# Table of Contents

Table of Contents

[Table of Contents 1](#_Toc47530747)

[Foundational Programming 2](#_Toc47530748)

[Variables and Types 2](#_Toc47530749)

[If / Else and Case Statements 2](#_Toc47530750)

[Arrays 2](#_Toc47530751)

[Methods and Functions 2](#_Toc47530752)

[Collections (List, Dictionary/HashSet) 2](#_Toc47530753)

[Looping 2](#_Toc47530754)

[Enums 2](#_Toc47530755)

[Classes – Public / Private / Protected Methods 2](#_Toc47530756)

[Classes – Static Methods vs Instance Methods 2](#_Toc47530757)

[Classes – Constructor and Passing Values through a Constructor 2](#_Toc47530758)

[Classes – Private / Protected Variables 2](#_Toc47530759)

[Inheritance with Classes (and why it should usually be avoided) 2](#_Toc47530760)

[Composition with Classes (why this is usually better than Inheritance) 2](#_Toc47530761)

[Interfaces 2](#_Toc47530762)

[Exceptions 2](#_Toc47530763)

[Generics 3](#_Toc47530764)

[Fourth subheading 3](#_Toc47530765)

[Second Heading 4](#_Toc47530766)

[Subheading 4](#_Toc47530767)

[Subheading 4](#_Toc47530768)

[Third Heading 5](#_Toc47530769)

[Subheading 5](#_Toc47530770)

[Subheading 5](#_Toc47530771)

[Subheading 5](#_Toc47530772)

[Fourth Heading 6](#_Toc47530773)

[Subheading 6](#_Toc47530774)

[Subheading 6](#_Toc47530775)

[Fifth Heading 7](#_Toc47530776)

[Subheading 7](#_Toc47530777)

[Subheading 7](#_Toc47530778)

[Subheading 7](#_Toc47530779)

[Bibliography 8](#_Toc47530780)

[Appendix 9](#_Toc47530781)

# Foundational Programming

## Variables and Types

Variables are storage containers; types are what is expected/allowed to be stored. Use variable type (e.g. int or string) or ‘var’ to declare variable at first use. ‘var’ determines the type of variables expected by the program, whereas ‘int’ or ‘string’ are determined by the programmer.

## If / Else and Case Statements

Tests specific conditions; makes decisions to determine if the condition is true or false. If / else is used where there are a small number of conditions that could be met; switch (case statements) are used where there are many conditions.

## Arrays

An array is a. Can be called as a specific type (int[]) or by the use of Object[].

## Methods and Functions

Write your info here

## Collections (List, Dictionary/HashSet)

Data types

## Looping

Write your info here

## Enums

Write your info here

## Classes – Public / Private / Protected Methods

Info from Slack fma-proteges-pub channel: (Post from Tom Hatcher; reply Mahmoud Hanafy.

|  |
| --- |
| * private = you don't want any other class to access this. * internal = you want other classes within the same project (assembly) to access them, but shouldn't be visible to the outer world * public = visible to the outer world; this is typically used for the things consumed by other projects. |

## Classes – Static Methods vs Instance Methods

Write your info here

## Classes – Constructor and Passing Values through a Constructor

Write your info here

## Classes – Private / Protected Variables

Write your info here

## Inheritance with Classes (and why it should usually be avoided)

Write your info here

## Composition with Classes (why this is usually better than Inheritance)

Write your info here

## Interfaces

Write your info here

## Exceptions

Write your info here

## Generics

Write your info here

## Naming Conventions / Identifiers

## C# programs use PascalCase for type names, namespaces, and all public members. In addition, the following conventions are common:

## Interface names start with a capital I.

## Attribute types end with the word Attribute.

## Enum types use a singular noun for non-flags, and a plural noun for flags.

## Identifiers should not contain two consecutive ‘\_’ characters. Those names are reserved for compiler generated identifiers.

Keywords – cannot be used unless prefixed with @:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ‘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’ | ‘overide’ | ‘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’ | ‘volatile’ | ‘while’ |  |

## Namespace

* can help control scope of class and method names in larger projects - a way to present program elements that are exposed to other programs
* keyword (lowercase) to declare a namespace
* used by .NET to organise its many classes
* must be a valid C# identifier name
* used within (internal organisation) a program

## Access Modifiers

Write your info here

## Subheading

Write your info here

# Second Heading

## Subheading

Write your info here

## Subheading

Write your info here

# Third Heading

## Subheading

Write your info here

## Subheading

Write your info here

## Subheading

Write your info here

# Fourth Heading

## Subheading

Write your info here

## Subheading

Write your info here

# Fifth Heading

## Subheading

Write your info here

## Subheading

Write your info here

## Subheading

Write your info here

# Bibliography

[Microsoft.com documents C#](https://docs.microsoft.com/en-us/dotnet/csharp/)

[W3 Schools](https://www.w3schools.com/cs/default.asp)

# Appendix