{
    static void Main()
    {
        //Вывод данных в консольном режиме
        Console.WriteLine("Hello, world!");
        Console.ReadKey();//Пауза в конце программы
    }
}
```

Переменные

Переменные в C# это ячейки памяти, которые хранят данные. Какие данные может хранить ячейка определяется ее типом. Следовательно тип определяет так же и размер ячейки. Программист должен уметь правильно выбирать типы данных для переменных в зависимости от решаемой задачи. Это приходит с опытом.

Переменные

Для хранения переменных используется специально организованный вид памяти под названием стек. В некоторых задачах использование стека, существенно облегчает их решение. Эти задачи мы рассмотрим позже.

Учтите, что под «использование стека» имеется ввиду использование «пользовательского» стека, то есть созданным самим программистом, а не стека, который отведен для переменных.

Переменная – это величина, имеющая имя, тип и значение. Значение переменной можно изменять во время работы программы.

Имя

Значение

Другой тип данных

! В переменной хранятся данные определенного типа!

Имена переменных в C#

В именах МОЖНО использовать

- латинские буквы и русские буквы

заглавные и строчные буквы различаются

- цифры

имя не может начинаться с цифры

- знак подчеркивания _

Имена переменных в C#

В именах нельзя использовать

- пробелы
- скобки, знаки +, =, !, ? и др.

Именем не может быть ключевое слово C#

class, using, namespace и др.

Полный список ключевых слов в дополнении

Типы данных

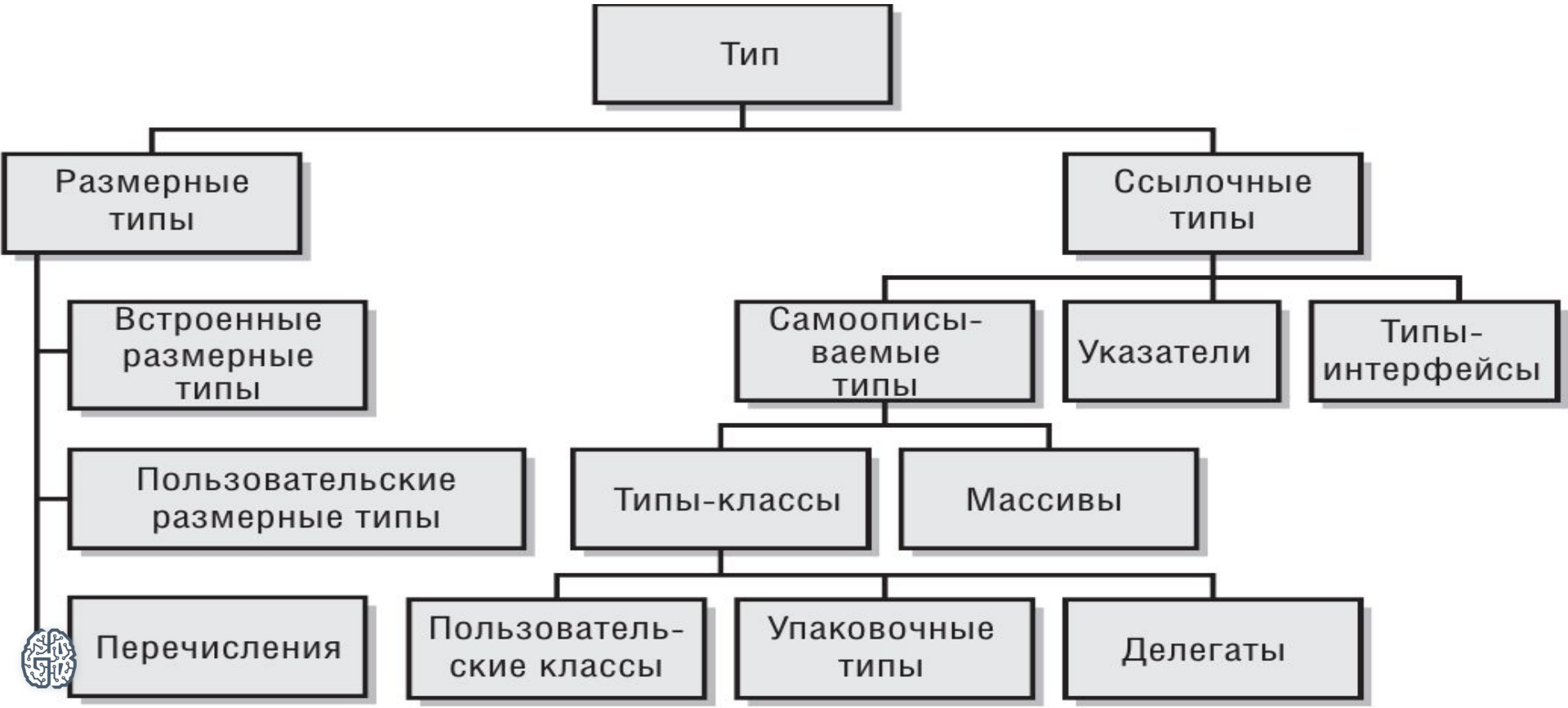
Типы данных определяют те виды задач, для решения которых можно применять данный язык.

Типы значений и ссылочные типы. Первое знакомство.

Типы данных

Важно понять, что в C# типы делятся на значимые и ссылочные. Значимые хранят значения, а ссылочные хранят ссылки на ячейки, в которых уже хранятся значения. Так как все объекты относятся к ссылочным типам, то очень важно понять разницу между ссылочными и значимыми типами.
Теоретически, можно было бы сделать все типы ссылочными, но это бы обязательно сказалось бы на быстродействии выполнения программы.

Схема типов в .NET Framework



Типы данных

C# является языком разработанным под .Net Framework. Псевдонимы типов данных это хорошо иллюстрируют. Мы можем использовать либо название из .Net Framework, либо пвсевдоним C#.
Поймите, что .Net Framework разрабатывался не только под один язык, и эти типы данных можно(и нужно) использовать в программах написанных на других языках. Это решает вопрос совместимости типов данных, который долгое время не давал возможности легко писать одну программу на разных языках.

Псевдонимы типов данных в C#

Со знаком		Без знака	
Тип .NET	**Псевдоним C#**	**Тип .NET**	**Псевдоним C#**
System.Object	*object*	*System.Enum*	*enum*
System.String	*string*	*System.Char*	*char*
System.SByte	*sbyte*	*System.Byte*	*byte*
System.Int16	*short*	*System.UInt16*	*ushort*
System.Int32	*int*	*System.UInt32*	*uint*
System.Int64	*long*	*System.UInt64*	*ulong*
System.Single	*float*	*System.Double*	*double*
System.Decimal	decimal	System.Boolean	*bool*

Петцольд Ч. «Программирование для Windows на C#. Том 1»

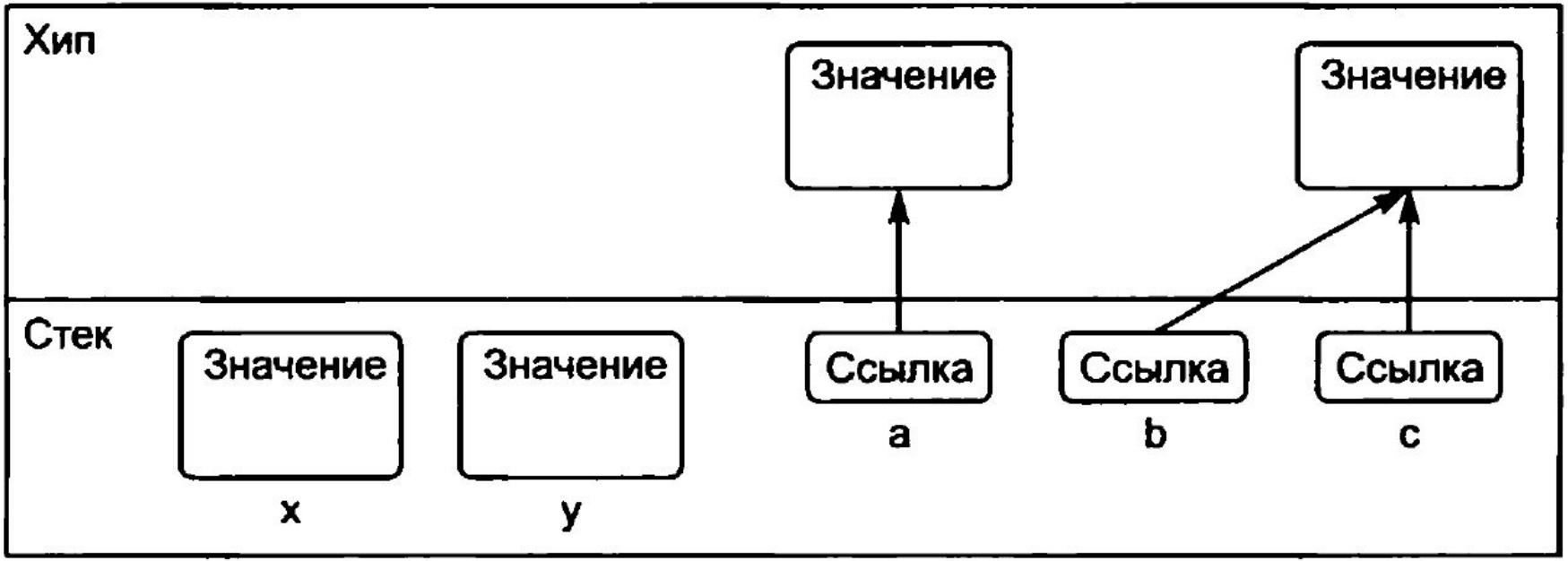
Типы значений и ссылочные типы данных

Если переменная относится к *типу значения*, то она содержит само значение, например 3,1416 или 2016

Если к *ссылочному типу*, она содержит ссылку на значение

Адрес в памяти компьютера. По этому адресу хранятся сами данные

Хранение в памяти значений значимого и ссылочного типа



Т.А. Павловская. “Программирование на языке высокого уровня

Типы значений

Тип	Значение
`bool`	Логический, предоставляет два значения: “истина” или “ложь”
`byte`	8-разрядный целочисленный без знака
`char`	Символьный
`decimal`	Десятичный (для финансовых расчетов)
`double`	С плавающей точкой двойной точности
`float`	С плавающей точкой одинарной точности
`int`	Целочисленный
`long`	Длинный целочисленный
`sbyte`	8-разрядный целочисленный со знаком
`short`	Короткий целочисленный
`uint`	Целочисленный без знака
`ulong`	Длинный целочисленный без знака
`ushort`	Короткий целочисленный без знака

Ссылочные типы

- object
- string
- массивы
- объекты

Какие типы использовать?

Тип данных определяет:

- Какие значения может принимать переменная
- Сколько места памяти занимает переменная
- Какие операции можно выполнять над переменной

Умение использовать правильный тип данных придет с опытом. Полезно знать, что в .Net Framework(или C#, как хотите) большое разнообразие встроенных типов данных, которые покрывают большинство потребностей программистов.

Целочисленные типы

Тип	Разрядность в битах	Диапазон представления чисел
byte	8	0–255
sbyte	8	–128–127
short	16	–32 768–32 767
ushort	16	0–65 535
int	32	–2 147 483 648–2 147 483 647
uint	32	0–4 294 967 295
long	64	–9 223 372 036 854 775 808–9 223 372 036 854 775 807
ulong	64	0–18 446 744 073 709 551 615

Пример

byte a; //объявили переменную **a** типа **byte**
int b, //объявили переменную **b** и переменную **c** типа **int** и
c=100; //переменной **c** присвоили значение 100

Типы для представления чисел с плавающей запятой

float	32 бита	5E-45 – 3,4E+38
double	64 бита	5E-324 – 1,7E+308

В C# чаще используется тип данных double

```
float a;      //объявили переменную a типа float
double b,     //объявили переменную b и переменную c типа double и
c=3.14;       //переменной c присвоили значение 3.14
```

Десятичный тип данных

decimal – предназначен для ведения финансовых расчетов

decimal 128 бит 1E-28 7,9E+28

Десятичный тип данных

Пример использования:

```
 decimal price;
decimal discount;
decimal discounted_price;
// Рассчитать цену со скидкой.
price = 19.95m;
discount = 0.15m; // норма скидки составляет 15%
discounted_price = price - (price * discount);
Console.WriteLine("Цена со скидкой: $" + discounted_price);
```

Символы

В C# символы представлены 16 разрядным кодом Unicode

(см. таблицы кодировки)

Пример

```
char ch;      //объявили переменную a типа char
ch = ‘A’;     //присвоили переменной значение ‘A’
ch=65;        //ошибка
char ch = '\x0041';//присвоили переменной значение символа с кодом 0041(десятичное 65)
```

Строки

Строка - ссылка на массив символов char

Строка в C# - это объект

System.String или string

Пример:
System.String s;

s=“System.String”;
string str=”string”;

Операции

математические операции (*, /, +, -, %)

присваивание (=)

операции отношения (<, >, ==, !=, >=, <=)

операция инкремента (++)

операция декремента (--)

```
int k = 2014;
int l = 10 + 6;
int j = l + k;
string str = “Tom“ + “ & ” + “Jerry”;
j++;  ++j;
--k;  k--;
```

Логический тип данных

```
bool b;        //объявили переменную b типа bool
b=false;       //в переменную b присвоили значение false
b=2*2==4;      //в переменную b присвоили значение true
```

bool – «истина»(true) и «ложь»(false)

Логические операции

Оператор	Значение
&	И
|	ИЛИ
^	Исключающее ИЛИ
&&	Укороченное И
||	Укороченное ИЛИ
!	НЕ

p	q	p & q	p | q	p ^ q	!p
false	false	false	false	false	true
true	false	false	true	true	false
false	true	false	true	true	true
true	true	true	true	false	false

*Неявно типизированные переменные

Неявно типизированные переменные это некоторое удобство для программистов, которое позволяет описать тип переменной альтернативным способом. В этом случае выбор типа возлагается на компилятор. Понимание этого удобства приходит с опытом. На первых парах, я бы рекомендовал явно указывать тип переменной.

Не путайте неявно типизированные переменные с динамическими переменными, тип которых может изменяться в процессе выполнения программы. Это не одно и тоже!

*Неявно типизированные переменные

Начиная с C# версии 3.0 компилятору предоставляется возможность самому определять тип переменной исходя из значений, которым она инициализируется

Пример:

var e=2.7183;//Тип double – компилятор сам принял решение

var e=2.7183F;//Тип float – компилятору «подсказали» указав F в конце числа

Суффиксы целых и вещественных констант

Суффикс	Значение
L, l	Длинное целое (long)
U, u	Беззнаковое целое (unsigned)
F, f	Вещественное с одинарной точностью (float)
D, d	Вещественное с двойной точностью (double)
M, m	Финансовое десятичного типа (decimal)

Приведение типов при помощи класса Convert

Преобразует значение одного базового типа данных к другому базовому типу данных

```
int i=100;
string s=Convert.ToString(i);
double d=3.14;
int i=Convert.ToInt32(d);
```

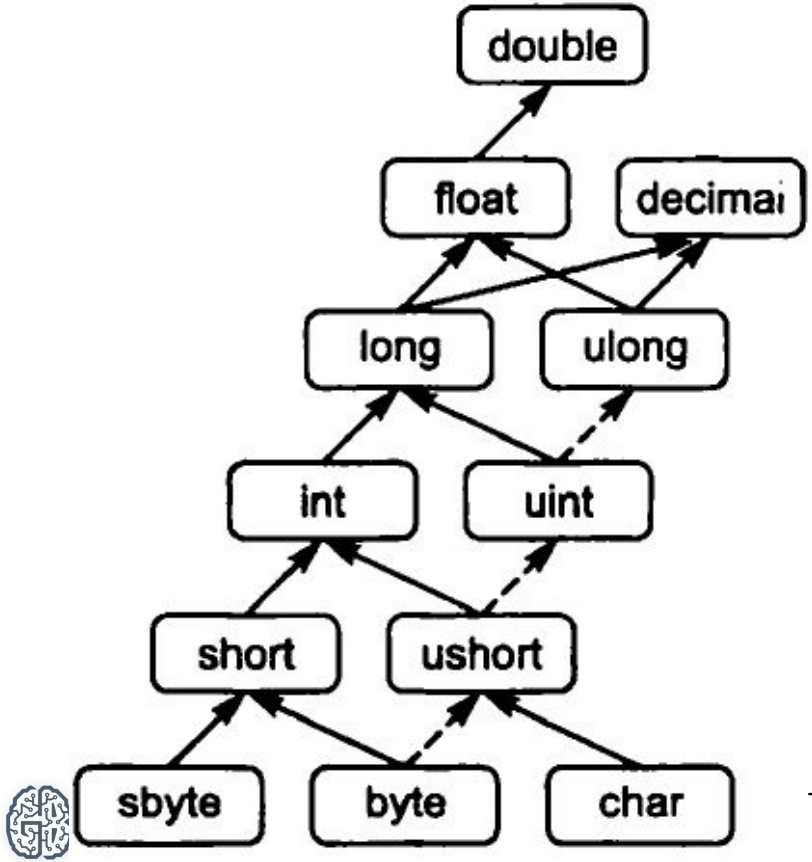
Преобразование и приведение совместимых типов

При вычислении выражений может возникнуть необходимость в преобразовании типов. Если операнды, входящие в выражение, одного типа и операция для этого типа определена, то результат выражения будет иметь тот же тип. Если операнды разного типа и/или операция для этого типа не определена, перед вычислениями автоматически выполняется преобразование типа по правила, обеспечивающим приведение более коротких типов к более длинным.

Преобразование и приведение совместимых типов

Если неявного преобразования из одного типа в другой не существует, программист может задать явное преобразование типа с помощью операции (тип)x.

Неявные арифметические преобразования типов



Т.А. Павловская. “Программирование на языке высокого уровня

Преобразование и приведение совместимых типов

Автоматическое преобразование типов (не явное преобразование)

```
int i;

float f;

i=10;

f=i;
```

Приведение несовместимых типов (явное преобразование)

```
double x, y;

int i=(int) (x/y)
```

Консоль

Консоль – это способ взаимодействия пользователя с компьютером. Для простого пользователя консоль часто ассоциируется с черным экраном с буквами. Для простого пользователя умение работать в консоли, это чуть ли не высший пилотаж. Но для программиста, консоль это один из самых простых способов взаимодействия с компьютером

Консоль

Работа с консолью в C# происходит с помощью класса Console, который содержит в себе большое количество методов и свойств для взаимодействия с консолью.
Что такое методы вы узнаете уже на этом уроке.
Свойства – пока можно воспринимать как переменные, которые принадлежат классу Console.

Два способа вывода на экран консоли

Для вывода данных на экран консоли используются методы Write и WriteLine. Разница лишь в том, что WriteLine в отличии от Write переводит курсор на следующую строчку. То есть если следующий вывод(или ввод) нужно делать со следующий строки, то используем WriteLine.

У этих методов есть множество разновидностей, которые профессионально называются перегрузками. Можно использовать перегрузку, когда получится одна длинная строка используя возможности C# автоматически преобразовывать данные в строку. А можно использовать перегрузку, в котором вначале задается строка форматирования, а потом данные для вывода. В этом случае у программиста появляется больше возможностей для управлением выводом.

Управляющие последовательности

Управляющая последовательность	Описание
\n	Новая строка (перевод строки)
\r	Возврат каретки
\t	Горизонтальная табуляция
\'	Одинарная кавычка
\"	Двойная кавычка
\\	Обратная косая черта

Форматированный вывод

Примеры:

```
WriteLine("В феврале {0} или {1} дней.", 28, 29);
   WriteLine("В феврале {0,10} или {1,5} дней.", 28, 29);
   WriteLine("Деление 10/3 дает: {0:#.##}", 10.0 / 3.0);
decimal balance;
balance = 12323.09m;
   WriteLine("Текущий баланс равен {0:C}", balance);
```

еще примеры

Различные спецификации формата в применении к целому числу 12 345

Тип форматирования	Код формата	Результат
Currency (денежные суммы)	C	$12,345.00
	C1	$12,345.0
	C7	$12,345.0000000
Decimal (десятичный)	D	12345
	D1	12345
	D7	0012345
Exponential (экспоненциальный)	E	1.234500E+004
	E1	1.2E+004
	E7	1.2345000E+004
Fixed point (с фиксированной точкой)	F	12345.00
	F1	12345.0
	F7	12345.0000000
General (общий)	G	12345
	G1	1E4
	G7	12345
Number (числовой)	N	12,345.00
	N1	12,345.0
	N7	12,345.0000000
Percent (процент)	P	1,234,500.00
	P1	1,234,500.0
	P7	1,234,500.0000000
Hexadecimal (шестнадцатеричный)	X	3039
	X1	3039
	X7	0003039

Петцольд Ч.
«Программирование для Windows на C#. Том 1»

Чтение данных с консоли

Для чтения данных из консоли существует метод ReadLine. Этот метод приостанавливает выполнение программы, пока не получит признак конца ввода(обычно это нажатие клавиши Enter). После чего передает данные из консоли в переменную.

При работе с ReadLine может возникнуть исключение. Про исключения мы поговорим попозже. Пока воспринимайте их как ошибки.

Ввод данных с консоли

```
string s = Console.ReadLine();
```

string Console.ReadLine()
Считывает следующую строку символов из стандартного входного потока.

Исключения:
System.IO.IOException
System.OutOfMemoryException
System.ArgumentOutOfRangeException

Исключения – это ошибки, которые могут произойти во время выполнения этого метода

Класс Convert и метод структур Parse

Так как метод ReadLine класса Console возвращает строку(то есть результатом его работы является строка), то часто ее нужно преобразовать в другой тип данных. Это можно сделать разными способами.

1. Использовать метод класса Convert
2. Использовать метод структуры(int, double, decimal и др.)

Разница в их использовании на данном этапе нам не принципиальна, поэтому можно использовать любой из способов. Сейчас важно, что вы уже познакомились с понятиями класса, структуры и метода и начали учиться использовать, то что уже сделано другими программистами.

Конвертация данных. 1 Способ. Использование класса Convert

```
int a = Convert.ToInt32(s);
```

int Convert.ToInt32(string value) (+ перегрузок: 18)
Преобразует заданное строковое представление числа в эквивалентное 32-разрядное знаковое целое число.

Исключения:
System.FormatException
System.OverflowException

```
double d = Convert.ToDouble(Console.ReadLine());
```

string Console.ReadLine()
Считывает следующую строку символов из стандартного входного потока.

Исключения:
System.IO.IOException
System.OutOfMemoryException
System.ArgumentOutOfRangeException

Конвертация данных. 2 способ. Использование структур

```
int a = int.Parse(s);
```

int int.Parse(string s) (+ перегрузок: 3)
Преобразовывает строковое представление числа в эквивалентное ему 32-битовое знаковое целое число.

Исключения:
- System.ArgumentNullException
- System.FormatException
- System.OverflowException

```
double d = double.Parse(Console.ReadLine());
```

string Console.ReadLine()
Считывает следующую строку символов из стандартного входного потока.

Исключения:
- System.IO.IOException
- System.OutOfMemoryException
- System.ArgumentOutOfRangeException

Ввод и вывод. Вариант 1. Автоматическое преобразование в строку

```
class Program
{
    static void Main(string[] args)
    {
        double x;
        double y;
        string str = Console.ReadLine();
        x = Convert.ToDouble(str);
        Console.Write("Введите второе число: ");
        y = Convert.ToDouble(Console.ReadLine());
        double z = x + y;
        Console.WriteLine(x + "+" + y + "=" + z);//Преобразование в строку
        Console.ReadKey();
    }
}
```

Ввод и вывод. Вариант 2. Форматированный вывод

```
class Program
{
    static void Main(string[] args)
    {
        double x;
        double y;
        string str = Console.ReadLine();
        x = Convert.ToDouble(str);
        Console.Write("Введите второе число: ");
        y = Convert.ToDouble(Console.ReadLine());
        double z = x + y;
        Console.WriteLine("{0} + {1} = {2}", x, y, z);//Форматированный вывод
        Console.ReadKey();
    }
}
```

Функции - методы

Функция или метод?

Функции и методы – это технически одно и тоже. Только функции могут не принадлежать классам, а методы принадлежат классу. С этой точки зрения в C# все функции это методы.

Чтобы понять в чем отличие нужно чуть больше познакомится с классами. Вы можете называть функции методами, а методы функциями. В C# большой ошибки не будет. Со временем вы обязательно поймете в чем разница.

Описание метода

Метод является частью класса, поэтому описываться он должен внутри класса. Чтобы описать метод нужно придумать ему имя, определиться будет ли он возвращать значение и если будет, то какого типа это значение. Далее в фигурных скобках описываться тело метода. Так как мы еще не очень знакомы с объектами все методы у нас должны быть статическими. Если метод статический, он принадлежит классу и для его вызова не нужно создавать объект класса.

Параметры метода

Методы чаще всего предназначены для обработки данных. Для передачи данных внутрь метода используются параметры. Для этого после названия метода в скобках через запятую перечисляются формальные параметры метода.

Методы

модификаторы

модификаторы

тип возвращаемого значения

имя

тело

```
public static void Pause()
{
    Console.ReadKey();
}
```

Вызов метода

Для вызова метода нужно написать имя метода и скобки. Скобки после названия – признак, что это метод, а не переменная или свойство. Если метод принимает параметры, то в скобках перечисляем фактические параметры (то что передается внутрь метода для обработки).

Вызов метода

```
using System;
class Program
{
    static void Pause()
    {
        Console.ReadKey();
    }
    static void Main()
    {
        Pause();//Вызов функции из другой функции
    }
}
```

Метод с параметрами

параметры метода

```
public static void Pause(string msg)
        {
            Console.WriteLine(msg);
            Console.ReadKey();
        }
```

Вызов метода с параметрами

```
static void Main()
{
    Pause("Нажмите любую клавишу");//Вызов функции с параметром
}
```

Перегрузка методов

Перегрузка – это создание метода с таким же именем, но с другими параметрами.

Раньше для подпрограмм, который делали похожие действия, создавали функции с похожими, но разными именами. В C# для этого можно создать функцию с одним именем, но с разными параметрами. Это существенно облегчает труд программистов, так как не требуется запоминать различные имена функций и упрощает чтение программы.

Не смотря, на некоторую кажущую сложность в перегрузке нет ничего сверхстественного. Попробуйте сами написать несколько функций с одинаковыми названиями, но с разным количеством параметров. При попытке обратится к перегруженной метода, компилятор подскажет вам что существую разные разновидности.

Перегрузка методов

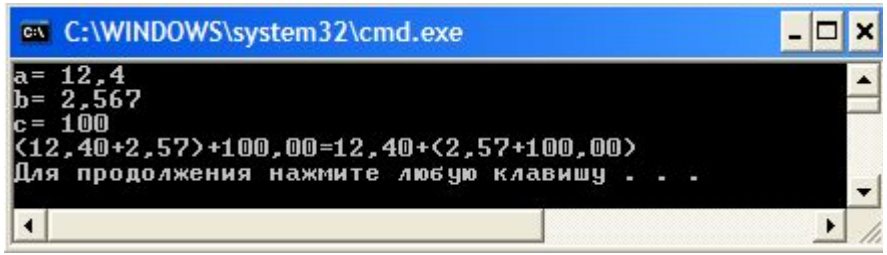
Перегрузка – это создание метода с таким же именем, но с другими параметрами

```
static void Pause(string msg)
{
    Console.WriteLine(msg);
    Console.ReadKey();
}

static void Pause()
{
    Console.ReadKey();
}
```

Практика

Запрашивается с клавиатуры три вещественных числа, и выводит на консоль следующее сообщение (вещественные числа выводятся с точностью до 2 знаков после запятой):



Решение

```
static void Main(string[] args)
{
    Console.Write("a=");
    //Считываем данные из консоль в переменную типа string
    string s = Console.ReadLine();
    double a = Convert.ToDouble(s);
    Console.Write("b=");
    //Считываем данные и тут же переводим их в тип double
    double b = Convert.ToDouble(Console.ReadLine());
    //Считываем данные и тут же переводим их в тип double
    double c = double.Parse(Console.ReadLine());
    Console.WriteLine("({0:0.00}+{1:0.00})+{2:0.00}={3:0.00}+({4:0.00}+{5:0.00})", a, b, c, c, b, a);
}
```

Интересно

```
2016.ToString();  //Преобразование в строку литерала

(i + j).ToString();  //Преобразование в строку типа int
```

Домашняя работа

1. Написать программу “Анкета”. Последовательно задаются вопросы (имя, фамилия, возраст, рост, вес). В результате вся информация выводится в одну строчку.

   а) используя склеивание;
   б) используя форматированный вывод.

1. Ввести вес и рост человека. Расчитать и вывести индекс массы тела по формуле I=m/(h*h);

3. а) Написать программу, которая подсчитывает расстояние между точками с координатами x1, y1 и x2,y2 по формуле r=Math.Sqrt(Math.Pow(x2-x1,2)+Math.Pow(y2-y1,2. Вывести результат используя спецификатор формата F2 (с двумя знаками после запятой);
   б) *Выполните предыдущее задание оформив вычисления расстояния между точками в виде метода;

4. Написать программу обмена значениями двух переменных
   а) с использованием третьей переменной;
   б) *без использования третьей переменной.

5. а) Написать программу, которая выводит на экран ваше имя, фамилию и город проживания.
   б) *Сделайте задание, только вывод организуйте в центре экрана
   в) **Сделайте задание б с использованием собственных методов (например, Print(string ms, int x,int y)

Домашняя работа

3. а)Написать программу, которая подсчитывает расстояние между точками с координатами x1, y1 и x2,y2 по формуле r=Math.Sqrt(Math.Pow(x2-x1,2)+Math.Pow(y2-y1,2. Вывести результат используя спецификатор формата F2(с двумя знаками после запятой);

б)*Выполните предыдущее задание оформив вычисления расстояния между точками в виде метода;

4. Напишите программу обмена значениями двух переменных

а) с использованием третьей переменной;

б)*без использования третьей переменной.

***Пишите программы разбивая на методы**

Домашняя работа

5.а)Напишие программу, которая выводит на экран ваше имя, фамилию и город проживания.
б)*Сделайте задание, только вывод организуйте в центре экрана
в)**Сделайте задание б с использованием собственных методов (например, Print(string ms, int x,int y)

***Пишите программы разбивая на методы**

Ключевые слова C#

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	volatile
void	while			

add	dynamic	from	get	global
group	into	join	let	orderby
partial	remove	select	set	value
var	where	yield		

Литература и ссылки

1. Т.А. Павловская. “Программирование на языке высокого уровня”, 2009 г.
2. Петцольд Ч. “Программирование на C#. Т1”, 2001
3. Климов А. “C#. Советы программистам”, 2008
4. Г.Шилдт. “C# 4.0. Полное руководство”

Конец

Почувствуй себя профессионалом

Многие пользователи слышали о так называемых DLL’ках. Чтобы быстро перейти от простого пользователя к программисту, давайте научимся их создавать сами. Тем более, что это продемонстрирует нам, что такое .Net Framework, так как по существу это просто большое количество DLL’ок написанных другими программистами.

Почувствуй себя профессионалом

Создайте новый проект. Выберите шаблон “Библиотека классов”, назовите его MyDLL. Скопируйте в тело класса две ранее созданные метода Pause и Pause(string str). Добавьте к их заголовком(перед static) слова public. Это нужно чтобы мы могли обратиться к ним из другого класса.

Запустите. DLL готова. Теперь перейдите обратно в наш проект и подключите эту DLL

*Создание собственной библиотеки

Новый проект

Библиотека классов

Описание в ней методов Pause() и Pause(string str)

Компиляция

Демонстрация подключения библиотеки

Не рассмотренные темы

Область действия и время существования переменных

sizeof() – размер типа в байтах

Консоль

Программа должна обрабатывать данные, а данные нужно откуда-то брать. Ввод данных с консоли наиболее простой способ, хотя, нужно понимать, что данные могут вводить из разных мест.

Консоль – это способ взаимодействия пользователя с компьютером. Для пользователя консоль ассоциируется с черным экраном с букавками. И часто ставиться чуть-ли не в высший пилотаж для работы за компьютером. Но для программиста, консоль это самый просто способ взаимодействия с компьютером

Форматированный вывод

WriteLine("Деление 10/3 дает:" +10.0/3.0);

WriteLine("форматирующая строка",arg0,arg1, … ,argN);

Форматирующая строка:

{argnum, width: fmt}

Форматированный вывод. Использование спецификатора {0:0.00 }

```
String.Format("{0:0.00}", 123.4567);      // "123.46"

String.Format("{0:0.00}", 123.4);         // "123.40"

String.Format("{0:0.00}", 123.0);         // "123.00"
```

еще примеры