Лекция №0. Основы языка C#

Темы:

1.Основные понятия языка. Состав языка. Алфавит и лексемы. Идентификаторы. Ключевые слова. Операторы и разделители. Комментарии

2.Типы данных. Классификация типов. Встроенные типы. Целые и вещественные числа. Значимые и ссылочные типы.

3.Переменные. Константы. Операторы и выражения. Преобразования типов. Побитовые операторы. Логические операторы. Сокращенная запись арифметических операторов.

4.Ввод-вывод. Класс Console. Методы Write, WriteLine. Форматированный вывод. Метод ReadLine и приведение к типу.

5.Блоки кода. Области видимости. Управление ходом программы. Условный оператор. Оператор выбора

6.Циклы. Цикл с постусловием. Цикл с предусловием. Цикл for. Ключевые слова break, pause continue.

7.Функции. Методы. Параметры методов. Передача параметров по значению, по ссылке. Выходной параметр. Возвращаемое значение.

АЛФАВИТ И ЛЕКСЕМЫ

В C# используется кодировка символов Unicode.

Соответствие между символами и кодирующими их числами называется кодировкой, или кодовой таблицей (character set).

АЛФАВИТ И ЛЕКСЕМЫ

В C# используется кодировка символов Unicode.

Соответствие между символами и кодирующими их числами называется кодировкой, или кодовой таблицей (character set).

Каждый символ представляется в Windows **одним байтом** (8 бит), поэтому в этой кодировке можно одновременно задать только **256 символов**.

АЛФАВИТ И ЛЕКСЕМЫ

Кодировка **Unicode** позволяет представить символы всех существующих алфавитов одновременно, что коренным образом улучшает переносимость текстов.

Каждому символу соответствует свой уникальный код. Естественно, что при этом для хранения каждого символа **требуется больше памяти**. Первые **128 Unicode-символов** соответствуют первой части кодовой таблицы ANSI.

АЛФАВИТ И ЛЕКСЕМЫ

Алфавит C# включает:

- буквы (латинские и национальных алфавитов) и символ подчеркивания (_), который употребляется наряду с буквами;
- цифры;
- специальные символы, например.+, *, { и &;
- пробельные символы (пробел и символы табуляции);
- символы перевода строки.

АЛФАВИТ И ЛЕКСЕМЫ

ЛЕКСЕМА — это минимальная единица языка, имеющая самостоятельный смысл.

Существуют следующие виды лексем:

- имена (идентификаторы);
- ключевые слова;
- знаки операций;
- разделители;
- литералы (константы).

АЛФАВИТ И ЛЕКСЕМЫ

Директивы препроцессора C# - это команды компилятора, влияющие на процесс компиляции. Эти команды определяют, какие разделы кода компилировать или как обрабатывать конкретные ошибки и предупреждения

АЛФАВИТ И ЛЕКСЕМЫ

Директивы препроцессора пришли в C# из его предшественника — языка C++.

Препроцессором называется предварительная стадия компиляции, на которой формируется окончательный вид исходного текста программы.

Например, с помощью директив (инструкций, команд) препроцессора можно включить или выключить из процесса компиляции фрагменты кода.

АЛФАВИТ И ЛЕКСЕМЫ

Комментарии предназначены для записи пояснений к программе и формирования документации.

Правила записи комментариев будут описаны далее.

АЛФАВИТ И ЛЕКСЕМЫ

Из лексем составляются выражения и операторы. Выражение задает правило вычисления некоторого значения.

Например, выражение a + b задает правило вычисления суммы двух величин.

АЛФАВИТ И ЛЕКСЕМЫ

Оператор задает законченное описание некоторого действия, данных или элемента программы.

Например: int a;

Это — оператор описания целочисленной переменной a.

ИДЕНТИФИКАТОРЫ

ИДЕНТИФИКАТОРЫ

Имена в программах служат той же цели, что и имена в мире людей.

Поэтому имена также называют идентификаторами

ИДЕНТИФИКАТОРЫ

Первым символом идентификатора может быть буква или знак подчеркивания, но НЕ ЦИФРА. Длина идентификатора не ограничена. ПРОБЕЛЫ внутри имен НЕ ДОПУСКАЮТСЯ.

ИДЕНТИФИКАТОРЫ

В идентификаторах C# разрешается ИСПОЛЬЗОВАТЬ помимо латинских букв буквы НАЦИОНАЛЬНЫХ АЛФАВИТОВ.

Например, Пёсик

ИДЕНТИФИКАТОРЫ

Более того, в идентификаторах можно применять даже так называемые escape-последовательности Unicode

\u, например, \u00F2

ИДЕНТИФИКАТОРЫ

Имена даются элементам программы, к которым требуется обращаться: переменным, типам, константам, методам, меткам и т.д

При выборе идентификатора необходимо иметь в виду следующее:

1) идентификатор НЕ ДОЛЖЕН совпадать с ключевыми словами;

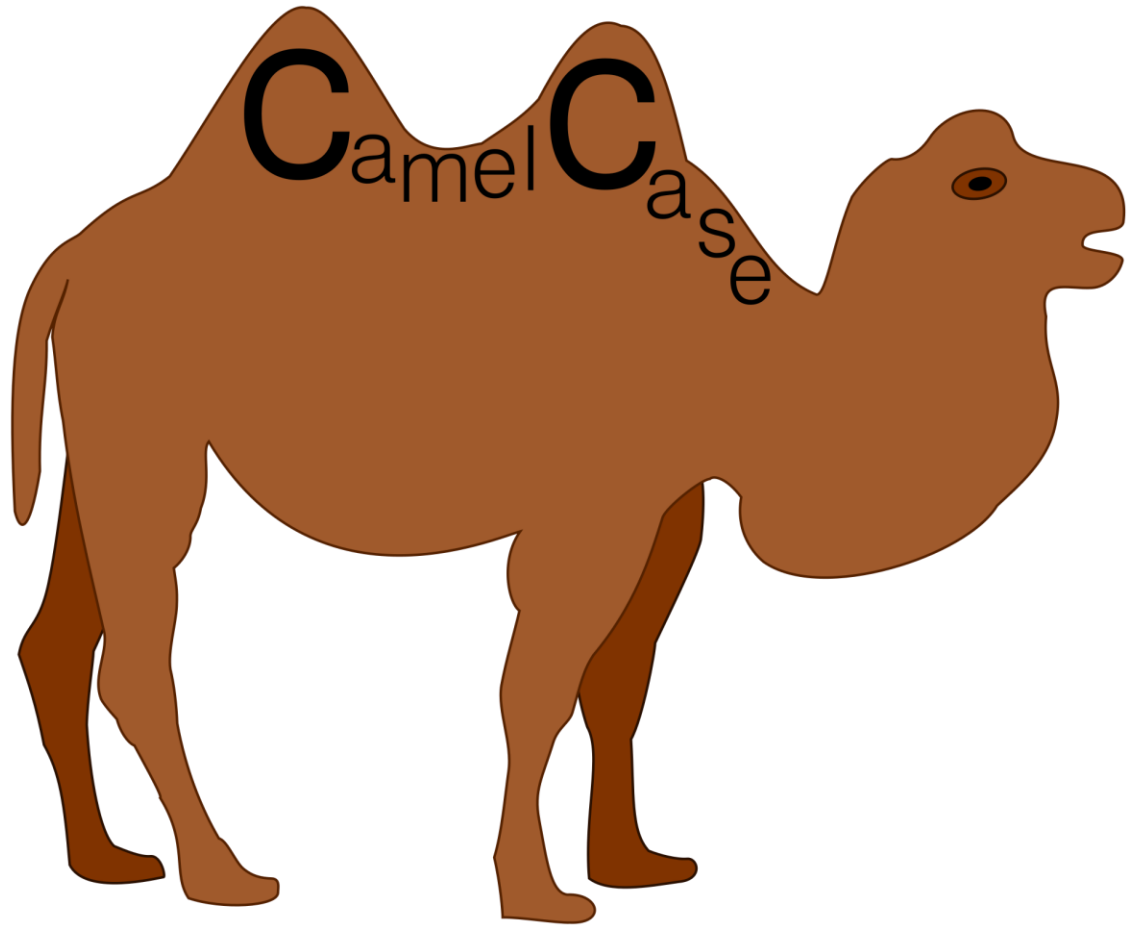
2) не рекомендуется начинать идентификаторы с двух символов подчеркивания, поскольку такие имена зарезервированы для служебного использования;

ИДЕНТИФИКАТОРЫ

Для улучшения читабельности программы следует давать объектам ОСМЫСЛЕННЫЕ ИМЕНА, составленные в соответствии с определенными правилами.

ИДЕНТИФИКАТОРЫ

- В НОТАЦИИ ПАСКАЛЯ каждое слово, составляющее идентификатор, начинается с прописной буквы, например, MaxLength, MyFuzzyShooshpanchik.
- Венгерская нотация отличается от предыдущей наличием префикса, соответствующего типу величины, например, iMaxLength, lpfnMyFuzzyShooshpanchik.
- Согласно НОТАЦИИ CAMEL, с прописной буквы начинается каждое слово, составляющее идентификатор, кроме первого, например, maxLength, myFuzzyShooshpanchik. Человеку с богатой фантазией абрис имени может напоминать верблюда, откуда и произошло название этой нотации.
- Еще одна традиция — разделять слова, составляющие имя, ЗНАКАМИ ПОДЧЕРКИВАНИЯ: maxjength, my_fuzzy_shooshpanchik, при этом все составные части начинаются со строчной буквы.


CamelCase

КЛЮЧЕВЫЕ СЛОВА

- **КЛЮЧЕВЫЕ СЛОВА — это зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены. Список ключевых слов C# приведен в таблице которую увидите дальше**

КЛЮЧЕВЫЕ СЛОВА

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	void
volatile	while			

Знаки операций и разделители

ЗНАК ОПЕРАЦИИ — это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются.
Например, в выражении a += b знак += является знаком операции, а a и b — операндами.

Символы, составляющие знак операций, могут быть как специальными, например, &&, | и <, так и буквенными, такими как as или new.

Знаки операций и разделители

Операции делятся на УНАРНЫЕ, БИНАРНЫЕ и ТЕРНАРНУЮ по количеству участвующих в них операндов.

Один и тот же знак может интерпретироваться по-разному в зависимости от контекста.

Все знаки операций, за исключением [ ], ( ) и ? :, представляют собой отдельные лексемы.

Знаки операций и разделители

Разделители используются для разделения или, наоборот, группирования элементов. Примеры разделителей: скобки, точка, запятая. Ниже перечислены все знаки операций и разделители, использующиеся в C#:

{} [] () . , : ; + - * / ‘ * & | A ! ~ = <> ? - ++ --
&& || << >> == != <= >= += -= *= /= %= &=
|= ^= <<= >>= ->

ЛИТЕРАЛЫ

Литералами, или константами, называют неизменяемые величины.

В C# есть логические, целые, вещественные, символьные и строковые константы, а также константа null

Описание и примеры констант каждого типа

Константа	Описание	Примеры
Логическая	**true** (истина) или **false** (ложь)	true false
Целая	Десятичная: последовательность десятичных цифр (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), за которой может следовать суффикс (U, u, L, 1, UL, Ul, uL, ul, LU, Lu, 1U, lu) Шестнадцатеричная: символы Ox или OX, за которыми следуют шестнадцатеричные цифры (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F), а за цифрами, в свою очередь, может следовать суффикс (U, u, L, 1, UL, Ul, uL, ul, LU, Lu, Ш, lu)	8 0 199226 8u OLu 199226L 0xA 0xlB8 0X00FF 0xAU 0x1B8LU 0X00FFl

Описание и примеры констант каждого типа

Константа	Описание	Примеры
Вещественная	С фиксированной точкой: [цифры] [.] [цифры] [суффикс] Суффикс — один из символов F, f, D, d, M, m С порядком: [цифры][.][цифры]{E|e}[+|-] [цифры] [суффикс] Суффикс — один из символов F, f, D, d, M, m	5.7 .001 35 5.7F .00ld 35 5F .00lf 35m
Символьная	Символ, заключенный в апострофы	'A' 'ю' '\0' '\n’ ' \xF' '\x74' '\uA81B'

Описание и примеры констант каждого типа

Константа	Описание	Примеры
Строковая	Последовательность символов, заключенная в кавычки	"Здесь был Vasia" "\t3начение г = \xF5 \n" "Здесь был \u0056\u0061" "C:\\temp\\file1.txt" @"C:\temp\file1.txt"
Константа null	Ссылка, которая не указывает ни на какой объект	null

Комментарии

В C# есть два вида комментариев: ОДНОСТРОЧНЫЕ И МНОГОСТРОЧНЫЕ.

Однострочный комментарий начинается с двух символов прямой косой черты (//) и заканчивается символом перехода на новую строку, многострочный заключается между символами-скобками /* и */ и может занимать часть строки, целую строку или несколько строк.

Комментарии не вкладываются друг в друга: символы // и /* не обладают никаким специальным значением внутри комментария.

Переменные

ПЕРЕМЕННАЯ — это именованная область памяти, предназначенная для хранения данных определенного типа.

Во время выполнения программы значение переменной можно изменять.

Все переменные, используемые в программе, должны быть описаны явным образом.

При описании для каждой переменной задаются ее имя и тип.

```
int a;
float x;
```

Переменные

ПЕРЕМЕННАЯ — это именованная область памяти, предназначенная для хранения данных определенного типа.

Во время выполнения программы значение переменной можно изменять.

Все переменные, используемые в программе, должны быть описаны явным образом.

При описании для каждой переменной задаются ее имя и тип.

```
int a;
float x;
```

Именованные константы

Можно ЗАПРЕТИТЬ ИЗМЕНЯТЬ ЗНАЧЕНИЕ ПЕРЕМЕННОЙ, задав при ее описании ключевое слово const, например:

```
const int b = 1;
const float x = 0.1, y = 0.1f; // const распространяется на обе переменные
```

Именованные константы

```
const int b =.1, a = 100:
const int x = b * a + 25;
```

Операции и выражения

Выражение — это правило вычисления значения. В выражении участвуют операнды, объединенные знаками операций. Операндами простейшего выражения могут быть константы, переменные и вызовы функций.

Операции и выражения

Например, a + 2 — это выражение, в котором + является знаком операции, а
a и 2 — операндами.

Пробелы внутри знака операции, состоящей из нескольких символов, не допускаются.

По количеству участвующих в одной операции операндов операции делятся на унарные, бинарные и тернарную. (25 слайд)

Преобразования встроенных арифметических типов-значений

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 ПРАВИЛАМ, обеспечивающим приведение более коротких типов к более длинным для сохранения значимости и точности.

Преобразования встроенных арифметических типов-значений

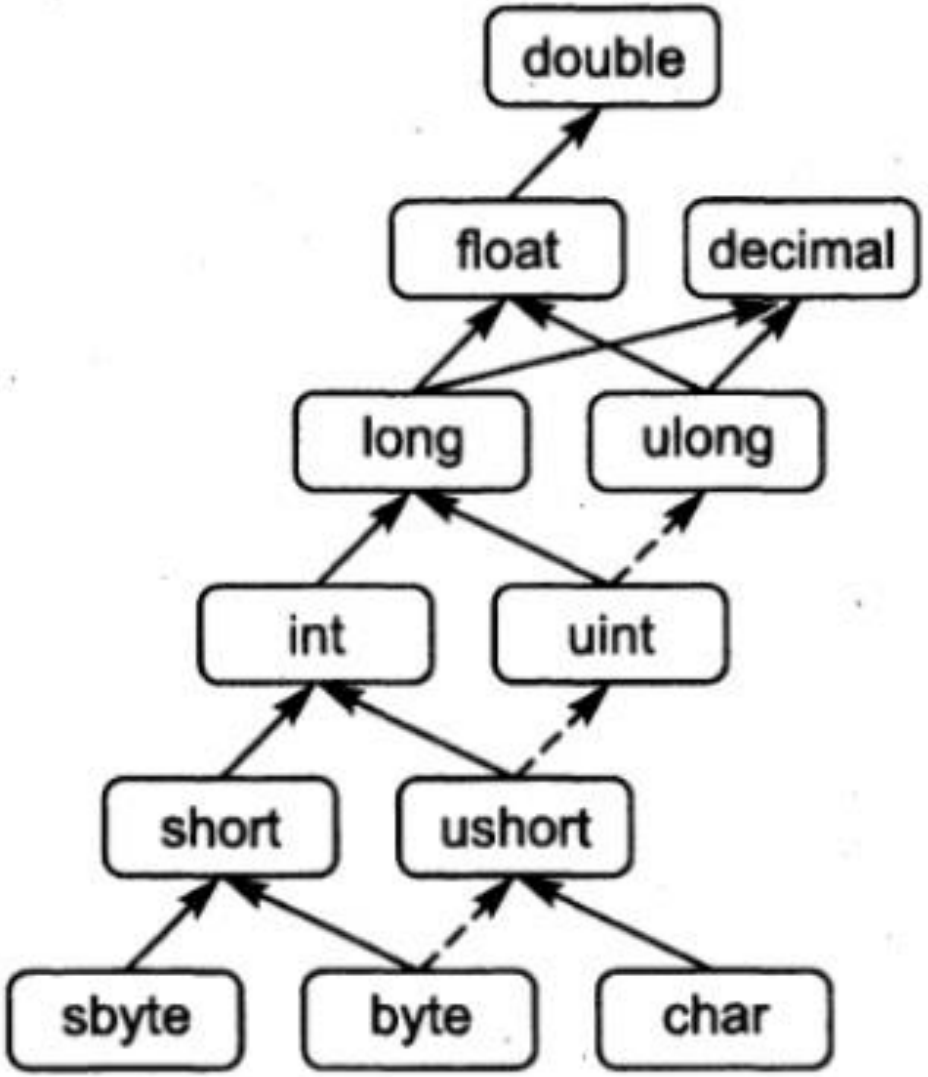
Автоматическое (НЕЯВНОЕ) ПРЕОБРАЗОВАНИЕ возможно не всегда, а только если при этом не может случиться потеря значимости.

Преобразования встроенных арифметических типов-значений

Если неявного преобразования из одного типа в другой не существует, программист может задать ЯВНОЕ ПРЕОБРАЗОВАНИЕ типа с помощью ОПЕРАЦИИ (ТИП)Х. Его результат остается на совести программиста.

Например: F = x – int(b) (где b = 5)

Правила неявного преобразования



ЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПА

Операция используется, как и следует из ее названия, для явного преобразования величины из одного типа в другой.

При преобразовании из более длинного типа в более короткий возможна потеря информации, если исходное значение выходит за пределы диапазона результирующего типа.

ЯВНОЕ ПРЕОБРАЗОВАНИЕ ТИПА

Формат операции: (тип)выражение

Здесь тип — это имя того типа, в который осуществляется преобразование, а выражение чаще всего представляет собой имя переменной, например:

```
long b = 300;
int a = (int) b; // данные не теряются
byte d = (byte) a; // данные теряются
```

ПРОСТЕЙШИЙ ВВОД-ВЫВОД

Любая программа при вводе исходных данных и выводе результатов взаимодействует с внешними устройствами. Совокупность стандартных устройств ввода и вывода, то есть клавиатуры и экрана, называется КОНСОЛЬЮ.

Standaardtypes

gehele getallen

1 byte	**sbyte**	± 127	**byte**	0..255	2^8-1
2 bytes	**short**	± 32767	**ushort**	0..65535	2^{16}-1
4 bytes	**int**	± 2 miljard	**uint**	0..4 miljard	2^{32}-1
8 bytes	**long**	± 10^{18}	**ulong**	0..10^{19}	2^{64}-1

met een decimale punt

4 bytes	**float**	7 cijfers, afgerond, ≤10^{38}
8 bytes	**double**	15 cijfers, afgerond, ≤10^{300}
16 bytes	**decimal**	28 cijfers, exact, ≤10^{28}

Keyword	Class	Range
bool	System.Boolean	true and false
byte	System.Byte	0 to 255
sbyte	System.SByte	-128 to 127
short	System.Int16	-32768 to 32767
ushort	System.Uint16	0 to 65535
int	System.Int32	-2,147,483,648 to 2,147,483,647
uint	System.UInt32	0 to 4,294,967,295
long	System.Int64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	System.UInt64	0 to 18,446,744,073,709,551,615
decimal	System.Decimal	-79,228,162,514,264,337,593,543,950,335 to 79,228,162,514,264,337,593,543,950,335
double	System.Double	-1.79769313486232e308 to 1.79769313486232e308
float	System.Single	-3.402823e38 to 3.402823e38
char	System.Char	0 to 65535

ПРОСТЕЙШИЙ ВВОД-ВЫВОД

Любая программа при вводе исходных данных и выводе результатов взаимодействует с внешними устройствами. Совокупность стандартных устройств ввода и вывода, то есть клавиатуры и экрана, называется КОНСОЛЬЮ

ПРОСТЕЙШИЙ ВВОД-ВЫВОД

Методы с одинаковыми именами, но разными параметрами называются перегруженными

ПРОСТЕЙШИЙ ВВОД-ВЫВОД

```
using System;
namespace ConsoleApplication1 {
   class Class1 {
   static void Main() {
      int i = 3;
      double y = 4.12;
      decimal d = 600m;
      string s = "Вася";
      Console.WriteLine("i = " + i );              //1
      Console.WriteLine("y = {0} \nd = {1}”, y, d); //2
      Console.WriteLine("s = " + s );              //3
   }
   }
}
```

ПРОСТЕЙШИЙ ВВОД-ВЫВОД

```
Console.WriteLine( "i = " + i.ToString() );
```

ПРОСТЕЙШИЙ ВВОД-ВЫВОД

Ввод числовых данных выполняется в **два этапа**:

1. Символы, представляющие собой число, вводятся с клавиатуры в строковую переменную.
2. Выполняется преобразование из строки в переменную соответствующего типа. Преобразование можно выполнить либо с помощью **специального класса Convert**, определенного в пространстве **имен System**, либо с помощью метода Parse, имеющегося в каждом стандартном арифметическом классе. В листинге далее используются оба способа.

```
using System;
namespace ConsoleApplication1
{ class Classl
   { static void Main() {
      Console.WriteLine( "Введите строку" );
      string s = Console.ReadLine();       // 1
      Console.WriteLine( "s = " + s );

      Console.WriteLine( "Введите символ" );
      char c = (char)Console.Read();           // 2
      Console.ReadLine();               // 3

      Console.WriteLine("c = " + c );
      string buf; //строка-буфер для ввода чисел
      Console.WriteLine("Введите целое число" );
      buf = Console.ReadLine();
      int i = Convert.ToInt32( buf );  // 4
      Console.WriteLine(i);
      Console.WriteLine("Введите вещественное число" );
      buf = Console.ReadLine();
      double x = Convert.ToDouble( buf ); // 5
      Console.WriteLine( x );
      Console.WriteLine( "Введите вещественное число" );
      buf = Console.ReadLine();
      double y = double.Parse( but ); // 6
      Console.WriteLine( y );
      Console.WriteLine( "Введите вещественное число" );
      buf = Console.ReadLine();
      decimal z = decimal.Parse( buf ); // 7
      Console.WriteLine( z );
      }
   }
}
```

ВЫРАЖЕНИЯ, БЛОКИ И ПУСТЫЕ ОПЕРАТОРЫ

Любое выражение, завершающееся **точкой с запятой**, рассматривается как **оператор**, выполнение которого заключается в вычислении выражения. Частным случаем выражения является пустой оператор ;

```
i++; // выполняется операция инкремента
a *= b + c; // выполняется умножение с присваиванием (a*(b+c))
fun( i, k ); // выполняется вызов функции
while( true ); // цикл из пустого оператора (бесконечный)
```

ВЫРАЖЕНИЯ, БЛОКИ И ПУСТЫЕ ОПЕРАТОРЫ

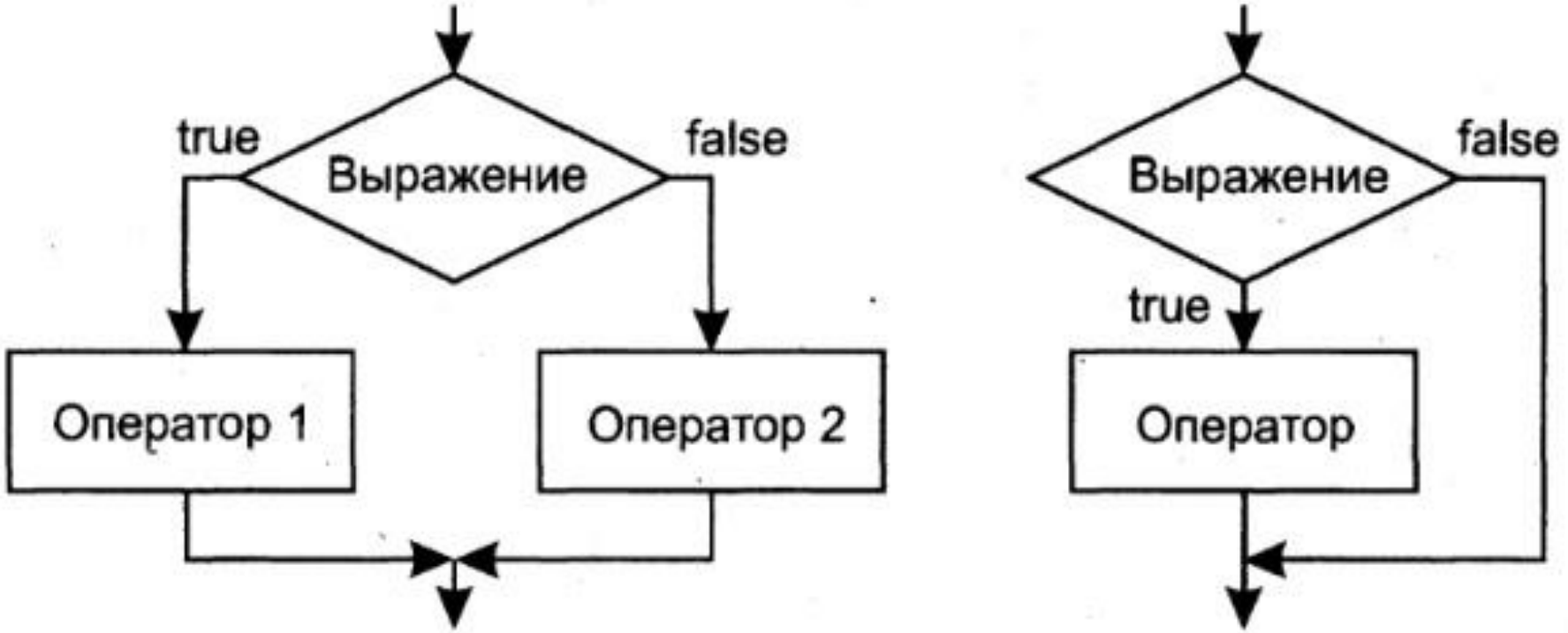
Любое выражение, завершающееся **точкой с запятой**, рассматривается как **оператор**, выполнение которого заключается в вычислении выражения. Частным случаем выражения является пустой оператор ;

```
i++; // выполняется операция инкремента
a *= b + c; // выполняется умножение с присваиванием (a*(b+c))
fun( i, k ); // выполняется вызов функции
while( true ); // цикл из пустого оператора (бесконечный)
```

ОПЕРАТОРЫ ВЕТВЛЕНИЯ

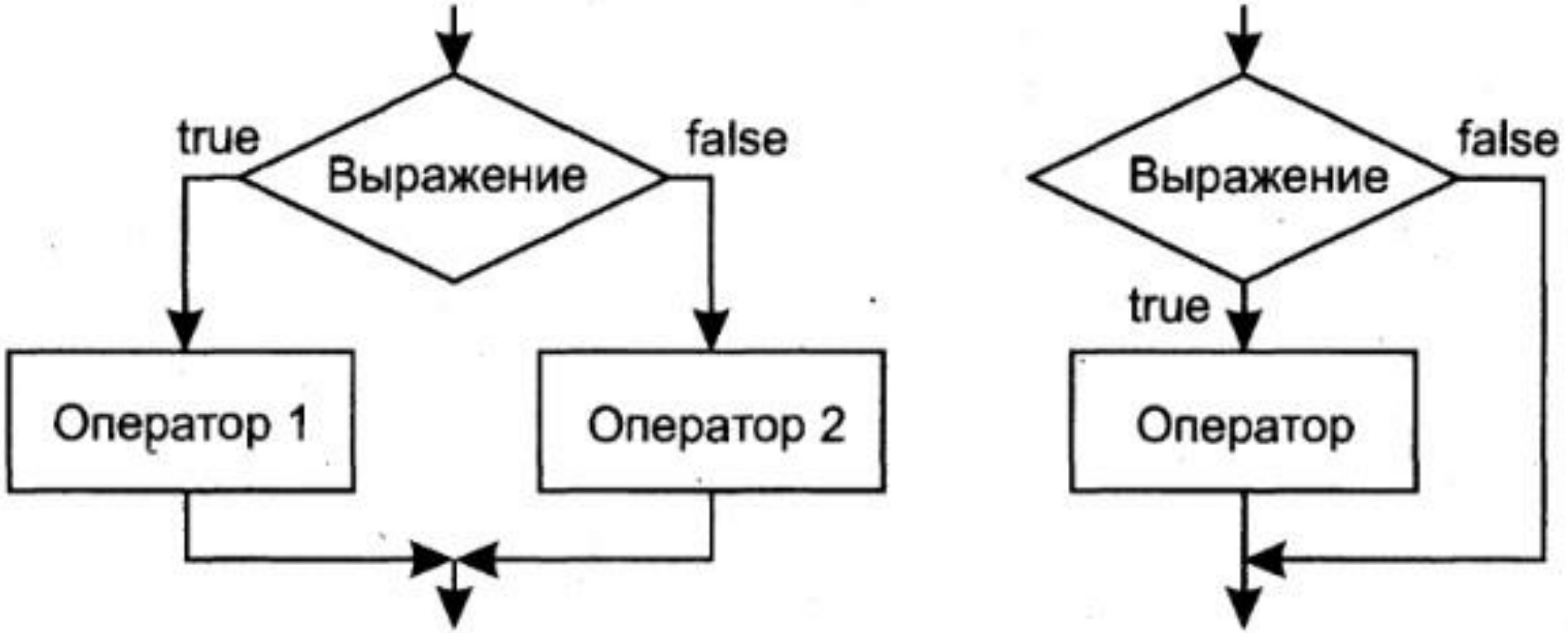
Операторы ветвления **if** и **switch** применяются для того чтобы в зависимости от конкретных значений исходных данных обеспечить выполнение разных последовательностей операторов. **Оператор if** обеспечивает передачу управления на одну из двух ветвей вычислений, а **оператор switch** — на одну из произвольного числа ветвей.

УСЛОВНЫЙ ОПЕРАТОР IF



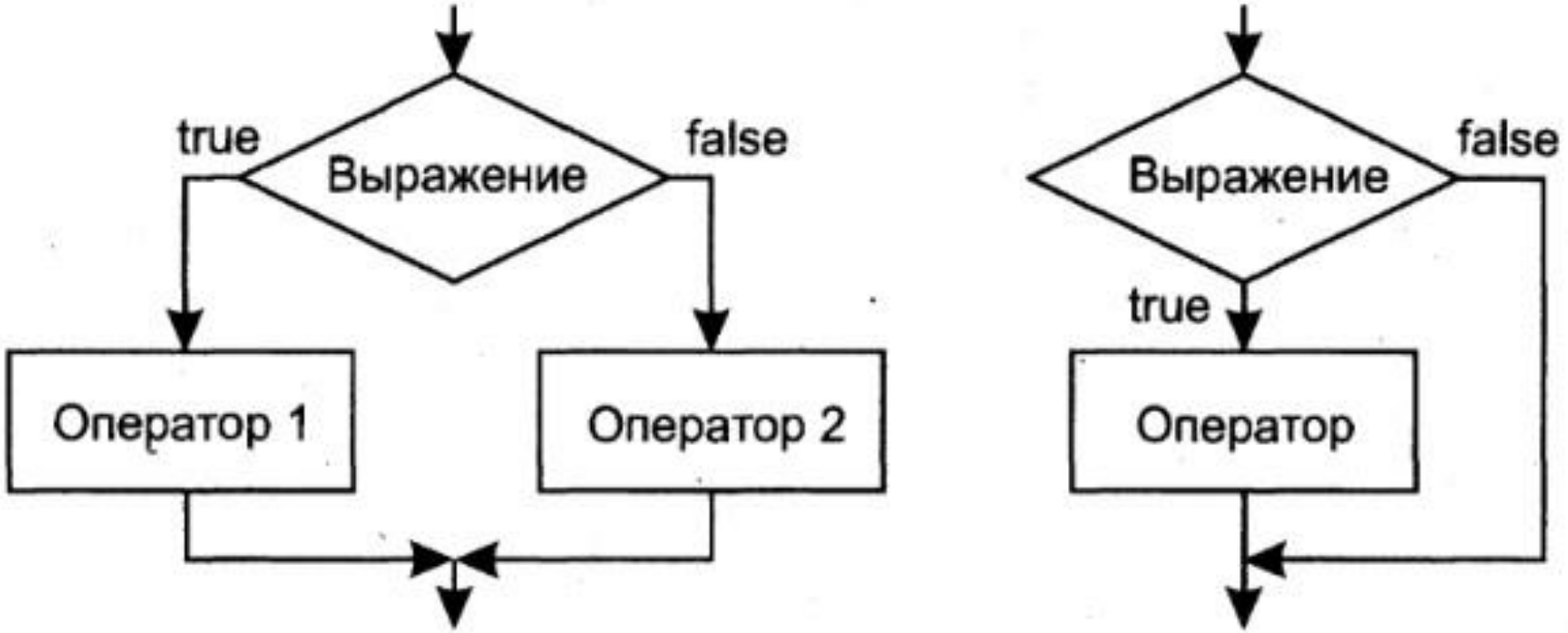
if (логическое_выражение) оператор1; [else оператор2;]

УСЛОВНЫЙ ОПЕРАТОР IF



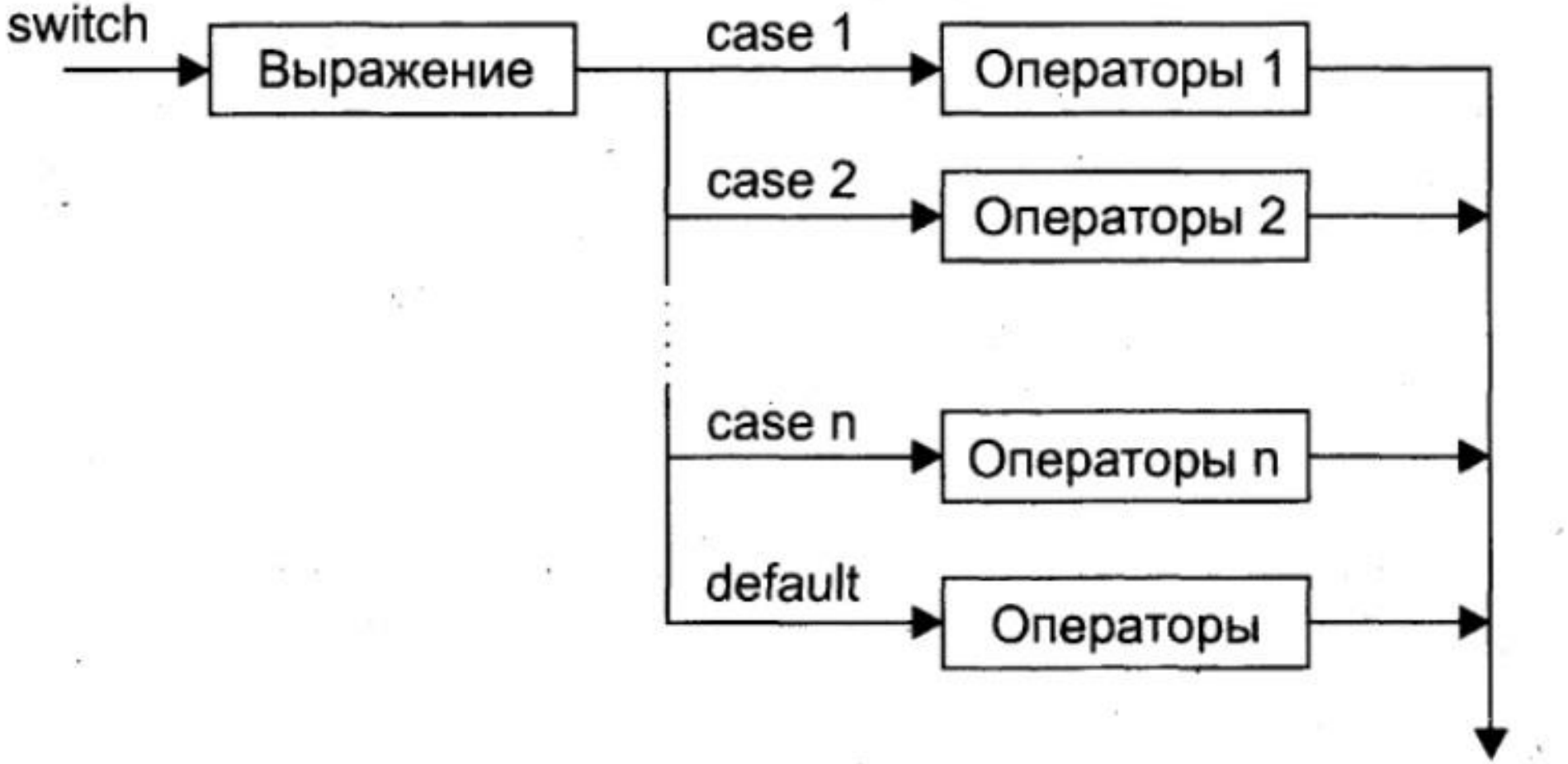
if (логическое_выражение) оператор1; [else оператор2;]

УСЛОВНЫЙ ОПЕРАТОР IF



if (логическое_выражение) оператор1; [else оператор2;]

ОПЕРАТОР ВЫБОРА SWITCH



ОПЕРАТОР ВЫБОРА SWITCH

```
switch ( выражение ){
  case константное_выражение_1: [ список_операторов_1 ]
  case константное_выражение_2: [ список_операторов_2 ]
  case константное_выражение_n: [ список_операторов_n ]
  [ default: операторы ]
}
```

ОПЕРАТОР ВЫБОРА SWITCH

```
char character_variable = 'B';

switch (character_variable)
{
    case 'A':
        Console.WriteLine("Variable has value A");
        break;
    case 'B':
        Console.WriteLine("Variable has value B");
        break;
    case 'C':
        Console.WriteLine("Variable has value C");
        break;

    default:
        Console.WriteLine("The variable not have the chartectors A,B and C");
        break;
}
Console.WriteLine("The variable has value {0}", character_variable);
Console.ReadLine();
```

ОПЕРАТОР ВЫБОРА SWITCH

```
char character_variable = 'B';

switch (character_variable)
{
    case 'A':
        Console.WriteLine("Variable has value A");
        break;
    case 'B':
        Console.WriteLine("Variable has value B");
        break;
    case 'C':
        Console.WriteLine("Variable has value C");
        break;

    default:
        Console.WriteLine("The variable not have the chartectors A,B and C");
        break;
}
Console.WriteLine("The variable has value {0}", character_variable);
Console.ReadLine();
```

ОПЕРАТОР ВЫБОРА SWITCH

```
char character_variable = 'B';

switch (character_variable)
{
    case 'A':
        Console.WriteLine("Variable has value A");
        break;
    case 'B':
        Console.WriteLine("Variable has value B");
        break;
    case 'C':
        Console.WriteLine("Variable has value C");
        break;

    default:
        Console.WriteLine("The variable not have the chartectors A,B and C");
        break;
}
Console.WriteLine("The variable has value {0}", character_variable);
Console.ReadLine();
```

ОПЕРАТОР ВЫБОРА SWITCH

```
char character_variable = 'B';

switch (character_variable)
{
    case 'A':
        Console.WriteLine("Variable has value A");
        break;
    case 'B':
        Console.WriteLine("Variable has value B");
        break;
    case 'C':
        Console.WriteLine("Variable has value C");
        break;

    default:
        Console.WriteLine("The variable not have the chartectors A,B and C");
        break;
}
Console.WriteLine("The variable has value {0}", character_variable);
Console.ReadLine();
```

ОПЕРАТОРЫ ЦИКЛА

Операторы цикла используются для **вычислений**, **повторяющихся многократно**. В **C#** имеется **четыре вида** циклов: цикл с предусловием **while**, цикл с постусловием **repeat**, цикл с параметром **for** и цикл перебора **foreach**. Каждый из них состоит из определенной последовательности операторов.

ОПЕРАТОРЫ ЦИКЛА

Блок, ради выполнения которого и организуется цикл, называется **телом цикла.**

Остальные операторы служат управления процессом повторения: это начальные установки, проверка условия продолжения цикла и модификация параметра цикла.
Один проход цикла называется **итерацией**.

ОПЕРАТОРЫ ЦИКЛА

Начальные установки служат для того, чтобы до входа в цикл задать значения переменных, которые в нем используются.

ОПЕРАТОРЫ ЦИКЛА

Параметром цикла называется п**еременная**, которая **используется** при проверке условия **продолжения цикла** и принудительно изменяется на каждой итерации, причем, как правило, на одну и ту же величину.

Если параметр цикла **целочисленный**, он называется **счетчиком цикла**.


Начальные установки
Выражение
Операторы
Модификация параметра цикла

а


Начальные установки
Операторы
Модификация параметра цикла
Выражение

б

ЦИКЛ С ПРЕДУСЛОВИЕМ WHILE

Формат оператора прост:

while (выражение) оператор

ЦИКЛ С ПОСТУСЛОВИЕМ DO

DO ОПЕРАТОР WHILE ВЫРАЖЕНИЕ;

Цикл завершается, когда выражение станет равным **false** или в теле цикла будет выполнен какой-либо оператор передачи управления.

ЦИКЛ С ПАРАМЕТРОМ FOR

for (инициализация; выражение; модификации) оператор;

Цикл завершается, когда выражение станет равным **false** или в теле цикла будет выполнен какой-либо оператор передачи управления.

```
for ( int i =O, j = 20; ...

int k, m;

for ( k = 1, m = 0; ...
```

ЦИКЛ С ПАРАМЕТРОМ FOR

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла.

```
for ( int i = 0, j = 20; i < 5 && j > 10; i++, j-- ) ...
```

МЕТОДЫ

Метод — это функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или экземпляром.

Методы определяют поведение класса.

МЕТОДЫ

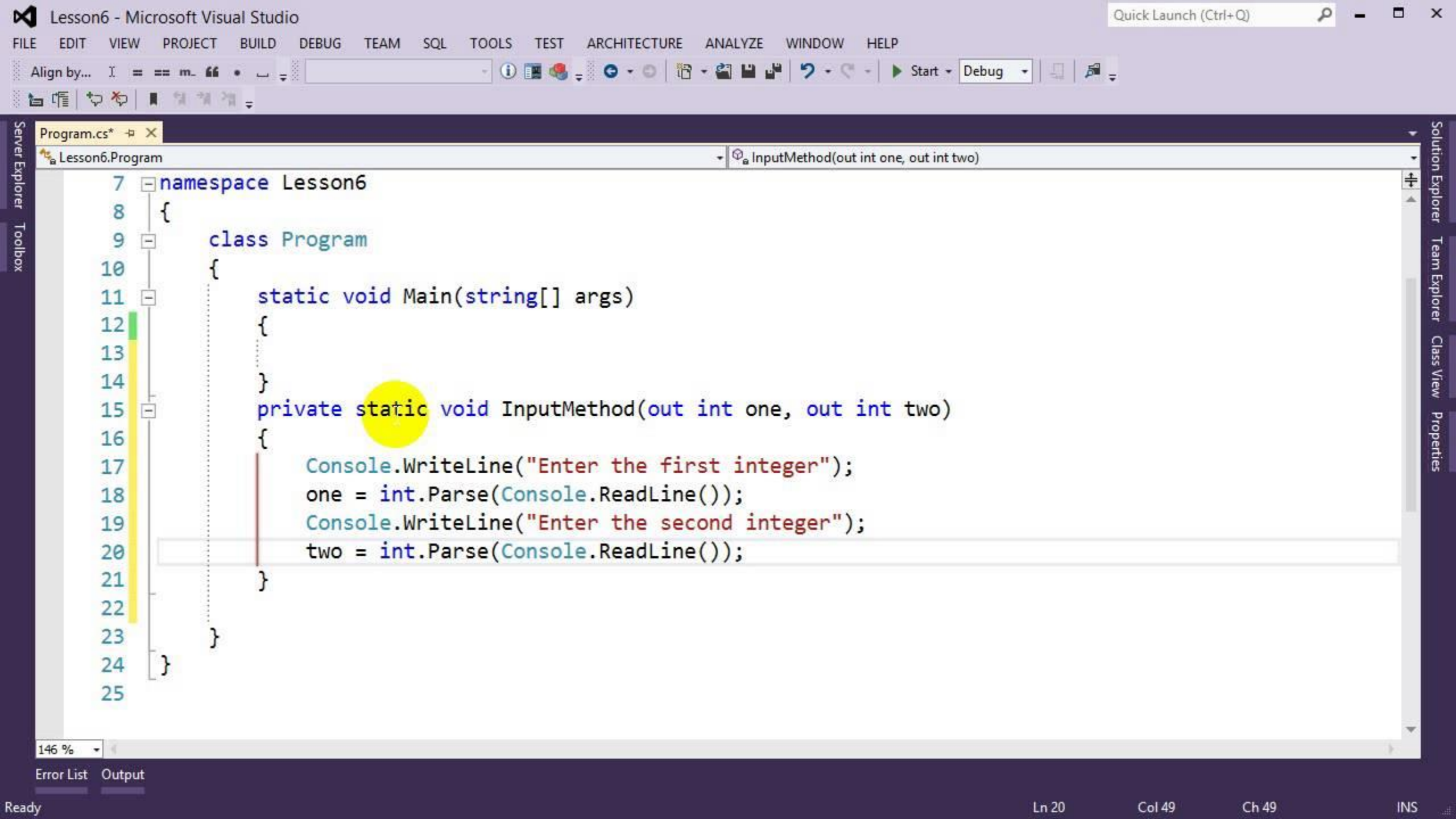
Синтаксис метода:

[атрибуты] [спецификаторы] тип имя_метода ([параметры])

МЕТОДЫ

C# коллекции

Метод	Тип	Описание
CopyTo()	void	Перегружен. Копирует элементы ArrayList в одномерный массив
Equals()	bool	Перегружен. Определяе равны ли объекты между собой
Add()	int	Добавляет новый элемент в конец
GetRange()	ArrayList	Возвращает новый массив ArrayList, содержащий диапазон элементов из оригинального массива
GetType()	Type	Возвращает тип текущего объекта
IndexOf()	int	Пререгружен. Возвращает номер позиции первого вхождения указанного элемента в массив
Insert()	void	Вставляет элемент в указанную позицию


Lesson6 - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Start
Debug
Program.cs*
Lesson6.Program
InputMethod(out int one, out int two)
namespace Lesson6
{
class Program
{
static void Main(string[] args)
{
}
private static void InputMethod(out int one, out int two)
{
Console.WriteLine("Enter the first integer");
one = int.Parse(Console.ReadLine());
Console.WriteLine("Enter the second integer");
two = int.Parse(Console.ReadLine());
}
}
}
146 %
Error List
Output
Ready
Ln 20
Col 49
Ch 49
INS

МЕТОДЫ

Параметры используются для обмена информацией с методом. Параметр представляет собой локальную переменную, которая при вызове метода принимает значение соответствующего аргумента.

Параметры методов

При вызове метода выполняются следующие действия:

1. Вычисляются выражения, стоящие на месте аргументов.
2. Выделяется память под параметры метода в соответствии с их типом.
3. Каждому из параметров сопоставляется соответствующий аргумент (аргументы как бы накладываются на параметры и замещают их).
4. Выполняется тело метода.
5. Если метод возвращает значение, оно передается в точку вызова; если метод имеет тип void, управление передается на оператор, следующий после вызова. При этом проверяется соответствие типов аргументов и параметров и при необходимости выполняется их преобразование. При несоответствии типов выдается диагностическое сообщение.

МЕТОДЫ

Существуют два способа передачи параметров: по значению и по ссылке.

1. **параметры-значения**;
2. **параметры-ссылки** — описываются с помощью ключевого слова ref;
3. **выходные параметры** — описываются с помощью ключевого слова out;
4. **параметры-массивы** — описываются с помощью ключевого слова params.

Параметры-значения

void P(int x)

Параметры-ссылки

```
void P( ref int x )
```

ВЫХОДНЫЕ ПАРАМЕТРЫ

Довольно часто возникает необходимость в методах, которые формируют несколько величин, например, если в методе создаются объекты или инициализируются ресурсы.