Программирование на C#

Карбаев Д.С., 2016

Рейтинг языков программирования

Position Jan 2012	Position Jan 2011	Delta in Position	Programming Language	Ratings Jan 2012	Delta Jan 2011
1	1	=	Java	17.479%	-0.29%
2	2	=	C	16.976%	+1.15%
3	6	↑↑↑	C#	8.781%	+2.55%
4	3	↓	C++	8.063%	-0.72%
5	8	↑↑↑	Objective-C	6.919%	+3.91%
6	4	↓↓	PHP	5.710%	-2.13%
7	7	=	(Visual) Basic	4.531%	-1.34%
8	5	↓↓↓	Python	3.218%	-3.05%
9	9	=	Perl	2.773%	-0.08%
10	11	↑	JavaScript	2.322%	+0.73%

Источник: TIOBE Software

Рейтинг языков программирования

Feb 2015	Feb 2014	Change	Programming Language	Ratings	Change
1	1		C	16.488%	-1.85%
2	2		Java	15.345%	-1.97%
3	4	▲	C++	6.612%	-0.28%
4	3	▼	Objective-C	6.024%	-5.32%
5	5		C#	5.738%	-0.71%
6	9	▲	JavaScript	3.514%	+1.58%
7	6	▼	PHP	3.170%	-1.05%
8	8		Python	2.882%	+0.72%

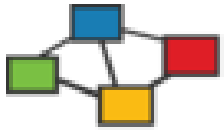
Рейтинг языков программирования

Feb 2016	Feb 2015	Change	Programming Language	Ratings	Change
1	2	↑	Java	21.145%	+5.80%
2	1	↓	C	15.594%	-0.89%
3	3		C++	6.907%	+0.29%
4	5	↑	C#	4.400%	-1.34%
5	8	↑	Python	4.180%	+1.30%
6	7	↑	PHP	2.770%	-0.40%
7	9	↑	Visual Basic .NET	2.454%	+0.43%
8	12	⇈	Perl	2.251%	+0.86%
9	6	↓	JavaScript	2.201%	-1.31%
10	11	↑	Delphi/Object Pascal	2.163%	+0.59%
11	20	⇈	Ruby	2.053%	+1.18%
12	10	↓	Visual Basic	1.855%	+0.14%
13	26	⇈	Assembly language	1.828%	+1.08%
14	4	⇊	Objective-C	1.403%	-4.62%

Вакансии

- Lead/Senior .NET/C# Software Engineer;
- Разработчик .NET / ведущий разработчик .NET

- CQG, EPAM, Mercury Development , Открытый код, КРОК, НПК Разумные решения, ИР-Тех, АйСёртанти, Webzavod



Smart Solutions
Living schedules - easy as 1-2-3

Справочная информация

Справочная информация

- Версия языка: C# 5.0
- Платформа разработки:
  Visual Studio 2015 Community
  http://www.visualstudio.com/
- Литература:
  - Герберт **Шилдт**. C# 4.0 – полное руководство, 2011.
  - Эндрю **Троелсен**. Язык программирования C# 5.0 и платформа .NET 4.5, 2013
  - Джесс Либерти. Программирование на C#, 2009
  - Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#, 2013.
- Ссылки:
  - MSDN Library: Средства разработки - Visual Studio 2015 - Visual C#
  - http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx
  - http://msdn.microsoft.com/ru-ru/vstudio/hh341490

Дополнительная литература

- Алгоритмы.Построение и анализ | Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн
- Совершенный код | С. Макконнелл
- Эффективное использование C# | Билл Вагнер
- Язык программирования C# | А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд
- C#. Программирование для профессионалов | Джон Скит
- Essential C# 4.0 | Mark Michaelis

Справочная информация

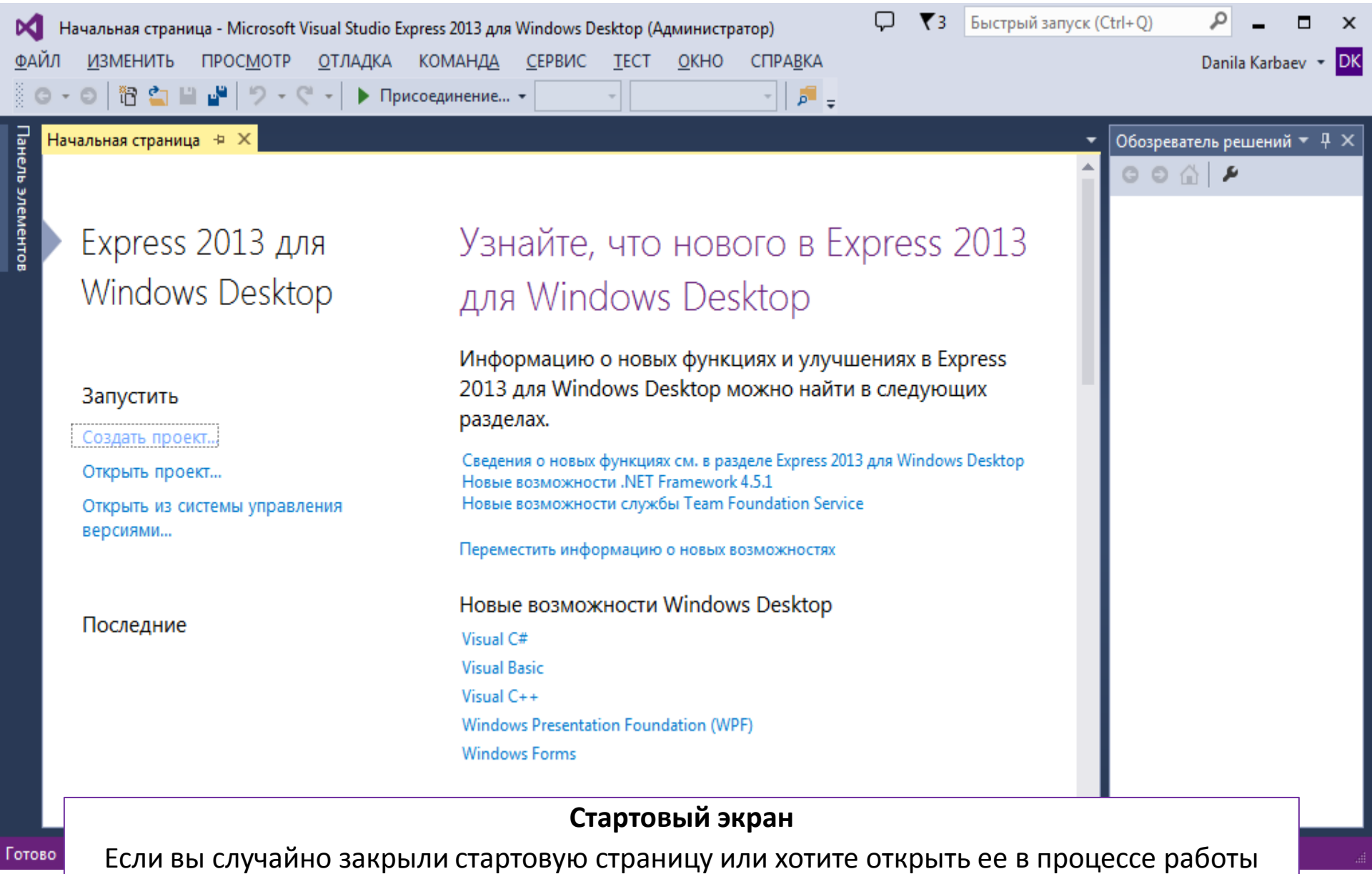
Вопросы и ответы

- **Q**: Что такое C#?
- **A**: C# - объектно-ориентированный язык программирования разработки приложений для платформы Microsoft .NET Framework .
- **Q**: Что такое .NET Framework?
- **A**: .NET Framework - программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является исполняющая среда Common Language Runtime (CLR), способная выполнять как обычные программы, так и серверные веб-приложения. .NET Framework поддерживает создание программ, написанных на разных языках программирования.
- **Q**: Есть ли какие-либо правила по оформлению исходного кода? Если да, то какие?
- **A**: Исходный код необходимо оформить в соответствии с общепринятыми нормами (http://msdn.microsoft.com/en-us/library/czefa0ke(v=vs.71).aspx)

Первая простая программа

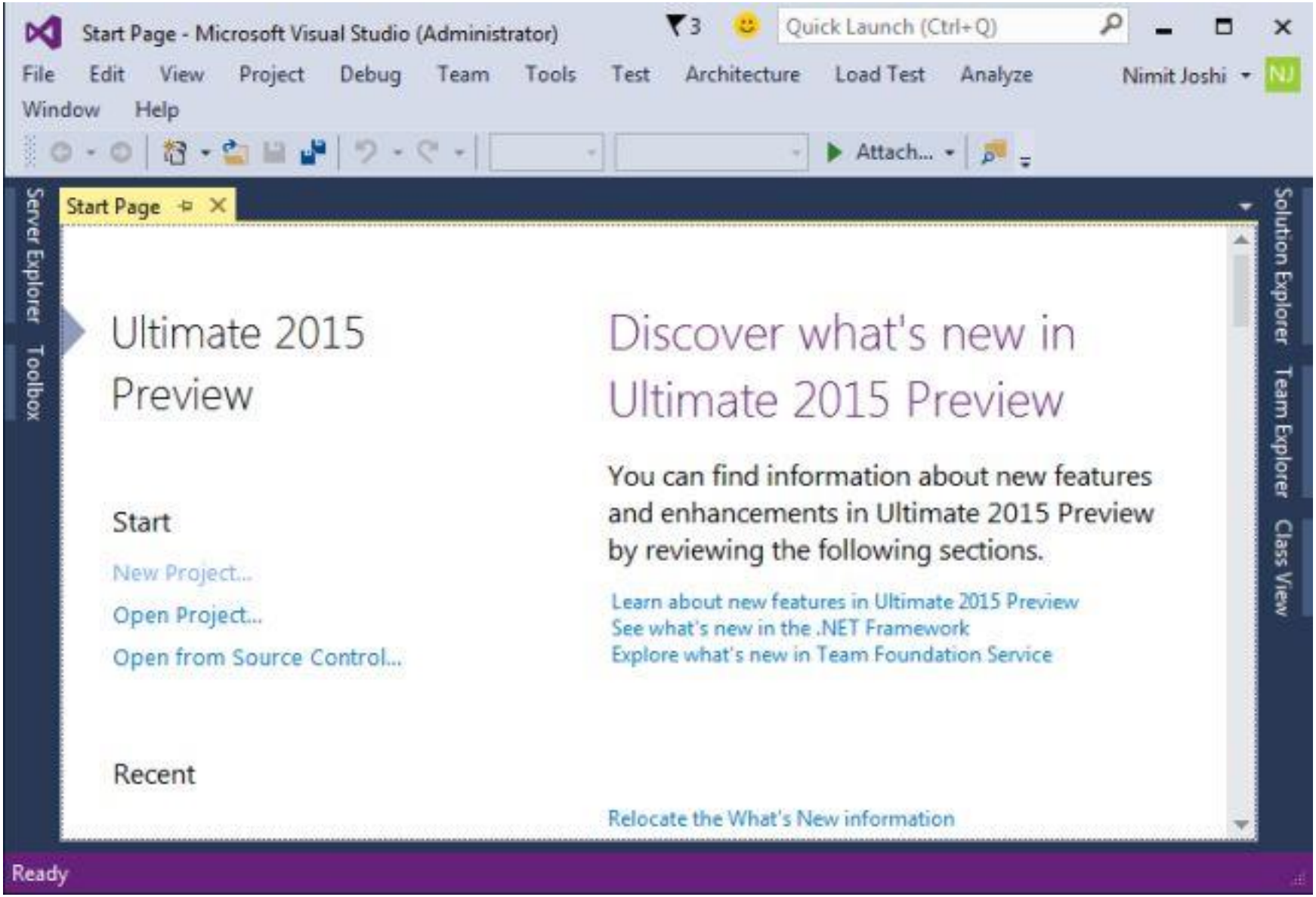
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        // Любая программа на C# начинается с вызова метода Main().
        static void Main(string[] args)
        {// Начало программы.
           // Вывод текста на экран.
            Console.WriteLine("Простая программа на C#.");
        } // Конец программы.
    }
}
```

Visual Studio 2013

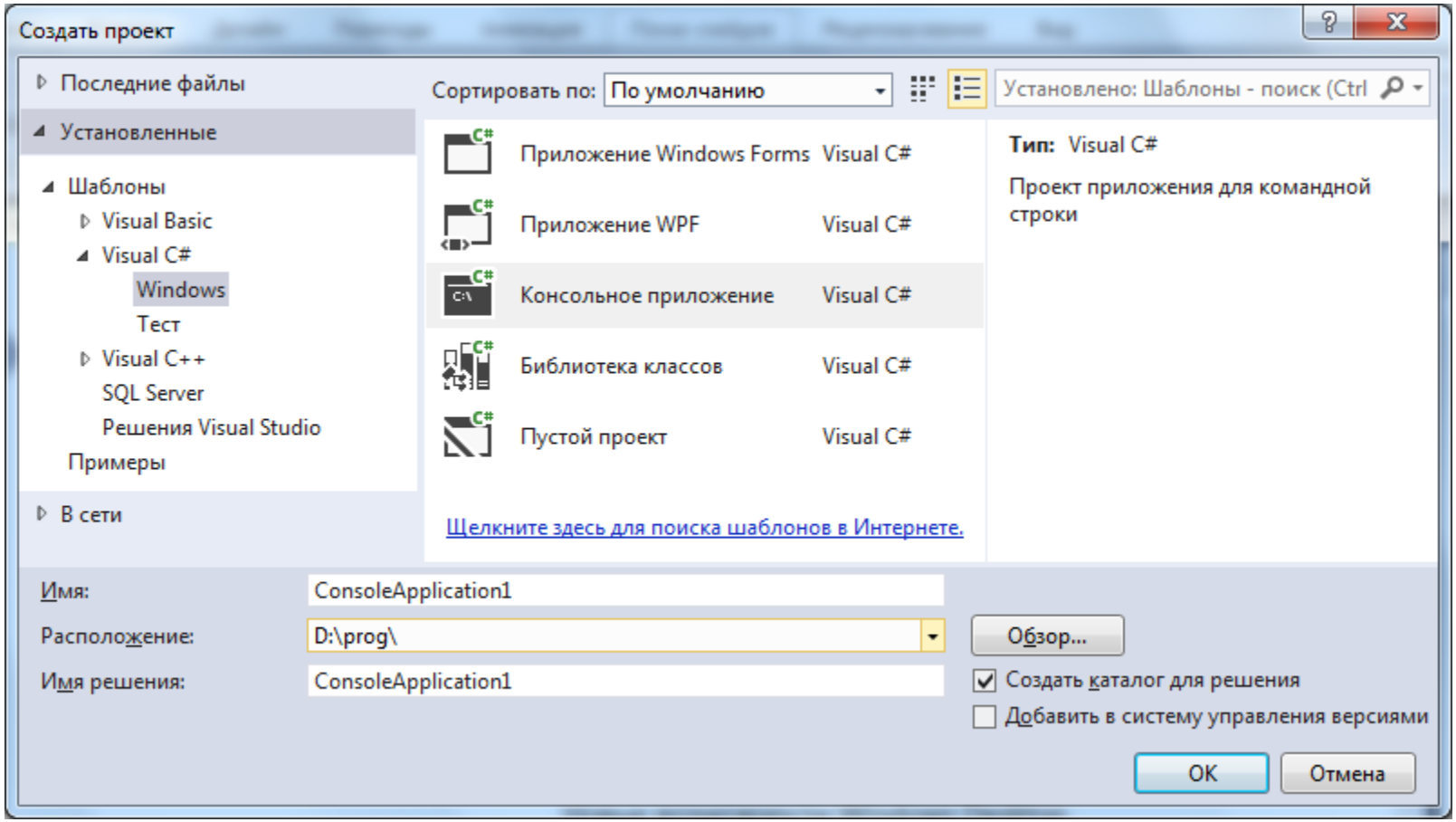


Стартовый экран

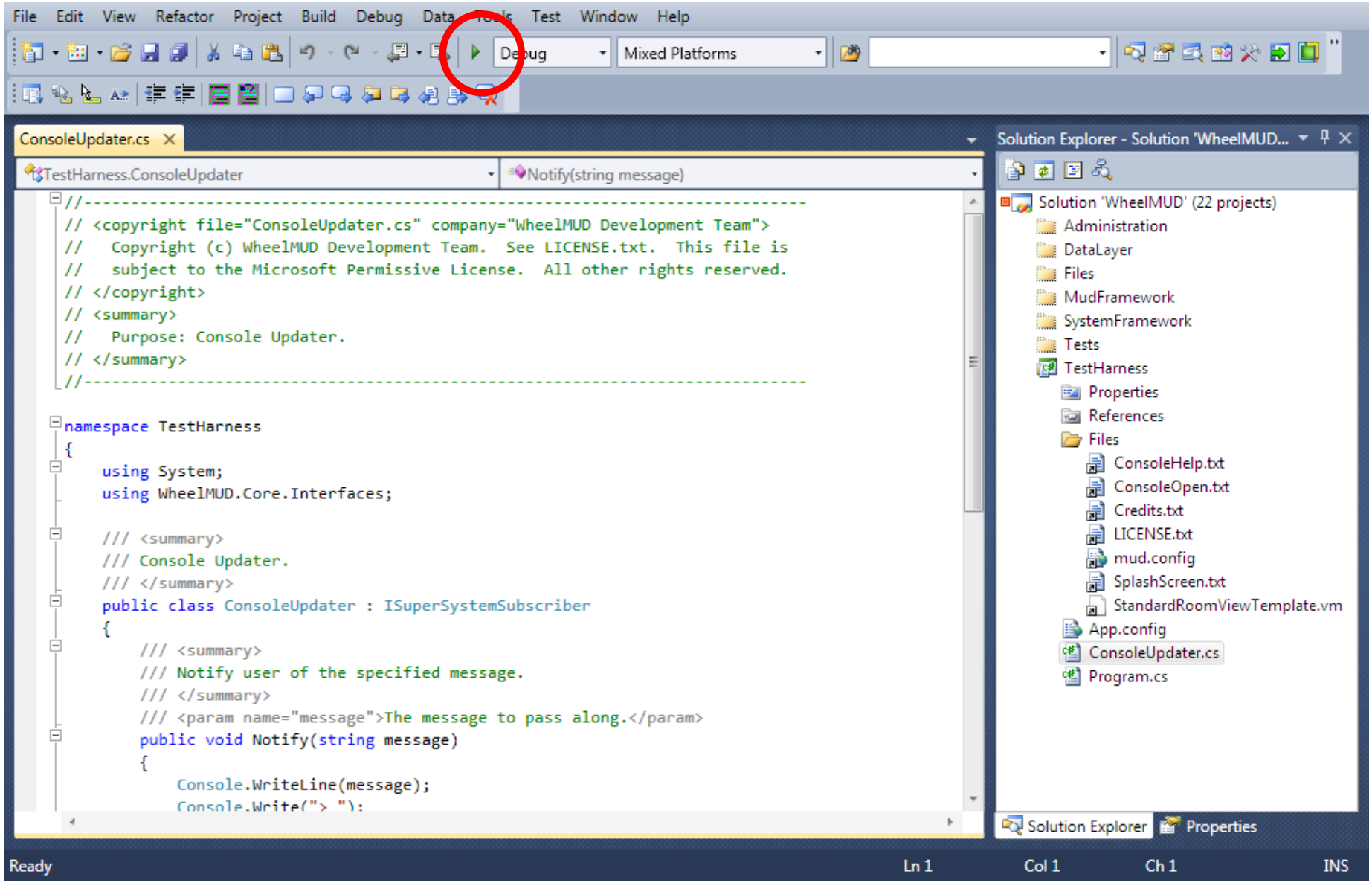
Если вы случайно закрыли стартовую страницу или хотите открыть ее в процессе работы над проектом, просто выберите пункт меню *Help/Show Start Page*.


Start Page - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
File Edit View Project Debug Team Tools Test Architecture Load Test Analyze Nimit Joshi
Window Help
Attach...
Server Explorer
Toolbox
Start Page
Solution Explorer
Team Explorer
Class View
Ultimate 2015 Preview
Start
New Project...
Open Project...
Open from Source Control...
Recent
Discover what's new in Ultimate 2015 Preview
You can find information about new features and enhancements in Ultimate 2015 Preview by reviewing the following sections.
Learn about new features in Ultimate 2015 Preview
See what's new in the .NET Framework
Explore what's new in Team Foundation Service
Relocate the What's New information
Ready

Visual Studio 2015



Для создания нового проекта выберите пункт меню File/New Project (Файл-Создать проект)



Пример рабочего интерфейса редактирования исходного кода программы.

Вторая простая программа

- В приведенной программе создаются две переменные — x и y:

```
        static void Main(string[] args){
            int x; // здесь объявляется переменная целого типа
            int y; // здесь объявляется еще одна переменная
            x = 100; // здесь переменной x присваивается значение 100
            Console.WriteLine("x содержит значение " + x);
            y = x / 2;
            Console.Write("y содержит значение x / 2: ");
            Console.WriteLine(y);
        }
```

- Выполнение этой программы дает следующий результат.

```
x содержит 100
y содержит x / 2: 50
```

- В целом, для объявления переменной служит следующий оператор:

```
тип имя_переменной;
```

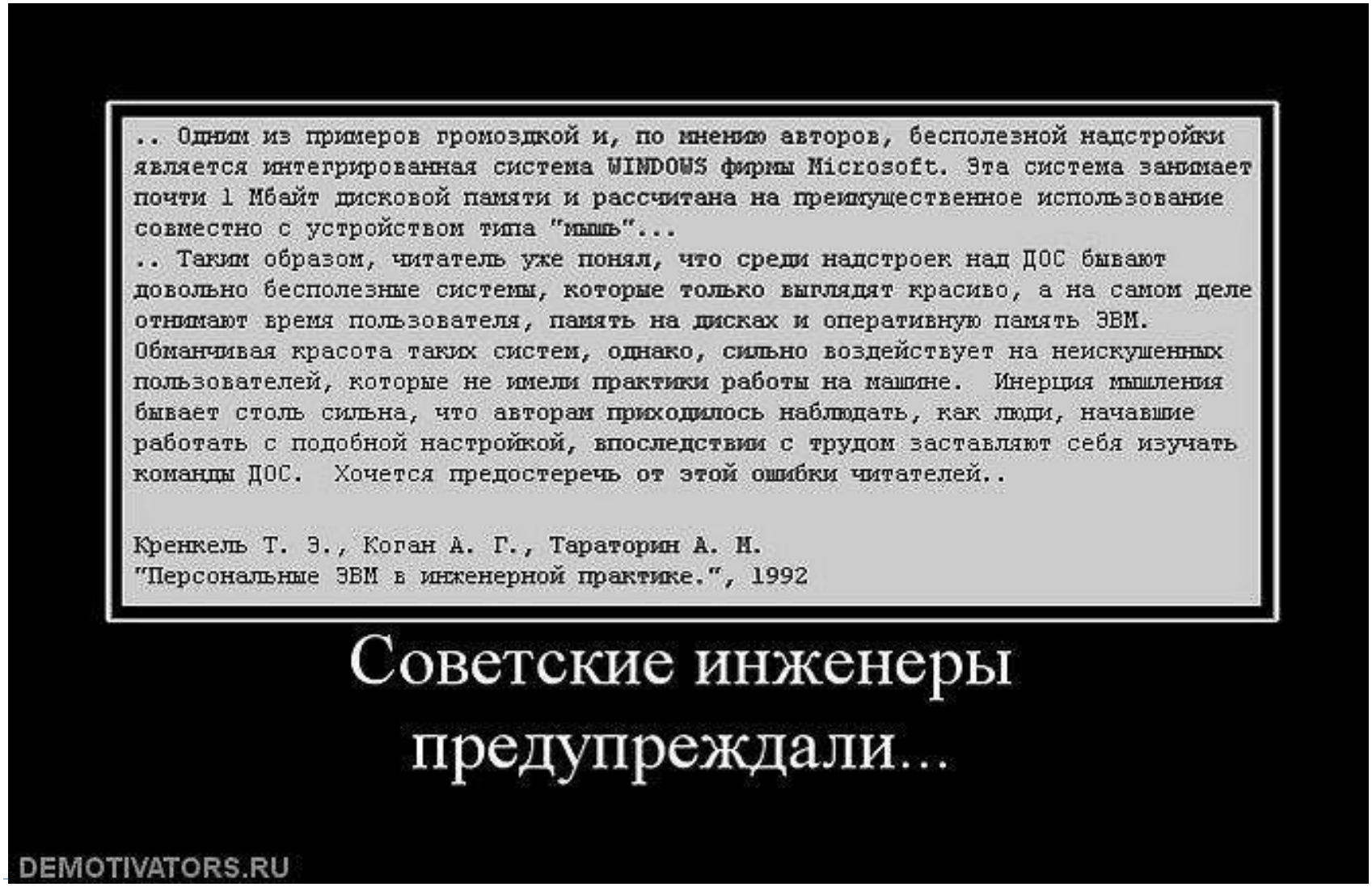
где тип — это конкретный тип объявляемой переменной,
имя_переменной — имя самой переменной.

Переменные и операции

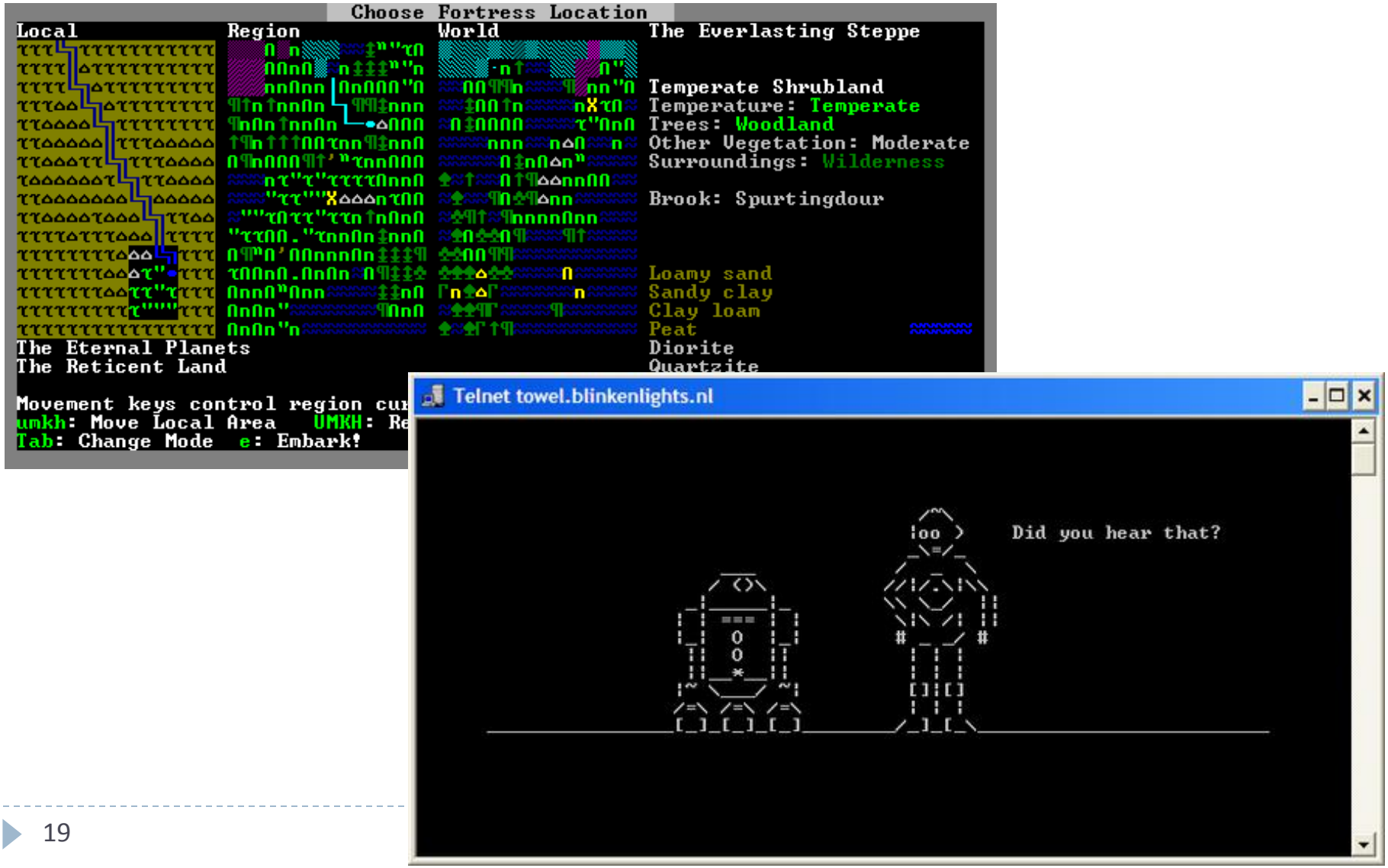
- Переменные x и y могут быть объявлены следующим образом:
  - `int x, y; // обе переменные объявляются в одном операторе`

- Арифметические операции
  - + Сложение
  - - Вычитание
  - * Умножение
  - / Деление
  - % Остаток от деления

Работа с консолью



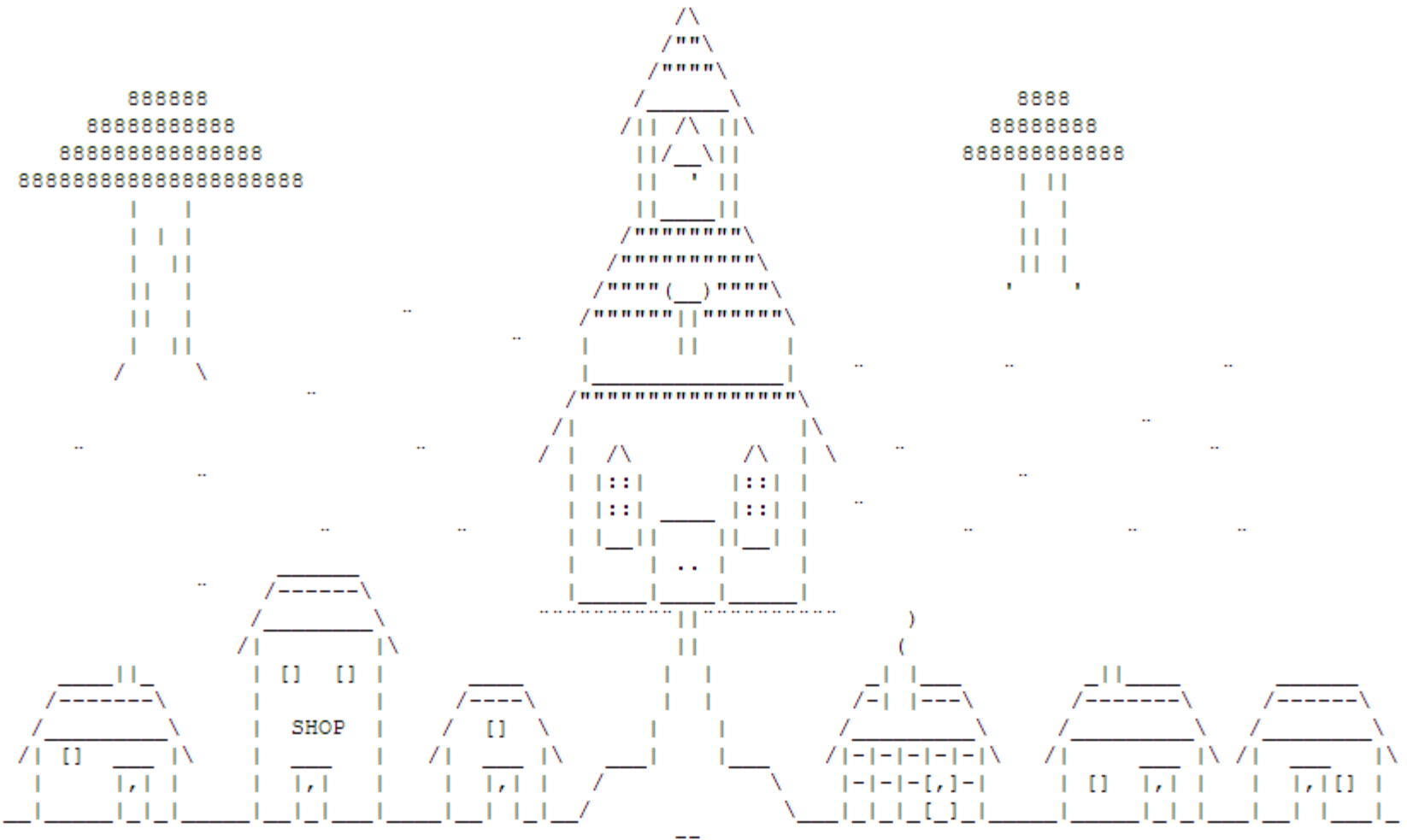
Работа с консолью



Candy Box 2

candybox2.net

```
+-------------------------------------------------------------------------------------------------+
| You have 49 candies |  THE  |      |  MAP  |                                    |      | C |
|                     | CANDY |      |       |                                    | SAVE | F |
|                     |  BOX  |      |       |                                    |      | G |
|                     |                  Your health : 100/100                                    |
+-------------------------------------------------------------------------------------------------+
```



Работа с консолью - «сложные» примеры))

- **Метод Read** читает символ из потока ввода. Он возвращает значение типа int, равное коду прочитанного символа, либо -1 (минус один), если ничего прочитано не было.

Программа показывает на экране введенные символы и их коды:

```
do {
    int i = Console.Read();
    if (i != -1)
        Console.WriteLine("{0} - {1} ", (char)i, i);
    else
        break;
} while (true);
```

- **Метод ReadLine** читает из потока ввода строку текста. Метод возвращает объект типа string или null, если ввод осуществить не удалось.

```
do {
    string s = Console.ReadLine();
    if (s != null)
        Console.WriteLine("Введенная строка: " + s);
    else
        break;
} while (true);
```

Работа с консолью

- **Метод Write** выводит на экран значение переданной ему переменной. Он определен для всех базовых типов и поддерживает форматированные строки. Таким образом, можно либо вызвать Write с указанным значением в качестве параметра:

```
Console.Write (1);
Console.Write (0.745);
Console.Write("Hello!");
```

либо передать строку форматирования и список значений. В строке форматирования применяется множество модификаторов. Здесь мы отметим лишь то, что вместо {n} подставляется n-й входной параметр (нумерация начинается с 0):

```
Console.Write("Привет, {0}! ", Name);
```

- **Метод WriteLine** отличается от Write только тем, что выводит символ перевода строки в конце.

Работа с консолью

- Напишем программу, которая будет осуществлять ввод данных от пользователя, обрабатывать их и выводить на экран.

```
        static void Main(string[] args)
        {
            //объявляем переменную для хранения строки введенных данных
            string strText;
            //выводим на экран информационное сообщение
            Console.WriteLine("Введите Ваше имя.");
            //вводим данные с консоли
            strText = Console.ReadLine();
            //Выводим на экран обработанные данные
            Console.WriteLine("Здравствуйте {0}", strText);


        }
```

Оформление исходного текста программы

- **Кодовый блок {…}** представляет собой набор логически связанных операторов, заключенных в фигурные скобки. Блок не оканчивается точкой с запятой, поскольку он состоит из группы операторов. Вместо этого окончание кодового блока обозначается закрывающей фигурной скобкой.
- **;** Точка с запятой обозначает окончание строки.
- Например, строки

```
x = y;
y = y + 1;
Console.WriteLine(x + " " + y);
```

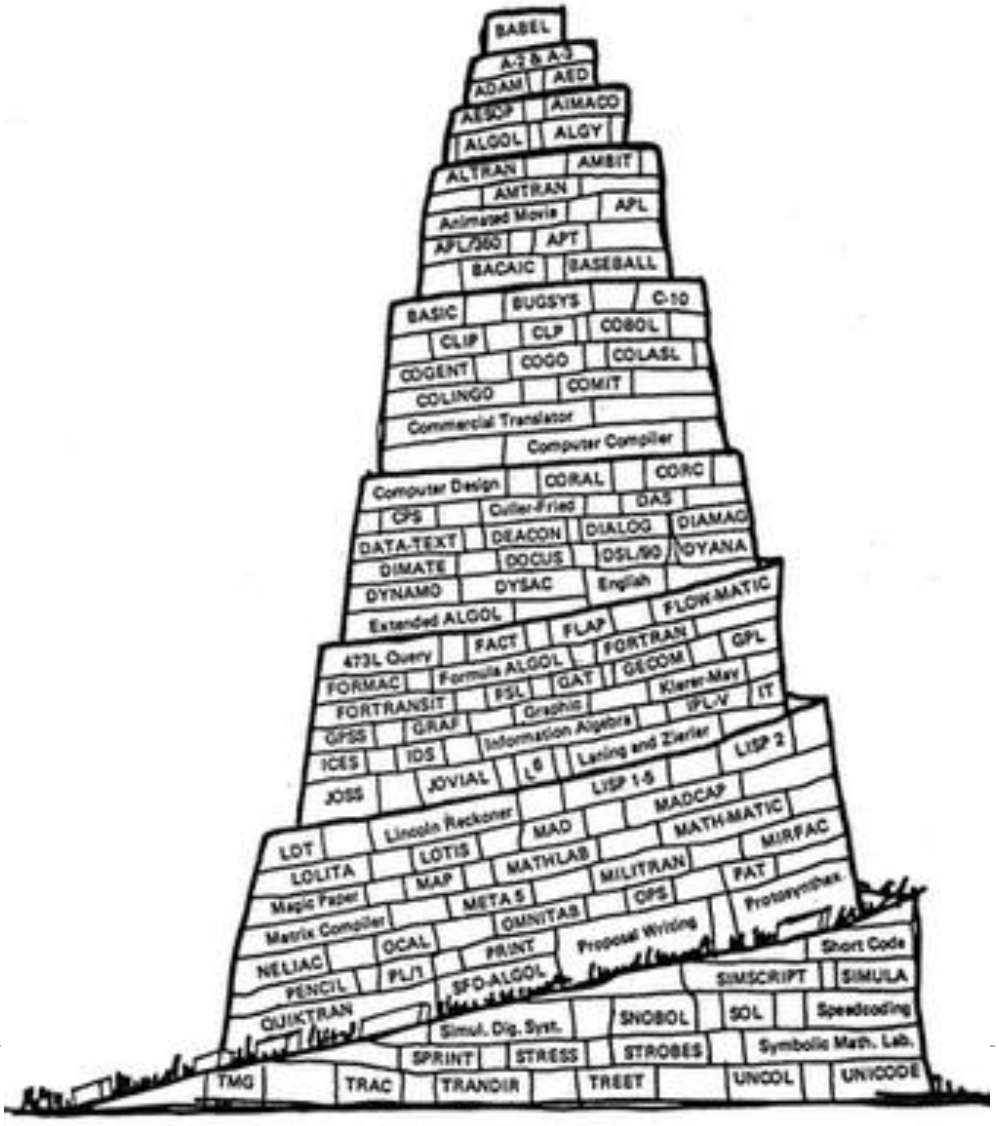
означают то же самое, что и строка кода

```
x = y; y = y + 1; Console.WriteLine(x + " " + y);
```

- Допускается перенос по строкам; следующий фрагмент кода считается в C# вполне допустимым:

```
Console.WriteLine("Это длинная строка вывода" + x + y + z +
"дополнительный вывод");
```

Ключевые слова C#



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			

Контекстные ключевые слова C#

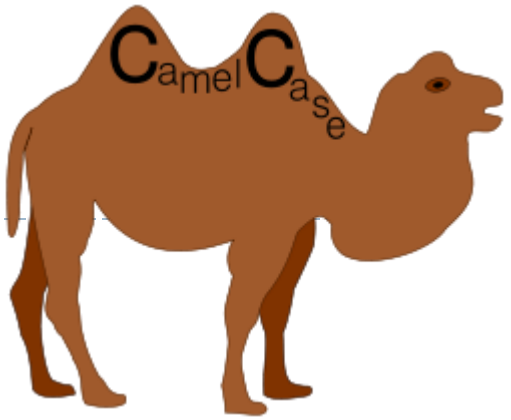
add	dynamic	from	get	global
group	into	join	let	orderby
partial	remove	select	set	value
var	where	yield		

- ***Ключевые слова*** — идентификаторы, имеющие специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены.
- ***Знак операции*** — один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются.
  - Например, сложение +, деление /, сложное присваивание +=.
- Операции делятся на ***унарные*** (с одним операндом, например x++), ***бинарные*** (с двумя, например a+b) и **тернарную** (с тремя, например a >= 0 ? a : -a).
- ***Разделители*** используются для разделения или, наоборот, группирования элементов. Примеры разделителей: скобки, точка, запятая.

Идентификаторы

- имя должно начинаться с буквы или _;
- имя должно содержать только буквы, знак подчеркивания и цифры;
- прописные и строчные буквы различаются;
- длина имени практически не ограничена.
- имена не должны совпадать с ключевыми словами, однако допускается: @if, @float…
- в именах можно использовать управляющие последовательности Unicode
- Примеры допустимых идентификаторов:
  - Test, up, X, top , У2, my, var, MaxLoad, sample23, _13, \u00F2\u01DD, @while
- Примеры неправильных имен:
  - 2late, Big gig, Б#г

Нотации



Понятные и согласованные между собой имена — основа хорошего стиля. Существует несколько *нотаций* — соглашений о правилах создания имен.

В C# для именования различных видов программных объектов чаще всего используются две нотации:

- *UpperCamelNotation (Нотация Паскаля)* - каждое слово начинается с прописной буквы:
  - MaxLength, NewWorldOrder
- *lowerCamelNotation* - с прописной буквы начинается каждое слово, составляющее идентификатор, кроме первого:
  - maxLength, newWorldOrder

Другие типы данных

- int – целочисленный (например: 100);
- float, double – с плавающей точкой (например: 100.1);
- string – строковый (например: “привет!”) .

Пример программы, демонстрирующей отличия между типами int и double:

```
static void Main(string[] args)
{
    int ivar; // объявить целочисленную переменную
    double dvar; // объявить переменную с плавающей точкой
    ivar = 100; // присвоить переменной ivar значение 100
    dvar = 100.0; // присвоить переменной dvar значение 100.0
    Console.WriteLine("Исходное значение ivar: " + ivar);
    Console.WriteLine("Исходное значение dvar: " + dvar);
    Console.WriteLine(); // вывести пустую строку
    // Разделить значения обеих переменных на 3.
    ivar = ivar / 3;
    dvar = dvar / 3.0;
    Console.WriteLine("Значение ivar после деления: " + ivar);
    Console.WriteLine("Значение dvar после деления: " + dvar);
}
```

Результат выполнения программы:

```
Исходное значение ivar: 100
Исходное значение dvar: 100
Значение ivar после деления: 33
Значение dvar после деления: 33,3333333333333
```

Другие типы данных

Пример программы, вычисляющей площадь круга:

```
static void Main(string[] args)
{
    double radius;
    double area;
    radius = 10.0;
    area = radius * radius * 3.1416;
    Console.WriteLine("Площадь равна " + area);
}
```

Результат выполнения программы:

```
Площадь равна 314,16
```

Типы значений в C#

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

Помимо простых типов, в C# определены еще три категории типов значений: перечисления, структуры и обнуляемые типы.

Целочисленные типы – short, int, long и др.

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

Целочисленные типы со знаком по абсолютной величине наполовину меньше своих аналогов без знака. Вот как, например, выглядит число 32 767 типа short в двоичном представлении.

0111111111111111 = 32767

Если установить старший разряд этого числа равным 1, чтобы получить значение со знаком, то оно будет интерпретировано как -1, принимая во внимание формат дополнения до двух:

1111111111111111 = -1

Но если объявить его как значение типа ushort, то после установки в 1 старшего разряда оно станет равным 65 535.

Целочисленные типы

- Пример: Вычисление расстояния от Земли до Солнца в дюймах

```
static void Main(string[] args)
{
  long inches;
  long miles;
  miles = 93000000; // 93 000 000 миль до Солнца
  // 5 280 футов в миле, 12 дюймов в футе,
  inches = miles * 5280 * 12;
  Console.WriteLine("Расстояние до Солнца: " + inches + " дюймов.");
}
```

результат выполнения этой программы:

```
Расстояние до Солнца: 5892480000000 дюймов.
```

Типы для представления чисел с плавающей точкой – float и double

- Разрядность типа float составляет 32 бита, что приближенно соответствует диапазону представления чисел от -3.4E+38 до 3.4E+38. с точностью до 7 знаков.
- А разрядность типа double составляет 64 бита, что приближенно соответствует диапазону представления чисел от ±5.0E-324 до ± 1.79E+308 с точностью до 15 знаков.
- **Пример:** Вычисление радиуса круга

```
static void Main(string[] args)
{
    double r;
    double area;
    area = 10.0;
    r = Math.Sqrt(area / 3.1416);
    Console.WriteLine("Радиус равен " + r);
}
```

Результат выполнения программы :

```
Радиус равен 1,78412203012729
```

Десятичный тип данных - decimal

- Десятичный тип имеет разрядность 128 бит для представления числовых значений в пределах -7.9E+28 до 7.9E+28 с точностью до 28 цифр.
- **Пример:** цена со скидкой рассчитывается на основании исходной цены и скидки в процентах

```
        static void Main(string[] args) {
            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);
        }
```

Результат выполнения программы:

```
Цена со скидкой: $16,9575
```

Символы

- В C# символы представлены не 8-разрядным кодом, как во многих других языках программирования, например C++, а 16-разрядным кодом, который называется уникодом (Unicode).
- В C# определен тип char, представляющий 16-разрядные значения без знака в пределах от 0 до 65 535. При этом стандартный набор символов в 8-разрядном коде ASCII является подмножеством уникода в пределах от 0 до 127. Следовательно, символы в коде ASCII по-прежнему остаются действительными в C#.
- **Пример:** переменной ch присваивается символ X.

```
char ch;
ch = 'X';
```

Значение типа char можно вывести на экран с помощью метода WriteLine ().

```
Console.WriteLine ("Значение ch равно: " + ch);
```

В C# отсутствует автоматическое преобразование символьных значений в целочисленные:

```
char ch;
ch = 88; // ошибка преобразования
```

Логический тип данных

- Тип bool представляет два логических значения: «истина» и «ложь» (true и false).
- Кроме того, в C# **не** определено взаимное преобразование логических и целых значений. Например, 1 не преобразуется в значение true, а 0 не преобразуется в значение false.
- Пример:

```
static void Main(string[] args)
{
    bool b;
    b = false;
    Console.WriteLine("b равно " + b);
    b = true;
    Console.WriteLine("b равно " + b);
    // Логическое значение может управлять оператором if.
    if (b) Console.WriteLine("Выполняется.");
    b = false;
    if (b) Console.WriteLine("Не выполняется.");
    // Результатом выполнения оператора отношения
    // является логическое значение.
    Console.WriteLine("10 > 9 равно " + (10 > 9));
}
```

Эта программа дает следующий результат:

```
b равно False
b равно True
Выполняется.
10 > 9 равно True
```

Переменные

- Переменные объявляются с помощью оператора следующей формы:

```
тип имя_переменной;
```

где тип — это тип данных, хранящихся в переменной; а имя_переменной — это ее имя.

- общая форма инициализации переменной:

```
тип имя_переменной = значение;
```

- примеры инициализации переменных:

```
int count = 10; // задать начальное значение 10 переменной count.
char ch = ‘X’; // инициализировать переменную ch буквенным значением X.
float f = 1.2F; // переменная f инициализируется числовым значением 1,2.
int a, b=8, c =19, d; // инициализировать переменные b и c
```

Переменные

- Пример динамической инициализации (вычисления гипотенузы прямоугольного треугольника по длине его противоположных сторон)

```
static void Main(string[] args) {
    // Длина сторон прямоугольного треугольника.
    double s1 = 4.0;
    double s2 = 5.0;
    // Инициализировать переменную hypot динамически,
    double hypot = Math.Sqrt((s1 * s1) + (s2 * s2));
    Console.Write("Гипотенуза треугольника со сторонами " +
    s1 + " и " + s2 + " равна ");
    Console.WriteLine("{0:#.###}.", hypot);
}
```

- Результат выполнения программы:

```
Гипотенуза треугольника со сторонами 4 и 5 равна 6,403
```

Переменные

- Начиная с версии C# 3.0, компилятору предоставляется возможность самому определить тип локальной переменной, исходя из значения, которым она инициализируется. Такая переменная называется **неявно типизированной,** объявляется с помощью ключевого слова var и должна быть непременно инициализирована.
- **Пример:**

```
var a = 2.7183; //double
var e = 2.7183F; //float
e = 12.2M; // Ошибка! e – уже инициализирована типом float
var s1 = 4.0, s2 = 5.0; // Ошибка! Можно инициализировать только одну переменную
```

Преобразование и приведение типов

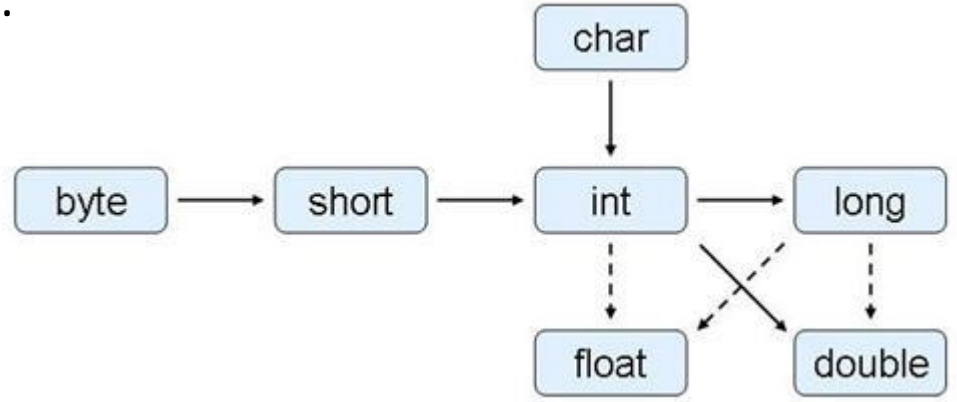
- Пример:

```
int i; float f;
i = 10; f = i; // присвоить целое значение переменной типа float
```

- Неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы;
- диапазон представления чисел целевого типа шире, чем у исходного типа.

не допускается неявное взаимное преобразование типов decimal и float или double, а также числовых типов char или bool. Кроме того, типы char и bool несовместимы друг с другом.

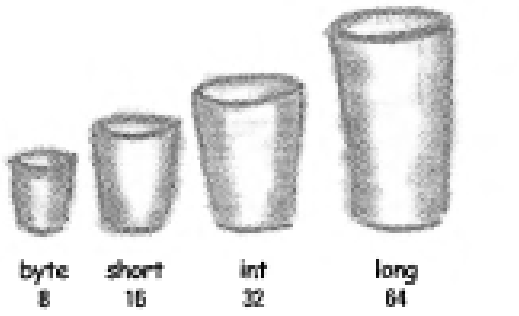


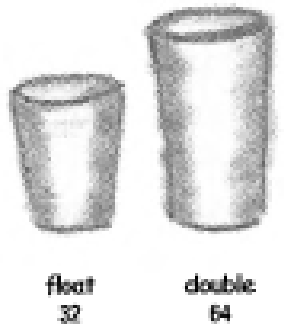
Преобразование и приведение типов

- Пример:

```
long L;
double D;
L = 100123285L;
D = L; //неявное преобразование
Console.WriteLine("L и D: " + L +
" " + D) ;
```

```
long L;
double D;
D = 100123285.0;
L = D; // Недопустимо!!!
Console.WriteLine("L и D: " + L
+ " " + D) ;
```





Приведение несовместимых типов

- общая форма приведения типов.

```
(целевой_тип) выражение;
```

здесь целевой_тип обозначает тот тип, в который желательно преобразовать указанное выражение.

- Рассмотрим для примера следующее объявление переменных.

```
double x, y;
x=3.0; y=2.0;
```

Если результат вычисления выражения x/y должен быть типа int, то следует записать следующее.

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

```
double x, y;
byte b; int i; char ch;
x = 10.0; y = 3.0;
// Приведение типа double к типу int, дробная часть числа теряется.
i = (int)(x / y); // 3
Console.WriteLine("Целочисленный результат деления x / y: " + i);

// Приведение типа int к типу byte без потери данных.
i = 255;
b = (byte)i; // 255
Console.WriteLine("b после присваивания 255: " + b + " - без потери данных.");

// Приведение типа int к типу byte с потерей данных.
i = 257;
b = (byte)i; // 1
Console.WriteLine("b после присваивания 257: " + b + " - с потерей данных.");
Console.WriteLine();

// Приведение типа int к типу char,
b = 88; // ASCII-код символа X
ch = (char)b; // X
Console.WriteLine("ch после присваивания 88: " + ch);
```

Д/З

- Установить Visual Studio 2015 Community:

http://www.visualstudio.com/ru-ru/products/visual-studio-community-vs

- Разобрать и воспроизвести примеры программ из 1-й лекции и 1-й лабораторной работы

sharplabz.karbaev.com