Синтаксис C#

(константы, переменные, массивы, консоль, пространство имен)

Андрей Голяков

Константа const

Константы позволяют задавать неизменяемые значения на этапе компиляции. При этом доступ к ним обеспечивается с помощью имён, как и в случае переменных.

В отличие от переменных, значения констант буквально вписываются в компилируемый код!

Константы помогают писать более универсальный и удобочитаемый код.

```
const string name = "Andrey";          // string constant
const int age = 36;                    // integer constant
const double height = 1.73;            // floating constant

const string fullName = name + " Golyakov"; // dependency allowed

age = age + 4;                         // gives compile error!
```

Тип данных object

Это специальный тип данных, который может хранить в себе значения любых типов данных, однако работа при использовании object для некоторых типов данных может быть очень неэффективной с точки зрения вычислительных операций.

```
object name   = "Andrey";              // storing a string in an object
object age    = 36;                    // storing an int in an object
object height = 1.73;                  // storing a double in an object

int length1   = name.Length;           // gives compile error!

int length2 = ((string)name).Length; // cast to access members
```

Тип данных dynamic

- Это ещё один специальный тип данных, который может хранить в себе значения любых типов данных, однако в отличие от object, может приводиться к нужному типу “на лету”, т.е. во время выполнения программы.
- Минусами здесь является то, что 1) в MS Visual Studio intelliSense не подсказывает имена внутренних членов для таких переменных, 2) возможны ошибки времени выполнения (runtime).

```
// storing a string in a dynamic object
dynamic anotherName = "Alexander";

// this compiles but might throw an exception at run-time!
int length = anotherName.Length;
Console.WriteLine(length);
```

Ключевое слово var

При объявлении переменных можно не указывать тип явно, он будет рассчитан на основе инициализирующего значения.

```
int population = 66_000_000;
double weight = 25;
float height = 1.88F;
decimal price = 4.99M;
string fruit = "Apples";
char letter = 'Z';
bool happy = true;
```

```
var population = 66_000_000;
var weight = 25;
var height = 1.88F;
var price = 4.99M;
var fruit = "Apples";
var letter = 'Z';
var happy = true;
```

Ключевое слово var (лучшие практики)

Хорошей практикой использования var является такое использование, когда из этой же строчки кода однозначно понятен тип данных переменной.

```
// Good
var population = int.Parse("15");
var fruit = "Apples";
var letter = 'Z';
var happy = true;
var o = new object();
```

```
// Bad
var weight = 250_000_000_000;
var price = 4.99;
var file = File.CreateText(...);
```

Ссылочные и значимые типы данных

- Числа и булевы переменные являются значимыми типами (value types). Значимые переменные обязательно должны иметь значение.

- string, object, dynamic являются ссылочными типами (reference types). Ссылочные типы могут не иметь значения, это записывается как null.

```
Console.WriteLine($"{default(int)}");       // 0
Console.WriteLine($"{default(bool)}");      // False
Console.WriteLine($"{default(DateTime)}");  // 1/01/0001 00:00:00

Console.WriteLine($"{default(string)}");    // ???
```

Nullable для значимых типов

Использование знака вопроса ? после имени типа превращает значимый тип данных в nullable тип.

```
int? a = null;
Console.WriteLine(a); // <null>
a = 4;
Console.WriteLine(a); // 4
```

Array: массив однотипных значений

Массивы используются, когда необходимо работать с несколькими значениями одного типа. Например, надо обработать несколько имён:

```
// declaring the size of the array
string[] names = new string[4];

// storing items at index positions
names[0] = "Andrey";
names[1] = "Maria";
names[2] = "Alexander";
names[3] = "Eugenia";

// looping through the array
for (int i = 0; i < names.Length; i++)
{
    Console.WriteLine(names[i]); // read the item at this index
}
```

Array: массив однотипных значений

Массивы могут инициализироваться целиком следующим образом:

```
// declaring and defining all the array at once
string[] names =
{
    "Andrey",
    "Maria",
    "Alexander",
    "Eugenia",
};

for (int i = 0; i < names.Length; i++)
{
    Console.WriteLine(names[i]); // read the item at this index
}
```

Console: Консольный ввод-вывод

Для обращения с консолью используется статический класс Console.

```
Console.WriteLine("Key Analyzer App"); // Writes text and starts a new line
Console.Write("Enter string: "); // Writes text without starting a new line
Console.WriteLine(); // Just starts a new line

string s = Console.ReadLine(); // Reading a line
Console.WriteLine(s); // Repeat the entered line on the screen

ConsoleKeyInfo cki1 = Console.ReadKey(); // Reads a single key
Console.WriteLine(cki1.Key.ToString());  // Writes the name of pressed key

ConsoleKeyInfo cki2 = Console.ReadKey(false); // Reading single key (hidden)
Console.WriteLine($"Key {cki1.Key}");  // Writes the name with template
```

Namespace: Пространство имен

Объявление пространств имен в самом начале файла – хорошая практика для сокращения кода и улучшения его читабельности. Это делается так:

```
using System;
```

В нашем примере класс Console находится в области видимости System и мы имеем к нему доступ по имени потому что мы импортировали его используя ключевое слово using. Если закомментировать импорт пространства имен System, придётся его писать перед каждым обращением к классу Console, иначе компилятор не найдет нужного класса:

```
// using System;
...
System.Console.WriteLine("Key Analyzer App");
System.Console.Write("Press string: ");
System.Console.WriteLine();
```

Домашнее задание #1 (попроще)

Написать консольное приложение, запрашивающее у пользователя имена трех человек. Затем также запрашивающее возраст этих людей. Затем программа должна вывести на экран введенную информацию в следующем формате:

```
Name: [name # 1], age in 4 years: [age of the #1 person in 4 years]
Name: [name # 2], age in 4 years: [age of the #2 person in 4 years]
Name: [name # 3], age in 4 years: [age of the #3 person in 4 years]
```

Пример вывода:

```
Name: Andrey, age in 4 years: 40
Name: Alex, age in 4 years: 23
Name: Artem, age in 4 years: 5
```

Программа не должна закрываться пока не нажата любая клавиша.
Необходимо выполнить задание с использованием массивов!

Домашнее задание #2 (посложнее)

Вывести на экран таблицу умножения Пифагора 10×10 элементов от 1 до 10. Исходные значения множителей должны храниться в массивах.

(* суперсложная часть) Спроектировать приложение так, чтобы изменение количества или значений множителей потребовало минимум изменений в коде (1–2 изменения).

Пример вывода:

*	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Спасибо за внимание.