Лексические основы C#

Структура программы

C# имеют расширение .cs.

```
usingSystem;
Class example // имя класса можно изменять
{
public static void Main()
  {
intx;  // объявление переменных
inty;
x = 100;
y = 50;
Console.WriteLine("x содержит" + x); // операция конкатенации(слияние строк)
Console.WriteLine("y содержит: " + y);
    double summ;
summ = x + y;
Console.WriteLine("x + y = {0}", summ);
Console.ReadLine();
   }
}
```

Алфавит и лексемы

Синтаксис – правила написания конструкции языка программирования.

Семантика – поясняет смысл синтаксических конструкций.

Алфавит – все символы, включая буквы, из которых строятся синтаксические конструкции. Все тексты на языке пишутся с помощью его алфавита.

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.*

Из символов составляются более крупные строительные блоки: лексемы, директивы препроцессора и комментарии.

Лексема — это минимальная единица языка, имеющая самостоятельный раз и навсегда неизменяемый смысл:

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

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

Директивы препроцессора – инструкции, команды, с помощью которых можно включить или выключить из процесса компиляции фрагменты кода.

Комментарии предназначены для записи пояснений к программе и формирования документации.

Из лексем составляются выражения и операторы.

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

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

int a;// оператор описания целочисленной переменной a.

Идентификаторы

Идентификатор *– имя объекта.* В идентификаторе могут использоваться буквы, цифры и символ подчеркивания. В языке C# различается регистр: NameFirst и namefirst – разные имена. Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра. Длина идентификатора не ограничена. Пробелы внутри имен не допускаются.

Имена даются элементам программы, к которым требуется обращаться: переменным, типам, константам, методам, меткам и т. д. Идентификатор создается на этапе объявления переменной (метода, типа и т. п.), после этого его можно использовать в последующих операторах программы. При выборе идентификатора необходимо иметь в виду следующее:

- идентификатор не должен совпадать с ключевыми словами
- не рекомендуется начинать идентификаторы с двух символов подчеркивания, поскольку такие имена зарезервированы для служебного использования.

Нотация — соглашение о правилах создания имен.

Венгерская нотация (ее предложил венгр по национальности, сотрудник компании Microsoft) отличается от других наличием префикса, соответствующего типу величины, например, iMaxLength, IpfnMyFuzzyShooshpanchik.

НотацияCamel - 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 нотация - разделять слова, составляющие имя, знаками подчёркивания max_length, my_fuzzy_shooshpanchik, при этом все составные части начинаются со строчной буквы.

BC# для именования различных видов программных объектов чаще всего используются две нотации: Паскаля и Camel

Ключевые слова

Ключевые слова — это зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены.

abstract as base boo Ibreak

byte case catch char checked

class constcontinue 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 sbytesealed

shortsizeof stackalloc static string

struct switch this throw true

try typeof uint ulong unchecked

unsafe ushort using virtual void

volatile while

Таблица 1 – Ключевые слова C#

Литералы

Литералами или *константами,* называют неизменяемые величины. ВС# есть логические, целые, вещественные, символьные и строковые константы, а также константа null. Компилятор, выделив константу в качестве лексемы, относит ее к одному из типов данных по ее внешнему виду.

Управляющей последовательностью ***простой escape-последовательностью*** называют определенный символ, предваряемый обратной косой чертой. Управляющая последовательность интерпретируется как одиночный символ и используется для представления:

- кодов, не имеющих графического изображения (например, \n — переход в начало следующей строки);
- символов, имеющих специальное значение в строковых и символьных литералах, например, апострофа '.

Если внутри строки требуется использовать кавычку, ее предваряют косой чертой, по которой компилятор отличает ее от кавычки, ограничивающей строку:

"Кинотеатр \"Аврора\""

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, 1U, lu)	8 O 199226 OLu 199226L 8u OxA 0x188 0X00FF OxAU 0xlB8LU 0X00FF1
Вещественная	*С фиксированной точкой*[1]*:* [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 .OOld 35 5.7F .OOlf 35m 5F .lle + 3 .lle-3 5E-10 0.2E6 0.2E6D 5E10
Символьная	Символ, заключенный в апострофы	'A' V '*' '\0' '\n' '\xF' '\x74' '\uA81B'
Строковая	Последовательность символов, заключенная в кавычки	"Здесь был Vasia" "иЗначение г — \0xF5 \n" "Здесь был \u0056\u0061" "C: WtempWfilel.txt" @" C:\temp\filel.txt
Константа null	Ссылка, которая не указывает ни на какой объект	null

Когда компилятор распознает константу, он отводит ей место в памяти в соответствии с ее видом и значением. Если по каким-либо причинам требуется явным образом задать, сколько памяти следует отвести под константу, используются *суффиксы*.

блица 4 - Суффиксы целых и вещественных констант

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

Управляющей последовательностью, или простой escape-последовательностью, называют определенный символ, предваряемый обратной косой чертой. Управляющая последовательность интерпретируется как одиночный символ и используется для представления:

- кодов, не имеющих графического изображения (например, \n — переход в начало следующей строки);
- символов, имеющих специальное значение в строковых и символьных литералах, например, апострофа ('). Если непосредственно за обратной косой чертой следует символ, не предусмотренный таблицей, возникает ошибка компиляции.

Таблица 5 - Допустимые значения последовательностей в C#

Вид	Наименование
\a	Звуковой сигнал
\b	Возврат на шаг
\f	Перевод страницы (формата)
\n	Перевод строки
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\\	Обратная косая черта
\'	Апостроф
\"	Кавычка
\0	Нуль-символ

Комментарии

Комментарии предназначены для записи пояснений к программе и формирования документации. Компилятор комментарии игнорирует. Внутри комментария можно использовать любые символы. 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.

Операции языка C#

Логические операции: & - И, | - ИЛИ, ^ - Исключающее ИЛИ, && - Сокращенное И, | | - Сокращенное ИЛИ, ! – НЕ.

Операция конъюнкции (логическое умножение):

A	B	A&&B
1	1	1
1	0	0
0	1	0
0	0	0

Операция дизъюнкции (логическое сложение):

A	B	A||B
1	1	1
1	0	1
0	1	1
0	0	0

Логическое отрицание (инверсия):

A	!A
1	0
0	1

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, a и b — операнды. Символы, составляющие знаки операций, могут быть как специальными, например, &&, | и <, так и буквенными, такими как as или new.

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.

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

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

< > ? ++ -- && || << >> == != <= >= += -= *= /= %=

&= |= ^= <<= >>= ->

Операции отношения и проверки на равенство

Операции отношения (<, < = , >, > = , = = , ! =) сравнивают первый операнд со вторым Операнды должны быть арифметического типа. Результат операции — логической типа, равен true или false.

Операция	**Результат**
x == y	true, если x равно y, иначе false
x ! = y	true, если x не равно y, иначе false
x < y	true, если x меньше y, иначе false
x > y	true, если x больше y, иначе false
x <= y	true, если x меньше или равно y, иначе false
x >= y	true, если x больше или равно y, иначе false

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 задаётся ее ***имя и тип***.

Пример описания целой переменной с именем a и вещественной переменной x: int a; float x;

При объявлении можно присвоить переменной некоторое начальное значение, т.е. **инициализировать** ее, например:

```
a, b = 3;
int x = 3.14;
```

При инициализации можно использовать не только константу, но и выражение — такое, чтобы на момент описания оно было вычисляемым, например:

```
b = 1, a = 100;
x = b * a + 25;
```

Инициализировать переменную прямо при объявлении не обязательно, но перед тем, как ее использовать в вычислениях, это сделать необходимо, иначе компилятор сообщит об ошибке.

Блок — это код, заключенный в фигурные скобки. Основное назначение блока — группировка операторов.

Метод Write делает то же самое, что и WriteLine, но не переводит строку.

Для ввода данных используется метод ReadLine.

Описание и вывод на экран локальных переменных.

```
namespace Example2
{
    class Program
    {
        static void Main()
        {
            int i = 5;
            double y = 5.67;
            decimal d = 984m;
            string s = "простейший вывод переменных";
            Console.Write("i = ");
            Console.WriteLine( i);
            Console.Write("y = ");
            Console.WriteLine( y);
            Console.Write("d = ");
            Console.WriteLine( d);
            Console.Write("s -  ");
            Console.WriteLine(s);
            Console.ReadLine();
        }
    }
}
```

Типы данных

- внутренние представление данных (множество их возможных значений).
- допустимые действия над данными.

Память, в которой хранятся данные, делится на две области: stack и heap.

Stack – используется для хранения величин, память под которую выделяет компилятор.

В динамической области (***heap***) память резервируется и освобождается во время выполнения программы с помощью специальных команд. Основным местом для хранения данных в C# является heap.

Кэш - область оперативной памяти предназначенная для временного хранения информации.

Классификация типов:

1)по строению элемента:
- простые;
- структурированные.

2)по своему «создателю»:
- встроенные;
- стандартные;
- определяемые программистом.

3)по распределению памяти:
- статические;
- динамические.

Таблица 6 – Встроенные типы C#

Имя типа	*Размер в битах*	*.NET-имя структуры*	*Описание*
bool	8	Boolean	Логический
byte	8	Byte	8-разрядное целое без знака
char	16	Char	Символьный
decimal	128	Decimal	Числовой тип для финансовых вычислений
double	64	Double	С плавающей точкой двойной точности
float	32	Single	С плавающей точкой
int	32	Int32	Целочисленный
long	64	Int64	Длинное целое
sbyte	8	Sbyte	8-разрядное целое со знаком
short	16	Int16	Короткое целое
uint	32	UInt32	Целое без знака
ulong	64	UInt64	Длинное целое без знака
ushort	16	UInt16	Короткое целое без знака

Строки

Используется тип string. В C# строка – это объект, а не массив символов.

Свойство Length позволяет получить длину строки. Чтобы получить значение отдельного символа строки (но не присвоить новое значение), можно использовать индекс (индексация с нуля).

Таблица 7 – Наиболее часто используемые методы обработки строк

static string Copy(string str)	Возвращает копию строки str
Int CompareTo(string str)	Возвращает отрицательное значение, если вызывающая строка меньше строки str, положительное значение, если наоборот, и нуль, если строки равны
int IndexOf(string str)	Выполняет в вызывающей строке поиск подстроки, заданной параметром str. Возвращает индекс первого вхождения искомой подстроки или –1, если она не будет обнаружена
int LastIndexOf(string str)	Аналогично, только возвращает индекс последнего вхождения
string ToLower()	Возвращает строчную версию вызывающей строки
string ToUpper()	Возвращает прописную версию вызывающей строки
string SubString(int start, int len)	Возвращает новую строку, которая содержит часть вызывающей строки. start – индекс начала, len – длина подстроки

Оператор == позволяет узнать, равны ли две строки (хотя обычно применительно к объектам он определяет, относятся ли обе ссылки к одному и тому же объекту). Аналогично оператор !=. остальные операторы отношений (>, >= и т.д.) сравнивают ссылки так же, как и объекты других типов.

С помощью оператора + можно объединить несколько строк.

Перечисления

Перечисление – это множество именованных целочисленных констант. Формат записи перечисления:

enum имя {список_перечисления};

enum color {Red, Blue, Black=10, Green};

Каждый символ списка перечисления означает целое число, причем каждое следующее число (представленное идентификатором) на единицу больше предыдущего. Значение первого символа перечисления равно нулю. Но символы в перечислении можно определить с помощью инициализатора (Black =10). Green теперь будет = 11.

Константу перечисления можно использовать везде, где допустимо целочисленное значение.

Математические функции класса Math.

Математические функции, реализованные в классе Math определены в пространстве имён System.

С помощью методов этого класса можно вычислить:

- тригонометрические функции: Sin, Cos, Tan;
- обратные тригонометрические функции: ASin, ACos, ATan, ATan2;
- гиперболические функции: Tanh, Sinh, Cosh;
- экспоненту и логарифмические функции: Exp, Log, Log10;
- модуль (абсолютную величину), квадратный корень, знак: Abs, Sqrt, Sign;
- округление: Ceiling, Floor, Round;
- минимум, максимум: Min, Max.;
- степень, остаток: Pow, IEEEReminder;
- полное произведение двух целых величин: BigMul;
- деление и остаток от деления: DivRem.

Таблица 8 - Основные поля и статические методы класса Math.

Имя	Описание	Результат	Пояснения
Abs	Модуль	Перегружен	|x| записывается как Abs(x)
Acos	Арккосинус	double	Acos(double x)
Asin	Арксинус	double	Asin(double x)
Atan	Арктангенс	double	Atan(double x)
Atan2	Арктангенс	double	Atan2(double x, double y) —угол, тангенс которого есть результат деления y на x
BigMul	Произведение	long	BigMul (int x, int y)
Ceiling	Округление до большего целого	double	Ceiling(double x)
Cos	Косинус	double	Cos(double x)
Cosh	Гиперболический косинус	double	CoshCdouble x)
DivRem	Деление и остаток	Перегружен	DivRem(x, y, rem)
E	База натурального логарифма (число e)	double	2,71828182845905
Exp	Экспонента	double	ex записывается как Exp(x)
Floor	Округление до меньшего целого	double	Floor(double x)
IEEERemainder	Остаток от деления	double	IEEERemainder(double x, double y)
Log	Натуральный логарифм	double	$Log_e x$ записывается как Log(x)
Log10	Десятичный логарифм	double	$Log_{10} x$ записывается как Log10(x)
Max	Максимум из двух чисел	Перегружен	Max(x,y)

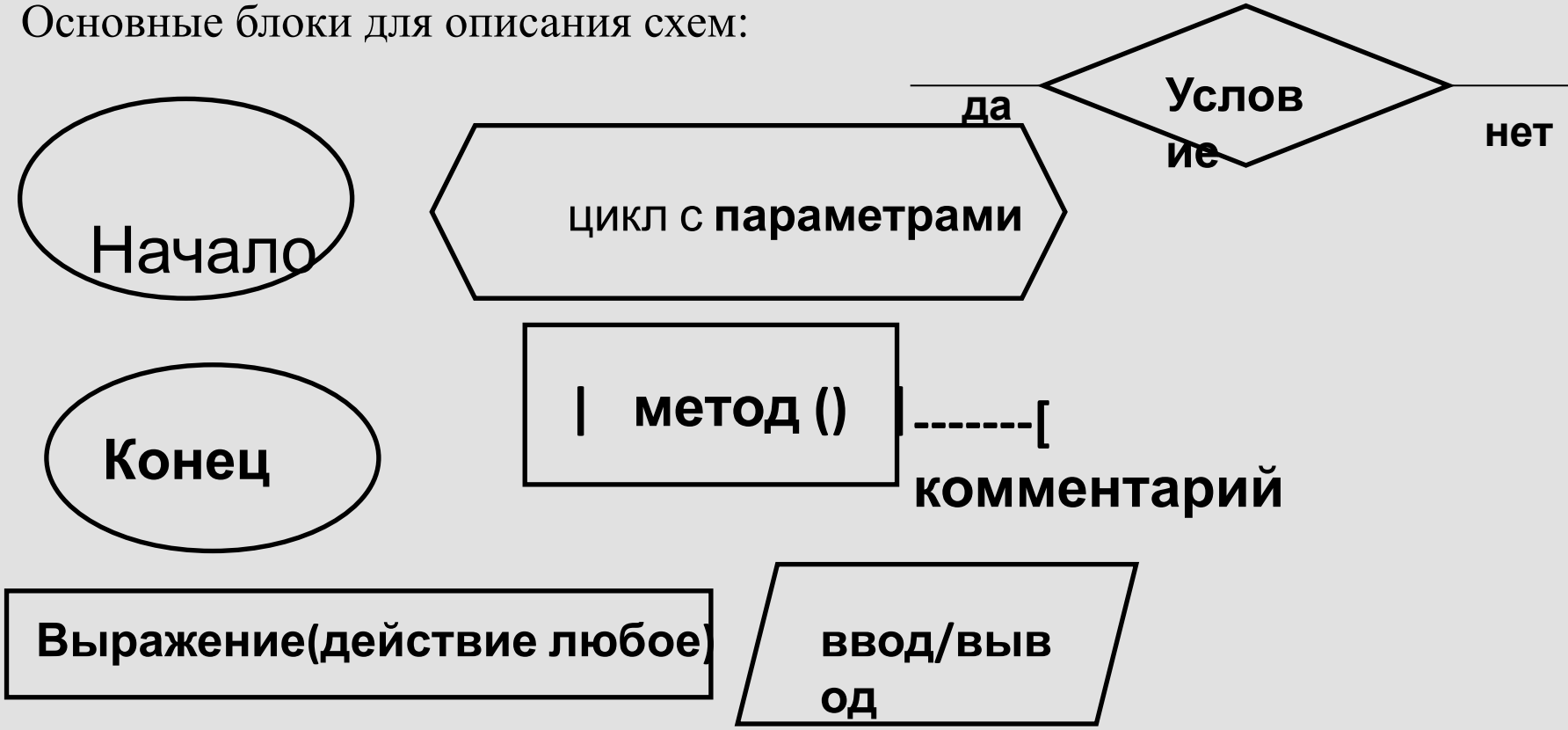
$\sqrt{x}$

Min	Минимум из двух чисел	Перегружен	Min(x,y)
PI	Значение числа	double	3,14159265358979
Pow	Возведение в степень	double	xy записывается как Row(x,y)
Round	Округление	Перегружен	Round(3.1)даст в результате 3,Round (3.8) даст в результате 4
Sign	Знак числа	int	Аргументы перегружены
Sin	Синус	double	Sin(double x)
Sinh	Гиперболический синус	double	Sinh(double x)
Sqrt	Квадратный корень	double	записывается как Sqrt(x)
Tan	Тангенс	double	Tan(double x)
Tanh	Гиперболический тангенс	double	Tanh(double x)

Алгоритм

Алгоритм - это последовательность команд, предназначенная исполнителю, в результате выполнения которой он должен решить поставленную задачу. Сам алгоритм как последовательность действий может быть описан словесно в виде двух схем или на алгоритмическом языке.

Основные блоки для описания схем:



Синтаксис объявления переменных:

тип имя переменная 1, переменная 2

int a, b; // объявление переменных

Инициализация – присвоение начальных значений данным

int sum = 0; // “=” - оператор присваивания

sum - имя(идентификатор). Можно дать любое Идентификатор- имя данных.