Fundamental Programming Concepts

- IDEs and coding environments
- Basic syntax
- Variable declaration (Local variable - Global variable)
- Data type and structures
- Flow control structures (*Conditionals* and *loops*)
- Object-oriented programming

- **Variables,** are containers for storing data values. Variables can hold values of any data type supported by the programming language. This value may change during program execution.
- **Basic syntax, e**very programming language has its syntax, and you must learn the fundamental syntax of the language you are learning. Syntax refers to the set of rules that define the structure of a language.
- **Data types and structures, d**ata types refer to the classification of data. The most common data types include:

  - String
  - Boolean (true or false)
  - Numbers, which includes integers.
  - Characters (includes single alphabets or numbers)
  - Arrays (a collection of data, usually of the same data type)

- **Flow control structures,** are the fundamental components of computer programs. They are commands that allow a program to "decide" to take one direction or another.

- **Iteration (Loops),** a **loop** is a programming structure that allows a statement or block of code to be run repeatedly until a specified condition is no longer true (will return Boolean, true or false). It is one of the most powerful and fundamental programming concepts.

Analyzing Your First C# Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace FirstProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            // My first program
            Console.WriteLine("Hello, world");
        }
    }
}
```

NOTES:

- A **program** contains one or more lines of instructions or statements.
- **Keywords** are words that have a predefined meaning in a programming language. Keywords are reserved words that developers or programmers cannot use as the name of variables, functions, etc.
- A **comment** provides information to the programmer (**One-Line Comments**: *// comment text* , **Multiline Comments**: */* this is a comment */*

- `static void Main(string[] args), ` is the starting point of every app. The **parentheses** after the identifier **Main** indicate that it's an app building block called a **method**. Methods are able to perform tasks and return information when they complete their tasks. Keyword **void** indicates that this method will not return any information after it completes its task.
- The left brace begins the body of the method declaration. A corresponding right brace must end the method's body.
- C# code block must start and end with braces.
- `Console.WriteLine("Hello, world"),` instructs the computer to **perform an action** - to display a **string**. The *Console.WriteLine* method displays a line of text in the console window.
- Class **Console** provides standard **input/output** capabilities that enable apps to read and display text in the console window.
- A **string** is a sequence of characters enclosed in a pair of single or double quotation marks.
- Most statements end with a **semicolon**.

Matching Left ({) and Right (}) Braces

Modifying the First C# Program

Displaying one line of text with multiple statements.

```
static void Main(string[] args)
{
    // My first program
    Console.Write("Hello, ");
    Console.WriteLine("world");

}
```

The C# Keywords

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	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	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
while				

Another C# Program: Adding Integers

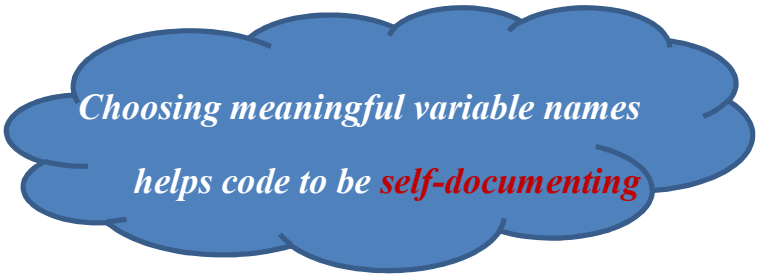
```
static void Main(string[] args)
{
    int a = 2;
    int b = 3;
    int c = a + b;

    Console.WriteLine(c);
}
```

- `int number1,` is a **variable declaration statement** that specifies the name (*number1*) and type of a variable (*int*) used in this program.
- A **avriable** is stored in the computer's memory and can be used or changed throughout the program. Each variable has a name and holds a value. A variable has name and type. To assign a value to a variable, use the equal sign (=). The = symbol is called an **assignment operator**.
- A variable's name enables the program to access the value of the variable in memory—the name can be any valid identifier.
- A variable's type specifies what kind of information is stored at that location in memory and how much space should be set aside to store that value.
- An integer value (int) is a number without a fractional part.
- Values and operators combine to make Expressions.

Type **int**

- The declaration in the above code specifies that the variable named **number1** is of type **int**—it will hold integer values (whole numbers such as 7, –11, 0 and 31914). The range of values for an int is –2,147,483,648 (int.MinValue) to +2,147,483,647 (int.MaxValue).



Exercise:

```
Number 1: 6
Number 2: 8
Number 3: 10
The Sum of three numbers is 24
```