Kurdistan Technical Institute

Fundamental of Programming

by

Dr. Zana Azeez Kakarash

2023 - 2024

Kurdistan Technical Institute

www.kti.edu.iq

IT Department

Variables and Data Types

Two

Variable in C#

Variables

- Variables are containers for storing data values.

- In Java, there are different **types** of variables, for example:

  - string – stores text, such as "Hello".
  - int – stores integers, without decimals, such as 123 or -123
  - float – stores floating point numbers, with decimals, such as 19.99 or -19.99
  - char – stores single characters, such as 'a' or 'B'
  - bool – stores values with two states: true or false

Creating Variables

- To create a variable, you must specify the type and assign it a value:

type **variableName** = value;

- Where type is int or string, and **variableName** is the name of the variable (like **x**).

- The = is used to assign values to the variable.

Example

```
String name = "John";
Console.WriteLine(name);
```

Example

Example One

```
String firstName = "John ";
String lastName = "Doe";
String fullName = firstName + lastName;
System.out.println(fullName);
```

Example Two

```
int x = 5;
int y = 6;
Console.WriteLine(x + y); // Print the value of x + y
```

More example

- If you assign a new value to an existing variable, it will overwrite the previous value:

```
int myNum = 15;
myNum = 20; // myNum is now 20
Console.WriteLine(myNum);
```

- **Other Types**

```
int myNum = 5;
double myDoubleNum = 5.99D;
char myLetter = 'D';
bool myBool = true;
string myText = "Hello";
```

Rules for naming variables

- The general rules for naming variables are:
    - Names can contain letters, digits, underscores, and dollar signs
    - Names must begin with a letter
    - Names should start with a lowercase letter and it cannot contain whitespace
    - Names can also begin with $ and _ (but we will not use it in this tutorial)
    - Names are case sensitive (″`myVar`″ and ″`myvar`″ are different variables)
    - Reserved words (like Java keywords, such as `int` or `bool`) cannot be used as names

Final Variables

- If you don't want others (or yourself) to overwrite existing values, use the `final` keyword.
- `final`, which means unchangeable and read-only

```
final int myNum = 15;
myNum = 20;  // will generate an error: cannot assign a value to a final variable
```

Create a variable named `carName` and assign the value `Volvo` to it.

```
          =          ;
```

Creating Many Variables

- To declare more than one variable of the same type, you can use a comma-separated list:

```
int x = 5, y = 6, z = 50;
Console.WriteLine(x + y + z);
```

- One Value to Multiple Variables

```
int x, y, z;
x = y = z = 50;
Console.WriteLine(x + y + z);
```

Fill in the missing parts to create three variables of the same type, using a comma-separated list:

```
    x = 5   y = 6   z = 50;
```

Display Variables

- The WriteLine() method is often used to display variable values to the console window.

- To combine both text and a variable, use the + character:

```
string name = "John";
Console.WriteLine("Hello " + name);
```

- For numeric values, the + character works as a mathematical operator

```
int x = 5;
int y = 6;
Console.WriteLine(x + y); // Print the value of x + y
```

Note

- You cannot assign an integer value to string type or vice-versa.

```
int num = "Steve";
```

- A variable must be assigned a value before using it, otherwise, will give a compile-time error.

```
int i;
int j = i; //compile-time error: Use of unassigned local variable 'i'
```

- The value of a variable can be changed anytime after initializing it.

```
int num = 100;
num = 200;
Console.WriteLine(num); //output: 200
```

var keyword

- C# introduced var keyword to declare method level variables without specifying a data type explicitly.

```
var j = 100; // implicitly typed local variable
```

- The compiler will infer the type of a variable from the expression on the right side of the = operator.

```
int i = 10;
var j = i + 1; // compiles as int
```

Example

```
static void Main(string[] args)
{
    var i = 10;
    Console.WriteLine("Type of i is {0}", i.GetType());

    var str = "Hello World!!";
    Console.WriteLine("Type of str is {0}", str.GetType());

    var dbl = 100.50d;
    Console.WriteLine("Type of dbl is {0}", dbl.GetType());

    var isValid = true;
    Console.WriteLine("Type of isValid is {0}", isValid.GetType());
```

```
Type of i is System.Int32
Type of str is System.String
Type of dbl is System.Double
Type of isValid is System.Boolean
```

Identifiers in C#

Identifiers

- An identifier is a name used to identify a class, variable, function or **namespace**.

- An identifier that **consist** of **letters**, **digits**, **underscores** (_), and **dollar signs** ($) and must **start** with a letter, an underscore (_), or a dollar sign ($). It cannot start with a digit.

- An identifier **cannot** be a **reserved** word like :

  int, double, if, try, private, long, return, case,class,float, static, do,while …

- An identifier **cannot** be **true**, **false**, or **null**.

Identifiers

- All variables must be identified with unique names.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalValue).

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

Example

- Some **legal** identifiers :

```
int    _a ;
int    $c ;
int    ____2_w ;
int    _$ ;
int    this_is_a_very_detailed_name_for_an_identifier ;
```

- Some **illegal** identifiers :

```
int   :b ;          // starts with :
int   -d ;          // starts with -
int   e# ;          // contains illegal character #
int   .f ;          // starts with .
int   7g ;          // starts with number
```

Data type in C#

Data Types

- A variable in C# must be a specified data type:

```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99f;    // Floating point number
char myLetter = 'D';         // Character
boolean myBool = true;       // Boolean
String myText = "Hello";     // String
```

- Data types are divided into two groups:
    - Primitive data types – includes byte, short, int, long, float, double, bool and char
    - Non-primitive data types – such as string, Arrays and Classes

Primitive Data Types

- A data type specifies the size and type of variable values.
- It is important to avoid errors, and memory

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
bool	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter, surrounded by single quotes
string	2 bytes per character	Stores a sequence of characters, surrounded by double quotes

Examples

```
int myNum = 100000;
Console.WriteLine(myNum);
```

```
long myNum = 15000000000L;
Console.WriteLine(myNum);
```

```
float myNum = 5.75F;
Console.WriteLine(myNum);
```

```
double myNum = 19.99D;
Console.WriteLine(myNum);
```

```
bool isCSharpFun = true;
bool isFishTasty = false;
Console.WriteLine(isCSharpFun);   // Outputs True
Console.WriteLine(isFishTasty);   // Outputs False
```

```
char myGrade = 'B';
Console.WriteLine(myGrade);
```

```
string greeting = "Hello World";
Console.WriteLine(greeting);
```

Exercise

1- Add the correct data type

```
    myNum = 9;
          myDoubleNum = 8.99;
    myLetter = 'A';
    myBoolean = false;
          myText = "Hello World";
```

2- Create two Boolean variables

```
      =     ;
      =     ;
```

3- Create a greeting variable, and display the value of it:

```
      =  "Hello";
Console.WriteLine(      );
```

Non-Primitive Data Types

- Non-primitive data types are called reference types because they refer to objects.

- The main difference between **primitive** and **non-primitive** data types are:

  - Primitive are defined in Java. Non-primitive are created by the programmer.
  - Non-primitive can be used to call methods to perform certain operations, while primitive cannot.
  - A primitive has always a value, while non-primitive can be null.
  - A primitive starts with a lowercase letter, while non-primitive starts with an uppercase letter.
  - The size of a primitive depends on the data type, while non-primitive have all the same size.
  - Examples of non-primitive types are strings, Arrays, Classes, Interface, etc.

Type Casting in C#

Type Casting

- Type casting is when you assign a value of one primitive data type to another type.
- In C#, there are two types of casting :
    - **Widening Casting** (automatically) – converting a **smaller** type to a **larger** type size

      `byte -> short -> char -> int -> long -> float -> double`

    - **Narrowing Casting** (manually) – converting a **larger** type to a **smaller** size type

      `double -> float -> long -> int -> char -> short -> byte`

Widening Casting

- `Widening` casting is done automatically when passing a smaller size type to a larger size type:

```
public class Main {
  public static void main(String[] args) {
    int myInt = 9;
    double myDouble = myInt; // Automatic casting: int to double

    Console.WriteLine(myInt);      // Outputs 9
    Console.WriteLine(myDouble);   // Outputs 9.0
  }
}
```

Narrowing Casting

- Narrowing casting must be done manually by placing the type in parentheses in front of the value:

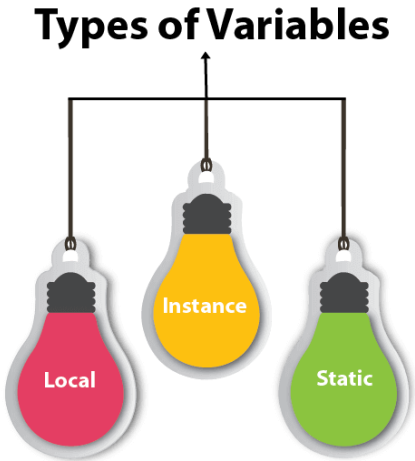
```
public class Main {
  public static void main(String[] args) {
    double myDouble = 9.78d;
    int myInt = (int) myDouble; // Manual casting: double to int

    Console.WriteLine(myDouble);   // Outputs 9.78
    Console.WriteLine(myInt);      // Outputs 9
  }
}
```

Types of Variable

Types of Variable

- There are three types of variables in C# :
  - Local variable
  - Instance variable
  - Static variable



- Local Variable : declared inside the method is called local variable.
- Instance Variable : declared inside the class but outside the method is called instance variable .
- Static variable : declared as static is called static variable. It cannot be local.

Types of Variable

```
class Test {
        int   x = 50 ;                          // instance variable
        static   int   y = 100 ;                // static variable
        public static void main (String args[ ]) {
                int   z = 90 ;                  // local variable
        }
        }
```

Thanks!

You can find me at

zana.azeez@kti.edu.iq