Introduction to programming in c#

About me

Piotr Kątny

Software Engineer and Technical Lead at SII Poland

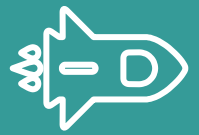
p.katny@gmail.com

Plan for today

- Introduction to C# and .NET
- Working with projects in Visual Studio
- Variables and expressions in c#
- **Console applications**
- **Operations**
- **Conditions**
- **Collections introduction**
- **Loops**
- ***(Classes and objects)***
- ***(String operations)***
- Exception handling

More variables

Review



1. What are variables?
2. How do we use them?
3. How do we name them?

Variable types

Primitive (built-in types) - embedded in C# language at the lowest level

- Integer – `sbyte, byte, short, ushort, int, uint, long, ulong;`
- Real floating-point types – `float, double;`
- Real type with decimal precision – `decimal`;
- Boolean type – `bool`;
- Character type – `char`;
- String – `string`;
- Object type – `object`

Variable types

Constants

```
var circleArea = 3.14 * radius * radius;

var electricityBillAmount = totalHours * 2.19;
```

Variable types

Constants

```
const double pi = 3.14159;
const decimal costPerHour = 1.27m;

var circleArea = pi * radius * radius;
var electricityBillAmount = totalHours * costPerHour;
```

Variable types

Enumerations

efficient way to define a **limited set** of **meaningful** named **constants** that may be assigned to a variable.

```
enum Day { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday };
```

```
enum Fruits { Orange, Banana, Apple, Strawberry };
```

By default consecutive int values, 0-indexed

Variable types

Enumerations with explicitly set values

efficient way to define a **limited set** of **meaningful** named **constants** that may be assigned to a variable.

```
enum JapaneseBreak
{
 Shiku = 5,
 KawaShiku = 10,
 KawaShikuFaya = 15,
 KaNaPa = 60
}
```

Variable types

Reference type

- does not contain the actual data stored in a variable, but a ***reference*** to the **variable**.
- in other words - they refer to a memory location. Several reference type variables can point to same variable value
- if the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value (example: ***paczkomat***)
- **built-in** reference types are: **object**, **dynamic,** and **string**.

Variable scope

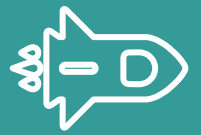
Variables can be defined at:

- **class** level (all “functions” in the class have access to it)
- **method** level (only available inside of the method)
- **block** level (only in the block)

In general we should define variables as close to their use as possible

When can we have a variable with the same name?

Questions



1. Difference between value and reference type?
2. How to make a variable accessible in all <***functions***> in our <***class***> ?

Console Apps

Console App

- Write information for user

```
Console.Write();
Console.WriteLine();
```

- Get input from user

```
Console.ReadKey();
Console.ReadLine();
```

Console App

- Other console features

```
Console.ForeGroundColor
Console.Title
Console.Beep()
Console. … (ctrl + space)
```

Console App – Exercise

- Create a console app that asks for your first name and last name and then displays „Hello *`first_name last_name`*”
- Examine result of `Console.ReadKey()`

Operators & Conditions

Operators

- Multiplicative: * , / , %
- Additive: + , -
- Relational: < , > , <= , >=
- Equality: = = , ! =
- Logical: && , | |
- Assignment: =

Full reference: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/

Conditions

- Statements that run only if a certain condition is met
- A test expression is used, and the result will be a Boolean value (true or false).
- Used to make a decision on what to do next

IF - ELSE

Conditions - examples

```
if (age < 18)
{
   Console.WriteLine("You cannot vote yet!");
}
```

Conditions - examples

```
if (age <= 18)
{
   Console.WriteLine("You cannot vote yet!");
}
else
{
   Console.WriteLine("You can vote and be a politician.");
}
```

Conditions - examples

```
if (age <= 18)
{
   Console.WriteLine("You cannot vote yet!");
}
else if (age >= 35)
{
   Console.WriteLine("You can become a president of Poland!");
}
else
{
   Console.WriteLine("You can vote and be a politician");
}
```

Conditions - examples

```
if (age <= 18)
{
   Console.WriteLine("You cannot vote yet!");
}
else if (age >= 35)
{
   Console.WriteLine("You can become a president of Poland!");
}
else if (age >= 30)
{
   Console.WriteLine("You can become a senator!");
}
else if (age >= 25)
{
   Console.WriteLine("You can be a president of your city!");
}
else if (age >= 21)
{
   Console.WriteLine("You can be a Member of Parliment!");
}
else
{
   Console.WriteLine("You can only vote so far");
}
```

Conditions - examples

Other way to write if-else that you may encounter:

```
var result = (a < b) ? a : b;
```

Conditions - exercises

1. find maximum between three numbers.
2. check whether a number is divisible by 5 and 11 or not.
3. check whether a character is alphabet or not.
4. input any alphabet and check whether it is vowel or consonant.
5. input month number and print number of days in that month.
6. find all roots of a quadratic equation (ax^2 + bx +c = 0)
7. input electricity units used and calculate total electricity bill according to the given conditions:
   - For first 50 units -> PLN. 0.50/unit
   - For next 100 units ->PLN. 0.75/unit
   - For next 200 units -> PLN. 1.20/unit
   - For unit above 250 -> PLN 1.50/unit
   - An additional surcharge of 10% is added to the bill

Conditions

Switch

The *`switch`* statement is often used as an alternative to an `if-else` construct if a single expression is tested against three or more conditions.

Conditions - switch

```
switch (number)
{
   case 1: Console.WriteLine("one");
           break;
   case 2: Console.WriteLine("two");
           break;
   case 3: Console.WriteLine("three");
           break;
   case 4: Console.WriteLine("four");
           break;
   case 5: Console.WriteLine("five");
           break;
   default: Console.WriteLine(" can only count to five :( ");
           break;
}
```

Collections intro

Collections

`array`

- fixed size,
- can be multi-dimensional,
- accessed by index

```
int[] intArray = { 1, 2, 7, 8 };
string[] stringArray = { "puchatek", "klapouchy", "prosiaczek" };
```

Collections

List<T> (T – can be `int, string`, etc.)

- can be modified (add/remove items),
- strongly typed,
- less efficient

Add, AddRange, Count, Insert, Sort, Find

Collections

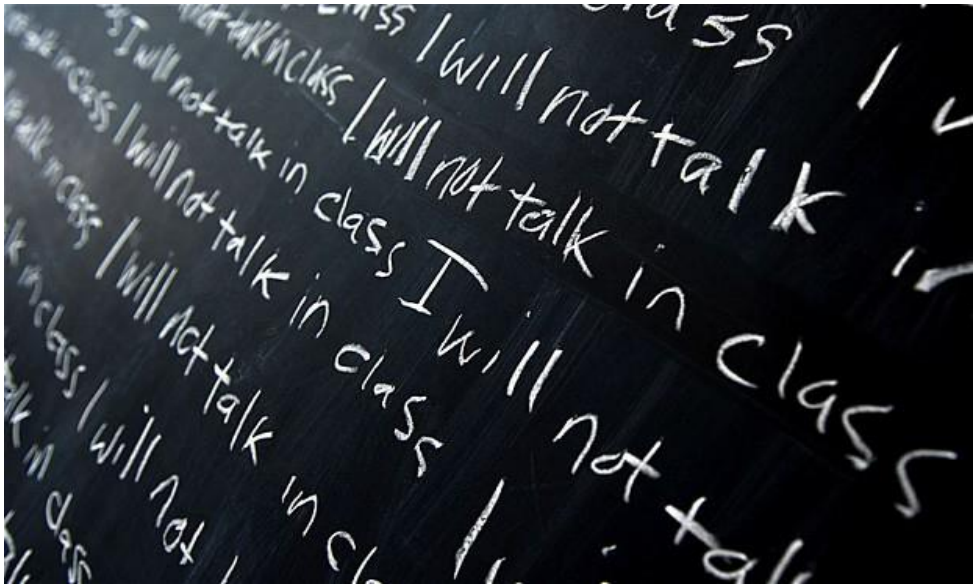
Other collections:

- Dictionary
- Queue
- Stack
- SortedList

Loops

Loops

If you have statements that need to repeat, you will probably want to use a loop structure.



For loops

- For loops will repeat a block of code a set number of times.
- **For loops** use a variable to **count** how many times the code has been repeated, called a **counter**.
- You control how many times the loop **repeats** by setting where the **counter** starts and ends.
- You also set **how much** the counter **goes up** by each time the code repeats. In most scenarios the counter is increased by 1 each time the loop repeats.

For loops

```
for (initializer; condition; iterator)
{
    body;
}
```

For loops

```
for (int i = 0; i < 100; i++)
{
    Console.WriteLine("I will never again use my cellphone in a classroom");
}

for (int i = 0; i < 15; i++)
{
    Console.WriteLine($"I said it {i} times");
}
```

While loops

While loop repeats a statement or a group of statements while a given condition is true. It tests the condition before executing the loop body.

```
while (peopleAtTheParty > 0)
{
   KeepPlayingMusic ();
}

while (peopleAtTheParty > 0 && time < midnight)
{
   KeepPlayingMusic();
}
```

Foreach loops

- The foreach statement iterates through a collection <***that implements the IEnumerable interface***>. In contract to **for** statement, the **foreach** statement does't use the indexes.

```
var cartoonCharacters = new string[] { "Mickey Mouse", "Donald Duck", "Goofy" };

foreach (var character in cartoonCharacters)
{
    Console.WriteLine(character);
}
```

Loop examples

```
string[] colors = { "red", "orange", "yellow", "green" };

Console.WriteLine(":::FOR:::");
for (int i = 0; i < colors.Length; i++)
{
    var value = colors[i];
    Console.WriteLine(value);
}

Console.WriteLine(":::FOREACH:::");
foreach (var value in colors)
{
    Console.WriteLine(value);
}
```

Loop exercises

1. print all natural numbers in reverse (from n to 1). - using while loop and using for loop
2. find sum of all even numbers between 1 to n
3. find power of a number using for loop
4. multiply 2 numbers without * operator
5. check whether a number is Prime number or not
6. print Fibonacci series up to n terms
7. enter a number and print it in words

(next) Objects and classes