# Лекция 1. **Элементы языка, типы данных, переменные, константы.**

**Цель лекции:** Формирование у студентов знаний, связанных с элементами языка и типами данных C#.

В результате изучения данной лекции студенты будут способны:

- Продемонстрировать знание элементов и типов данных языка C#
- Различать переменные и константы
- Продемонстрировать понимание ситуаций для применения явных и неявных преобразований типов

Множество символов языка C# включает прописные и строчные буквы латинского алфавита, цифры и знаки. Буквы и цифры используются для формирования идентификаторов, констант, ключевых слов. *В C# большие и маленькие буквы разные, как в именах переменных, так и при написании служебных слов.*

*Лексема* – это минимальная единица языка, имеющая самостоятельный смысл. Существуют следующие виды лексем: имена (идентификаторы), ключевые слова, знаки операций, разделители, константы. Совокупность ключевых слов составляет словарь языка. Все зарезервированные ключевые слова содержат только строчные буквы (символы нижнего регистра) и написаны на английском языке. Нелатинский алфавит допускается только в комментариях, внутри символьных строк между двойными кавычками и для символьных переменных между одинарными кавычками.

*Пробел* используется не только в качестве разделителя лексем языка в тексте программы, но и как символьный знак. Компилятор не реагирует на дополнительные, «лишние», пробелы в тексте программы, поэтому их можно вводить для придания тексту наглядности.

Для решения задачи в любой программе выполняется обработка каких-либо *данных*. Среди данных можно выделить *константы* и *переменные*, которые, в свою очередь, могут быть самых различных типов. В табл.1.1 приведены 13 типов данных, которые положены в основу языка программирования C#.

Таблица 1.1 – Типы данных C#

| Тип данных | Значение | Что хранит | Занимает в памяти |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| Int | от –2 147 483 648<br>до 2 147 483 647 | Целочисленный тип | 4 байта |
| Uint | От 0 до 4 294 967 295 | Целочисленный без знака | 4 байта |
| Long | от – 9 223 372 036 854 775 808<br>до 9 223 372 036 854 775 807 | Длинный целочисленный | 8 байтов |
| Ulong | от 0<br>до 18 446 744 073 709 551 615 | Длинный целочисленный без знака | 8 байтов |
| Byte | От 0 до 255(без знака) | 8- ми разрядный целочисленный без знака (двоичные числа) | 1 байт |
| Sbyte | От –128 до 127 | 8-ми разрядный целочисленный со знаком | 1 байт |
| Short | От –32768 до 32767 | Короткий целочисленный | 2 байта |
| Ushort | От 0 до 65535 | Короткий целочисленный без знака | 2 байта |
| Double | от ±5,0e–324 до ±1.7e+308 | Числа с плавающей точкой двойной точности | 8 байтов |
| float | От ±1.5E–45 до ±3.4E+38 | Числа с плавающей точкой одинарной точности | 4 байта |
| decimal | От ±7,9E–28 до ±7.9E+28 | Десятичный (для финансовых расчетов) | 16 байтов |
| Bool | True, False | Логический, представляет два значения: "истина" или " ложь ". | различно |
| Char | Один символ | Символьный (один символ – кодировка Unicode) | 2 байта |
| String | От 0 до 2 миллиардов символов в кодировке Unicode | Строковый тип | Различно |

## Константы

*Константами* называют неизменяемые величины. В C# есть целые, вещественные, символьные, строковые, логические константы. Вещественные константы могут быть представлены в двух формах: с фиксированной точкой и с порядком. Символьные константы заключаются в одинарные кавычки.

*Пример*

10 – это целочисленная константа;
‘a’ – это символьная константа (символ, заключенный в апострофы); 15.347 – это вещественная константа с фиксированной точкой;
1.34e3 – это вещественная константа с порядком (1.34e3 = $1{,}34*10^3$ =1340).
True – это константа типа bool; False – это константа типа bool.

По умолчанию вещественные константы относятся к типу double. У целочисленных констант по умолчанию может быть один из следующихтипов: int, uint, long, ulong в зависимости от значения константы. Явно указать тип константы можно с помощью суффикса.

*Пример*

12 – константа типа int 12L – константа типа long
10.23F –это константа типа float

*Пример*

«Информатика» – это строковая константа (набор символов, заключенных в двойные кавычки).
Помимо обычных символов, строковая константа может содержать одну или несколько управляющих последовательностей символов. Среди них самые распространенные:
\n – перевод строки;
\t – горизонтальная табуляция.

*Пример*

```
Console.WriteLine("Первая строка\nВторая строка\nТретья строка");
Console.WriteLine("Один\tДва\tТри");
```

Результат:

Первая строка Вторая строка Третья строка Один Два Три;

Кроме описанных строковых литералов, существуют *буквальные строковые литералы.* Такой литерал начинается с символа @, после которого следует строка в кавычках. Буквальные строковые литералы выводятся в том же виде, в каком они введены в исходном тексте программы. Преимущество буквальных строковых констант заключается в том, что они позволяют указать в программе выводимый результат именно так, как он должен выглядеть на экране.

*Пример*

```
Console.WriteLine(@"Первая строка Вторая строка
Третья строка");
```

Результат:

Первая строка Вторая строка Третья строка

## Именованные константы

Именованные *константы* применяются для того, чтобы вместо значений констант можно было использовать в программе их имена. Такие константы при описании задаются ключевым словом const.

*Пример*

```
const int b=10, float y=0.15F;
```

## 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*. *Синтаксис оператора объявления переменной:*

```
тип имя_переменной;
```
*Пример*
```
int i, k; bool b1; float x; char c; double
z;
```
Имена переменных называют *идентификаторами*. В идентификаторе могут использоваться буквы, цифры и символ подчеркивания. Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра. Длина идентификатора не ограничена. Пробелы внутри имен не допускаются. Идентификатор не должен совпадать с ключевыми словами. В идентификаторах C# можно использовать буквы разных алфавитов, например, русского или греческого. Прописные и строчные буквы различаются, например, A и a – два разных имени. Задать значение переменной можно с помощью оператора присваивания. Кроме того, задать начальное значение переменной, т.е. инициализировать ее, можно при ее объявлении. Ниже приведен синтаксис оператора инициализации переменной:

*Синтаксис оператора:*
```
тип имя_переменной = значение;
```
где значение – это конкретное значение, задаваемое при создании переменной. Оно должно соответствовать указанному типу переменной.

*Пример*

int i=10; // задать начальное значение 10 переменной *i*

float y = 1.245F; // переменная *f* инициализируется числовым значением 1,245 double z = 5.76; // или z = 5.76D; z = -76.45d;

s1 = "ABCDEF";

b1 = true;

При присваивании числового значения (константы) переменной типа float следует использовать суффикс F(f). При присваивании переменной double или float константы следует ставить после десятичной точки 0, даже если число не имеет значащих десятичных знаков. В противном случае число может интерпретироваться компилятором как целое.

*Пример*

double a = 123.0; float w = 827.0f;

Самым распространенным в программировании целочисленным типом является тип int. Переменные типа int нередко используются для управления циклами, индексирования массивов. Типы с плавающей точкой позволяют представлять числа с дробной частью. Для математических расчетов наибольшей популярностью пользуется тип double, так как позволяет производить вычисления с большой точностью. Этот тип используется во многих математических функциях библиотеки C#. Например, метод Math.Sqrt(x) возвращает значение типа double, которое представляет собой квадратный корень из аргумента типа double, передаваемого данному методу.

Тип decimal предназначен для денежных расчетов. При присваивании константы переменной типа decimal следует использовать суффикс M(m), например:

Decimal cost = 13569.34m, size = 9845M;

*Пример*

bool b1= true, b2= false;

тип bool – это булевы переменные, для них определены только два значения: true (истина) и false (ложь).

*Пример*

Char a = 'x’; // символьная переменная

string b = "первая строка"; // строковая переменная

**Неявно типизированные переменные**

Неявно типизированная переменная объявляется с помощью ключевого слова var и должна быть непременно инициализирована. Компилятору предоставляется возможность самому определить тип переменной, исходя из значения, которым она инициализируется.

*Пример*

```
var x = 2.4578;
```

В данном примере переменная *x* инициализируется вещественной константой, которая по умолчанию имеет тип double, и поэтому переменная относится к типу double.

*Пример*

```
var y = 4.5783F; // переменная отнесена к типу float
```

**Явное и неявное преобразование данных**

Язык C# строго типизирован, то есть контролирует типы данных и определяет разрешенные для них операции. Арифметические действия производятся над переменными одного типа, поэтому *исходные составляющие арифметического выражения должны быть приведены (преобразованы) к одному типу.*

*Общее правило: неявно можно выполнять все преобразования,* которые не приведут к потере информации. Поэтому данные типов bool, double, decimal не могут быть неявно преобразованы ни в какие типы данных. Тип float может быть преобразован в double; int может быть преобразован в long, float, double, decimal; long может быть преобразован в float, double, decimal.

Явное преобразование выполняется следующим образом:

```
(новый тип данных) переменная
```

*Пример*

```
x = (float)56.3; // константа типа double преобразуется в float i = (int)8.6; // результат i = 8
```

В языке программирования C# всем переменным до их использования в выражениях должны быть присвоены значения, в том числе нули и пустые строки. Использование в выражениях неинициализируемых переменных считается ошибкой!

## Область видимости переменных

Область видимости локальной переменной или локальной константы ограничена текущим блоком. *Блок операторов* представляет собой последовательность операторов, заключенную в фигурные скобки (токены { }). В этом блоке нельзя объявить другую локальную переменную с тем же именем.

## Операции и выражения в языке C#, математические функции (методы)

*Выражение* задает порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций.

*Пример:*

a*b–Math.sin(x)

По характеру выполняемых действий *операции* можно разделить на следующие группы:

арифметические, операции сравнения, булевские (логические) операции, поразрядные логические и сдвиговые операции, строковая операция (конкатенация) и др. Арифметические операции сложения, вычитания, умножения и деления применимы к любым числовым типам.

Арифметические операции обозначают привычными знаками +, –

, *, /, %. При работе с целыми числами операция / дает частное, а операция % остаток от деления. Последняя операция допустима только для целых. *При делении целых чисел остаток будет отброшен.* Для деления двух целых необходимо менять тип хотя бы одного из них.

*Пример*

```
i = 14; k = 4; x = (float)i/k;
```

Ниже в табл.1.2 приведены арифметические операции, которые использует C#. К арифметическим данным могут применяться математические функции (методы), которые содержаться в библиотеке Math. Кроме того, у класса Math есть два полезных поля: число $\pi$ и число $e$ . Список математических методов класса System.Math приведен в таблице 1.3.

| Операция | Знак | Запись | Действие |
|---|---|---|---|
| Арифметическое сложение | + | a + b | Складывает два числа |
| Унарный плюс | + | +a | |
| Арифметическое вычитание | – | a – b | Вычитает из одного числа другое |
| Унарный минус | – | –a | |
| Умножение | * | a * b | Умножает два числа |
| Деление | / | a / b | Делит два числа и возвращает результат с плавающей точкой |
| Деление по модулю (операция вычисления остатка) | % | a%b | Получение остатка от целочисленного деления (7%3=1). Можно применять и к типам с плавающей точкой (10.0%3.0=1). |
| Присваивание | = | a=b | Задает новое значение переменной |
| Инкремент | + + | i++;<br>++i; | постфиксная форма, i = i+1;<br>префиксная форма, i = i+1. |
| Декремент | – – | i– –;<br>– – i; | постфиксная форма, i = i–1;<br>префиксная форма, i = i–1. |
| Доступ к элементу | . (точка) | | Выбор члена (класса или объекта) |
| Выделение памяти | new | | Создание объекта (создание экземпляра) |

Таблица 1.3 – Список математических методов класса System.Math

| Математические методы | Обозначение в математике | Назначение |
|---|---|---|
| 1 | 2 | 3 |
| `Math.Abs(x)` | $\|x\|$ | Модуль (абсолютная величина числа $x$) |
| `Math.Acos(x)` | $\arccos x$ | Арккосинус числа $x$ (в диапазоне от –1 до 1) |
| `Math.Asin(x)` | $\arcsin x$ | Арксинус числа $x$ (в диапазоне от –1 до 1) |
| `Math.Atan(x)` | $arctg\, x$ | Арктангенс числа $x$ (в диапазоне от –1 до 1) |
| `Math.Ceiling(x)` | | Округление до большего целого. Возвращение ближайшего целого, большего, чем значение аргумента $x$. *Пример* `Math.Ceiling(1.04)` даст в результате 2 |
| `Math.Cos(x)` | $\cos x$ | Косинус угла $x$. Угол $x$ задается в радианах. |
| `DivRem(x,y,rem)` | | Деление и остаток |
| `Math.E` | $e$ | Основание натурального логарифма (e = 2,718) |

| | | |
|---|---|---|
| `Math.Exp(x)` | $e^x$ | Экспонента.<br>$e$ – основание натурального логарифма, возведенное в степень $x$. |
| `Math.Floor(x)` | | Округление до меньшего целого.<br>Возвращает ближайшее целое число, меньшее, чем значение аргумента $x$.<br>*Пример*<br>`Math.Floor(1.04)` даст в результате 1 |
| `Math.IEERemainder(x,y)` | | Остаток от деления.<br>Возвращает остаток от деления числа $x$ на число $y$ |
| `Math.Log(x)` | $\ln x$ | Натуральный логарифм $x$, где $x > 0$ |
| `Math.Log10(x)` | $\lg x$ | Десятичный логарифм $x$ |
| `Math.Log(x, a)` | $\log_a x$ | Логарифм $x$ по основанию $a$ |
| `Math.Max(x,y)` | | Максимум из двух чисел. |
| `Math.Min(x,y)` | | Минимум из двух чисел. |
| `Math.PI` – константа | | Значение числа $\pi$ =3,1415926 |
| `Math.Pow(x,y)` | $x^y$ | Возведение $x$ в степень $y$ |
| `Math.Round(x)` | | Округление.<br>Округляет заданное число до ближайшего целого (арифметическое округление)<br>*Пример*<br>`Math.Round(3.1)` даст в результате 3.<br>`Math.Round(3.8)` даст в результате 4. |
| `Math.Round(x, n)` | | Округление.<br>Округляет число с плавающей запятой до заданного количества дробных разрядов $n$ |

| | | |
|---|---|---|
| Math.Sign(x) | | Знак числа.<br>Возвращает значение, определяющее знак целого числа. $Sign(x)=\begin{cases}-1 & x<0\\ 0 & \text{если } x=0\\ +1 & x>0\end{cases}$ |
| Math.Sin(x) | $\sin x$ | Синус угла $x$. Угол $x$ задается в радианах. |
| Math.Sqrt(x) | $\sqrt{x}$ | Корень квадратный из числа $x$, где $x \geq 0$ |
| Math.Tan(x) | $tg\,x$ | Тангенс угла $x$. Угол $x$ задается в радианах. |
| Math.Truncate(x) | | Вычисляет целую часть заданного десятичного числа. |
| Rnd(x) | | Случайное число из диапазона 0 – 1 |

***Примечание***. Результат вычисления большинства математических функций – тип double, аргументы функций также должны быть типа double, например, Math.Sin(double x). Названия функций пишутся с большой буквы. Результат функции Math.Sign(x) имеет тип int.

Любое выражение, завершающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается в вычислении выражения. Блок, или *составной оператор*, – это последовательность описаний и операторов, заключенная в фигурные скобки. Блок воспринимается компилятором как один оператор и может использоваться

Значение переменной y будет равно 10, так как в этом случае значение переменной x сначала присваивается переменной y, а затем увеличивается на 1. В обоих случаях значение переменной x оказывается равным 11. Отличие состоит лишь в том, когда именно это значение станет равным 11: до или после его присваивания переменной y.

## Ввод-вывод в консольном режиме

Программируя на C# в .NET Framework, можно разрабатывать: консольное приложение, Windows-приложение.

Любая программа при вводе исходных данных и выводе результатов взаимодействует с внешними устройствами. Совокупность стандартных устройств ввода и вывода, т.е. клавиатуры и экрана, называется *консолью*.

В языке C# нет операторов ввода и вывода. Вместо них используются функциональные методы класса Console, входящего в библиотеку классов программной платформы Microsoft NET Framework. Эту библиотеку называют Framework Class Library, или сокращенно FCL. Библиотека FCL является одним из важнейших компонентов всей платформы Microsoft NET Framework. Класс Console определён в пространстве имён System. Почти во всех консольных программах применяются методы этого класса:

Write(выводимая строка);
WriteLine(выводимая строка);
ReadLine() – возвращает введенную строку.

Важное обстоятельство: метод ReadLine() всегда возвращает данные типа string, в случае необходимости преобразования должны быть запрограммированы.

Аргументом методов Write, WriteLine тоже должна быть символьная строка. Разница между Write и WriteLine заключается в том, что после вывода строки WriteLine осуществляется автоматически переход на следующую строку на экране, Write этого не делает. Для преобразования типов данных можно использовать методы класса Convert.

Например, ToInt32 выполняет перевод в int, ToDouble выполняет перевод в double, ToString выполняет перевод в string. Какие методы имеются в классе Convert можно узнать очень легко: достаточно набрать это слово, поставить точку и на экране появится весь перечень его методов.

**Заключение**

В рамках данной лекции были рассмотрены фундаментальные строительные блоки языка программирования C#: лексическая структура, система базовых типов данных, правила объявления и инициализации переменных и констант, а также механизмы приведения типов.

Освоение базового синтаксиса и понимание принципов строгой типизации C# является критически важным этапом для дальнейшего изучения программирования. Грамотный выбор типов данных (например, использование decimal для финансовых расчетов или int для счетчиков) напрямую влияет на точность вычислений, производительность программы и рациональное использование оперативной памяти. Умение различать ситуации, требующие явного или неявного преобразования типов, позволяет избежать ошибок компиляции и непредвиденной потери данных на этапе выполнения программы.

**Вопросы для самопроверки**

**Лексика и синтаксис:** Что такое лексема в языке C# и какие виды лексем существуют? В каких случаях в коде допускается использование символов кириллицы?

**Идентификаторы:** Перечислите основные правила и ограничения при составлении имен переменных. Являются ли имена summa и Summa идентичными для компилятора C#?

**Константы и литералы:** В чем заключается разница между обычными строковыми константами и буквальными строковыми литералами (с символом @)? Для чего используются управляющие последовательности \n и \t?

**Переменные и константы:** Чем отличается именованная константа (const) от обычной переменной? Обязательно ли инициализировать константу при её объявлении?

**Типы данных и суффиксы:** Какой тип данных по умолчанию присваивается вещественным и целочисленным литералам? Зачем при инициализации переменных типов float и decimal необходимо указывать специальные суффиксы (F и M)?

**Неявная типизация:** Каковы правила использования ключевого слова var? Можно ли объявить переменную через var без присвоения ей начального значения?

**Преобразование типов:** Сформулируйте главное правило неявного преобразования типов в C#. Приведите пример ситуации, когда программисту необходимо применить операцию явного приведения типов.

**Список используемых источников**


1. **Троелсен Э., Джепикс Ф.** Язык программирования C# и платформа .NET. — СПб.: Диалектика, 2022. — 1312 с. *(Фундаментальное руководство по синтаксису и архитектуре языка).*

2. **Албахари Д., Албахари Б.** C#. Справочник. Полное описание языка. — СПб.: БХВ-Петербург, 2023. — 1088 с. *(Подробный разбор типов данных, литералов и правил преобразования).*

3. **Шилдт Г.** C# 4.0: полное руководство. — М.: Вильямс, 2019. — 1056 с. *(Классическое учебное пособие по основам языка C#).*

4. **Павловская Т. А.** C#. Программирование на языке высокого уровня: Учебник для вузов. — СПб.: Питер, 2020. — 432 с. *(Учебник с упором на базовые элементы языка и методику обучения).*

5. **Официальная документация Microsoft Learn:** Руководство по программированию на C#. Встроенные типы и преобразования [Электронный ресурс]. — URL: microsoft.com (дата обращения: 13.11.2026).