computrain
IT & Digital Skills

C# Programming

www.computrain.nl

Topics

Visual Studio

C# Syntax

Console Application

Windows Forms Application

Event Driven Development

Variable Types

Functions

Arrays

Collections

Object Oriented Programming

.NET Framework

Database Access

MVC.NET

API

Introduction

Material

Computrain Workbook

C# Programming Fundamentals

Exercises & Solutions

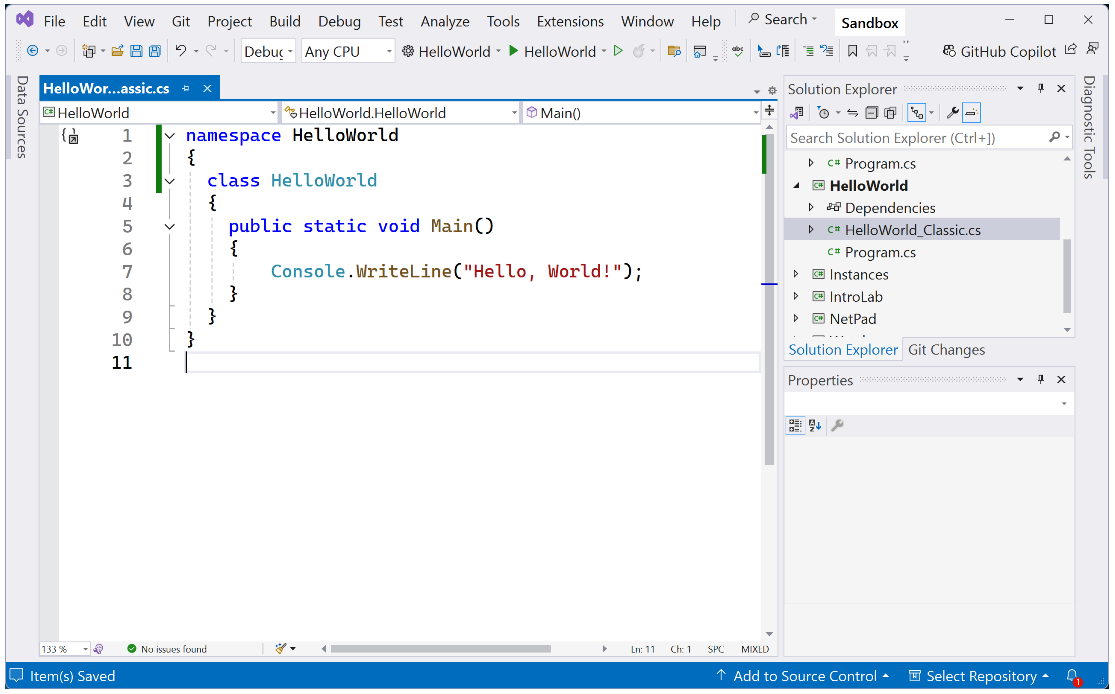
www.computrain.nl/oefenbestanden

Useful Links

- https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/
- https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/
- https://riptutorial.com/Download/csharp-language.pdf

Visual Studio 2022

- Menu
- Toolbar
- Toolbox
- Designer Window
- Code Window
- Solution Explorer
- Projects
- Properties Window

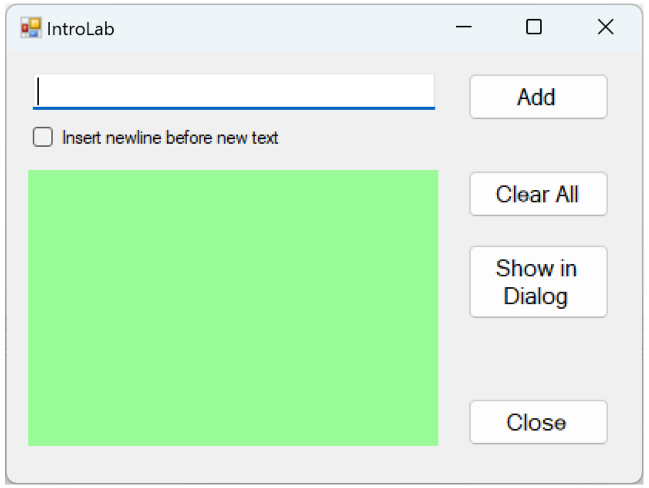


```
namespace HelloWorld
{
    class HelloWorld
    {
        public static void Main()
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

▶ Demo page 13, 14 & 15

Exercise

Exercise 2.4, page 15



Help & Reference

- Intellisense
- Online Reference
- Object Browser

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/

C#

C# Features

- C# is a cross-platform general purpose language
- C# is a strongly typed language
- C# is based on object-oriented principles
- C# incorporates features from other paradigms, not least functional programming
- C# is the most popular .NET language.
- C# apps use the extensive runtime libraries provided by the .NET SDK

Program structure

- using directives that reference a given namespace
- class
- public static void Main()

```
using System;

class Hello
{
    public static void Main()
    {
        // This line prints "Hello, World"
        Console.WriteLine("Hello, World");
    }
}
```

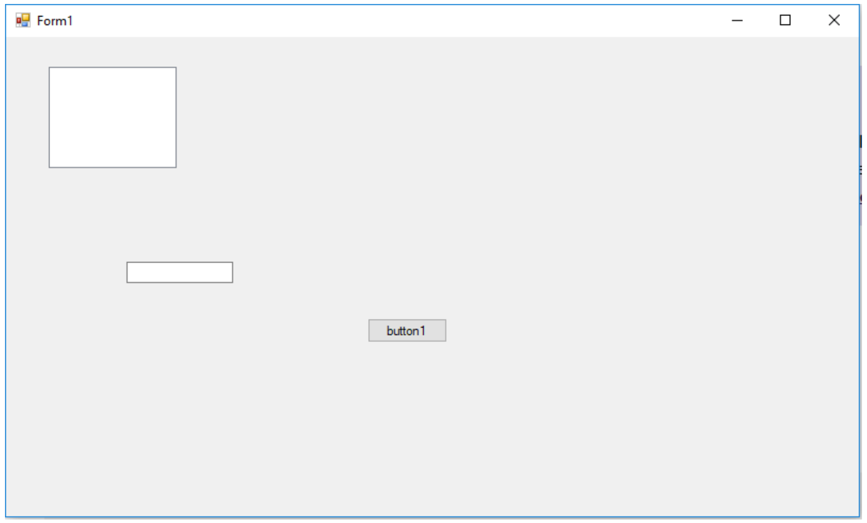
When you use top-level statements, the compiler synthesizes the containing class and method for the program's entry point.

```
Console.WriteLine("Hello, World");
```

Event driven programming

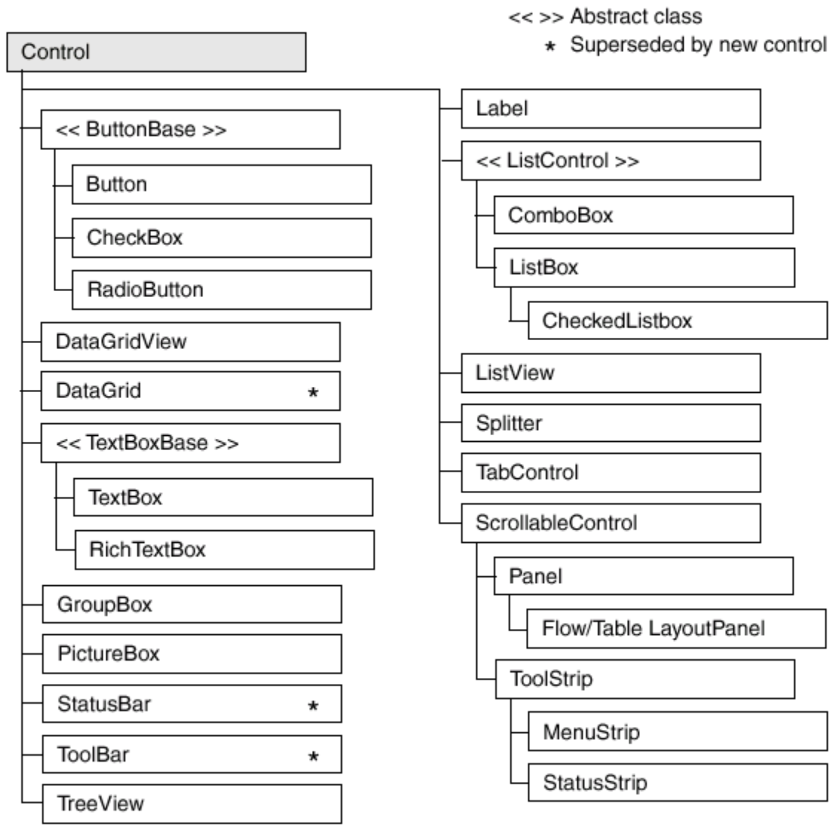
Windows Forms

Windows Forms is a Graphical User Interface(GUI) class library which is bundled in *.Net Framework*.



Windows Forms Controls

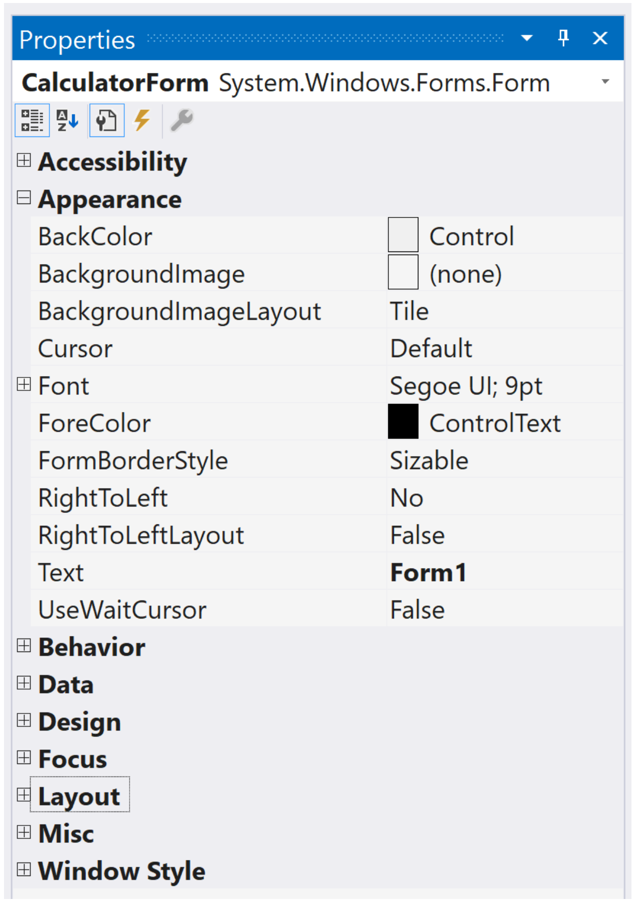
- Label
- TextBox
- Button
- CheckBox
- RadioButton
- ListBox
- ListView
- ComboBox
- DataGridView
- ...



https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/controls-to-use-on-windows-forms

Properties

- Foreground & background colors
- Font
- Anchor
- Dock – Top, Right, Bottom, Left, Fill
- Padding
- Margins
- AutoSize



⊞ **Accessibility**
⊟ **Appearance**

BackColor	☐ Control
BackgroundImage	☐ (none)
BackgroundImageLayout	Tile
Cursor	Default
⊞ Font	Segoe UI; 9pt
ForeColor	■ ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
Text	**Form1**
UseWaitCursor	False

⊞ **Behavior**
⊞ **Data**
⊞ **Design**
⊞ **Focus**
⊞ **Layout**
⊞ **Misc**
⊞ **Window Style**

Events

Mouse Events

- MouseClick
- MouseDoubleClick
- MouseDown
- MouseEnter
- MouseHover
- MouseLeave
- MouseMove
- MouseUp
- MouseWheel
- Click

Keyboard Events

- KeyDown
- KeyPress
- KeyUp

Property changed events

Other events

Standard controls & C# syntax

C# Syntax

- Case sensitive
- Statements end with semi-colon ;
- Comments with // or /* and */
- Code block with curly brackets { and }

https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview

Variables

- Declaration with type
- Initialization
- Type inference with var
- Dynamic with dynamic

```
int x = 8, y = 6;
double quotient = x / y;
```

int	32-bits number
short	16-bits number
long	64-bits number
float	32-bits floating point number
double	64-bits floating point number
decimal	128-bits floating point number
string	text
char	one single character
bool	**true** or **false**
ushort	positive short
uint	positive int
ulong	positive long

Naming convention

An identifier is the name you assign to a type (class, interface, struct, delegate, or enum), member, variable, or namespace

- PascalCasing
- camelCasing for method parameters and local variables
- Use meaningful and discriptive names
- Prefer clarity over brevity
- Coding conventions are essential for code readability and consistency

https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names

Keywords

abstract	delegate	if	override	this
as	do	implicit	protected	throw
base	double	in	public	true
bool	else	int	readonly	try
break	enum	interface	ref	typeof
byte	event	internal	return	uint
case	explicit	is	sbyte	ulong
catch	extern	lock	sealed	unchecked
char	false	long	short	unsafe
checked	finally	namespace	sizeof	ushort
class	fixed	new	stackalloc	using
const	float	null	static	virtual
continue	for	object	string	void
decimal	foreach	operator	struct	volatile
default	goto	out	switch	while

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/keywords

Contextual Keywords

add	field file	not	select
allows	from	notnull	set
alias	get	nuint	unmanaged
and	global	on	unmanaged
ascending	group	or	value
args	init	orderby	var
async	into	partial	when
await	join	partial	where
by	let	record	where
descending	managed	remove	with
dynamic	nameof	required	yield
equals	nint	scoped	

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/statement-keywords

Statement keywords

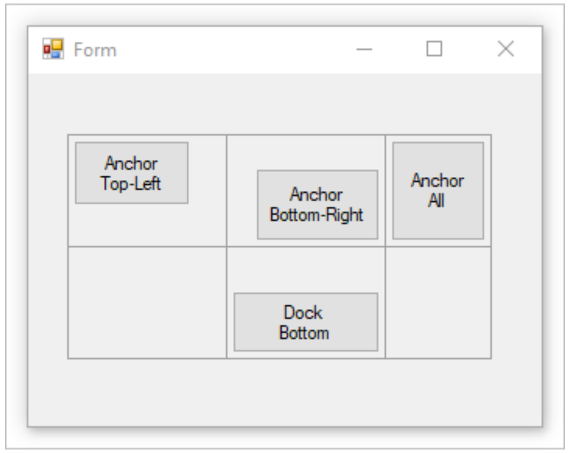
Category	**C# keywords**
Selection statements	if, switch
Iteration statements	do, for, foreach, while
Jump statements	break, continue, goto, return
Exception-handling statements	throw, try-catch, try-finally, try-catch-finally
checked and unchecked statements	checked, unchecked
fixed statement	fixed
lock statement	lock
yield statement	yield

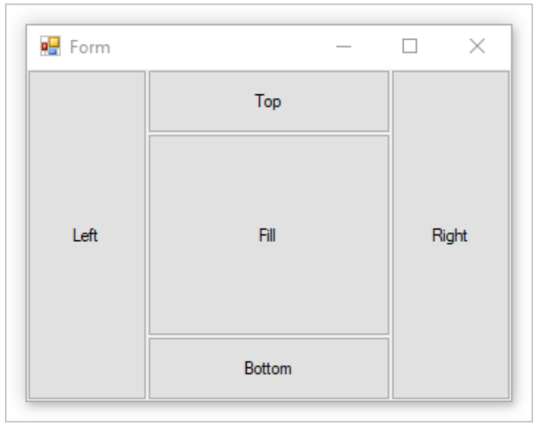
https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/statement-keywords

Layout

Control placement in Windows Forms is determined not only by the control, but also by the parent of the control.

- Fixed position and size
- FlowLayoutPanel
- TableLayoutPanel





Event Handler

Function

Anonymous Function

Lambda Expression

```
btnAdd.Click += HandleButtonClick
```

```
private void HandleButtonClick(object sender, EventArgs e)
{
   lblAllText.Text += "\n";
   lblAllText.Text += txtNewText.Text;
   txtNewText.Focus();
   txtNewText.Clear();
}
```

```
btnAdd.Click += delegate(object sender, EventArgs e) { ... }
```

```
btnAdd.Click += (sender, e) => { ... }
```

.NET API's

Fundamental types

- byte
- int
- long
- float
- double
- decimal

Datastructures

- Array
- List
- Dictionary
- Uri
- DateTime

Utility APIs

- HttpClient
- XDocument
- StreamReader
- StreamWriter

App-model APIs

- ASP.NET
- .NET MAUI
- Windows Desktop
- Windows Forms

https://learn.microsoft.com/en-us/dotnet/api/

Object Oriented

- Classes and Objects
- Attributes
- Methods

```
lblAllText.Text = "";
```

```
txtNewText.Clear();
```

▶https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/

C#

```
private void btnAdd_Click(object sender, EventArgs e)
{
   if (chkNewLine.Checked)
   {
      lblAllText.Text += "\n";
   }
   lblAllText.Text += txtNewText.Text;
   txtNewText.Focus();
   txtNewText.Clear();
}
```

Enumerated types

Type definition with 'named values'

```
enum OrderState
{
    Requested,
    InProcess,
    OnHold,
    Aborted,
    Finished
}
```

▶ Demo page 30

Application Settings

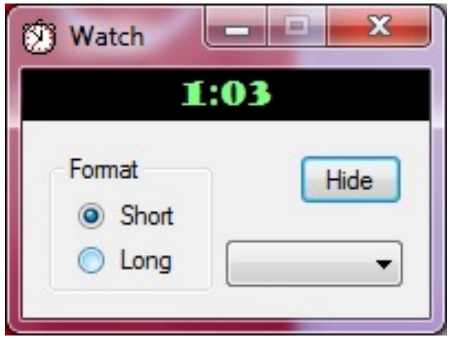
Properties.Settings.Default

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <userSettings>
        <Demo03.Properties.Settings>
            <setting name="Text" serializeAs="String">
                <value>Start</value>
            </setting>
        </Demo03.Properties.Settings>
    </userSettings>
</configuration>
```

▶ Demo page 31

Exercise 4.4: Watch

Exercise 4.4, page 31



Visual Studio

Integrated Development Environment

'The Compiler is Your Friend'

Aligning controls

Properties van selected controls

AutoComplete

Matching brackets

Reformat

Jump to declaration

Refactor

Copying a line

Regions

Tasks

Commenting blocks

Keyboards shortcuts

Toolbox code

Code snippets

Exercise

Exercise 5.1, page 39

.NET Framework

.NET Framework

Solution for "DLL Hell"

.NET talen - C#, VB.NET

CLI - Common Language Infrastructure

GAC - Global Assembly Cache

C:\Windows\assembly

Namespaces

CLR - Common Language Runtime
Common Type System

MSIL - Microsoft Intermediate Language
JIT - Just In Time compilation

▶ Demo page 46, 49 & 48

Functions

Functions

Definition

Arguments, Parameters

Pass by Value versus Pass by Reference - ref

Output parameters - out

Return value - return

```
int TelOp(int a, int b = 4)
{
   int som = a + b++;
   return som;
}
```

```
int x = 8, y = 6;
int i = TelOp(x, y);
```

▶ Demo page 57, 59, 60, 61, 63 & 63

Console

Console window

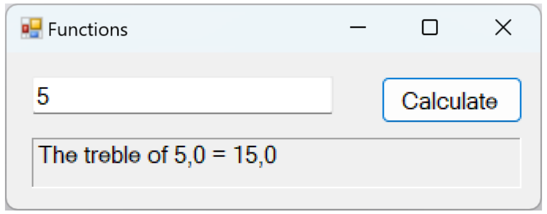
Console.Writeline(".......")

Formatting

String.Format("Number {0}", 5);
String.Format("Number {0,5:F2}", 3.1415);

Exercise

Exercise 7.11 page 64



```
5
The treble of 5,0 = 15,0
```

Instances, dialogs & scope

Variables

Declaration of variables

Value type

- simple types: int, short, long, bool, float, double
- enumerated type
- struct types

Reference type

- classes

Instance

null

Garbage collection

this

```
a = new Label();
```

▶ Demo page 72

Dialogs

ShowDialog

Show

DialogResults

```
Form1 f;
f = New Form1();
f.ShowDialog();
f.Show();
```

▶ Demo page 74

Scope

Local variables

using

Static functies

Access modifiers

public

protected

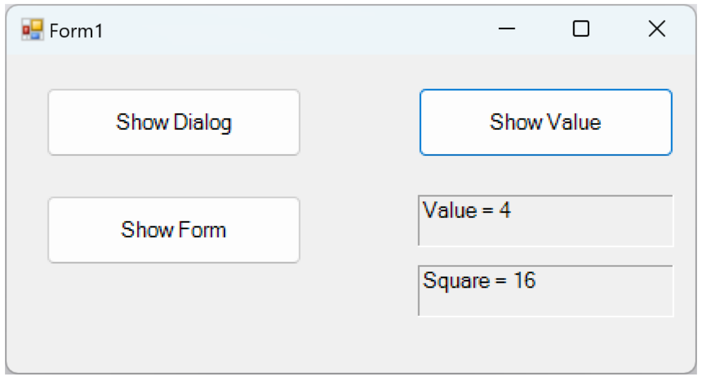
internal

private

file

Exercise

Exercise 8.9 page 77



Value = 4

Square = 16

Arrays

Arrays

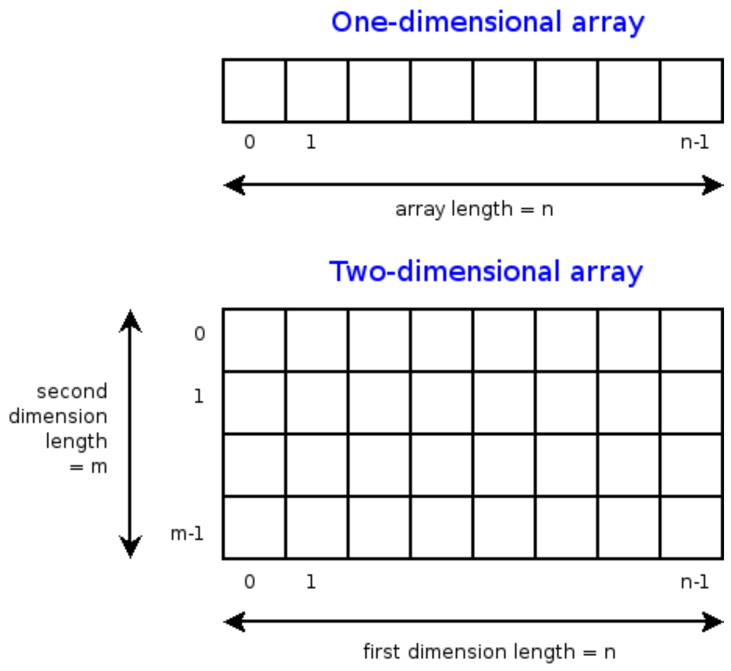
Declaratie

Array Members

Initialisatie

Meerdimensionele arrays

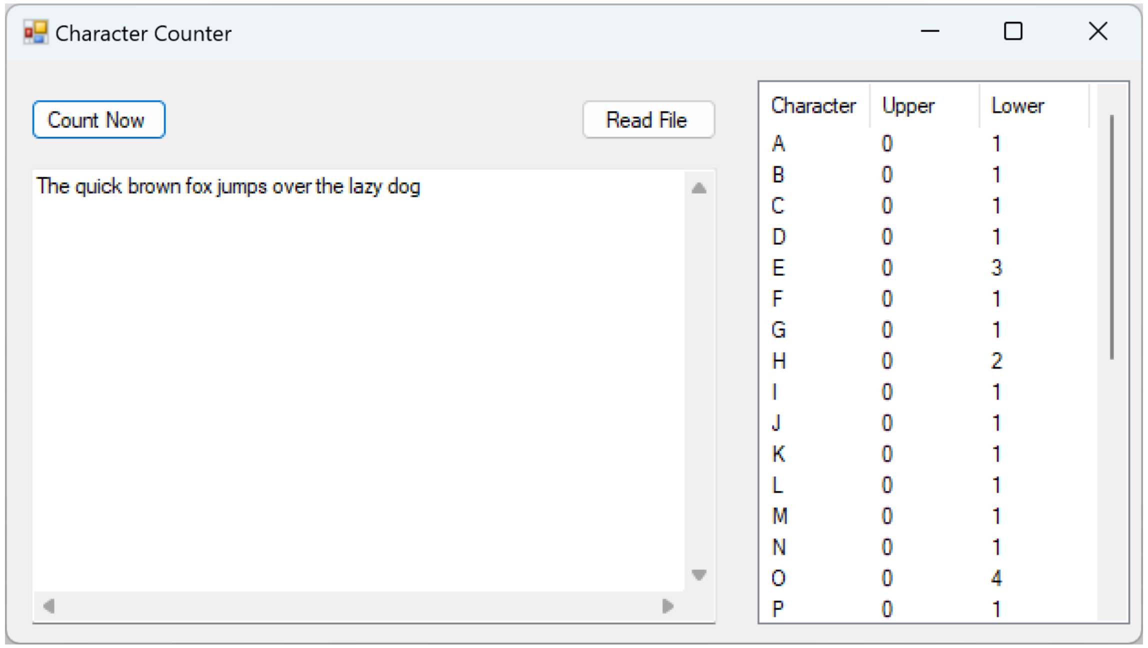
```
int[] Numbers = new int[100];
int[] Numbers = { 1, 2, 3, 4 };
```



▶ Demo page87

Exercise

Exercise 9.5 page 88



The quick brown fox jumps over the lazy dog

Character	Upper	Lower
A	0	1
B	0	1
C	0	1
D	0	1
E	0	3
F	0	1
G	0	1
H	0	2
I	0	1
J	0	1
K	0	1
L	0	1
M	0	1
N	0	1
O	0	4
P	0	1

Structures & Collections

Structures

fields

value - type

accessing members

```
private struct Person
{
   public string Name;
   public string Address;
   public DateTime DateOfBirth;
}

Person p;
p.Name = "Peter";
```

▶ Demo page 94

Generics

ArrayList

List<T>

List.Add()

List.Delete()

```
Person p1, p2;

List<Person> collection = new List<Person>();

collection.Add(p1)
collection.Add(p2)

foreach(Person p in collection)
{
   Console.Writeln(p.Name);
}
```

▶ Demo page 96

Events

Event property

Triggers code

▶ Demo page 97

LINQ

Language INtegrated Query

SQL => databases

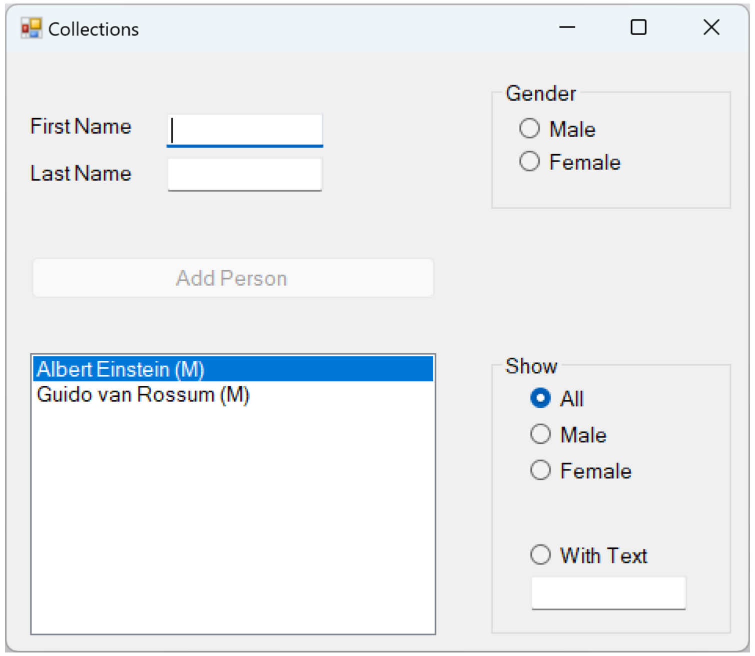
LINQ => collections

```
int[] scores = [97, 92, 81, 60];

var scoreQuery = from score in scores
                 where score > 80
                 orderby score
                 select score;

foreach (var i in scoreQuery)
{
    Console.Write(i + " ");
}
```

▶ Demo page 98

Exercise

Exercise 10.7 page 98

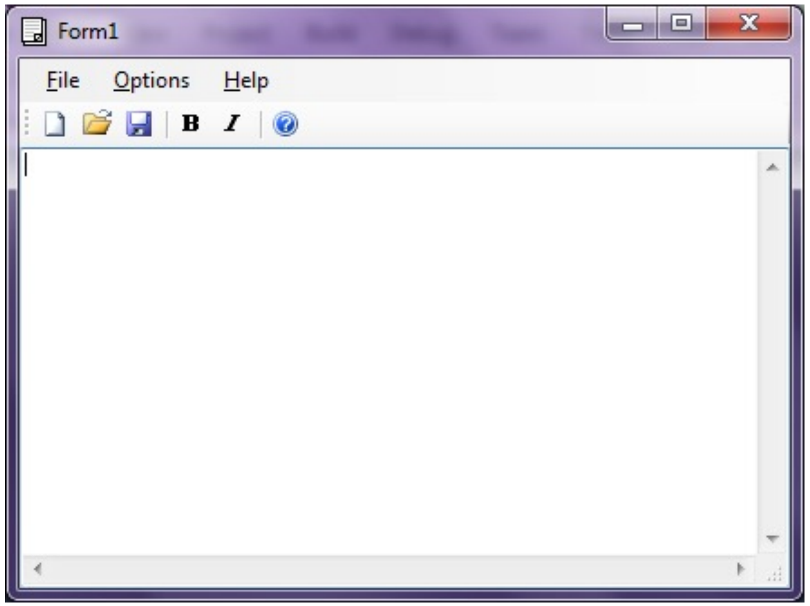


Albert Einstein (M)
Guido van Rossum (M)

Exercise Netpad

Exercise

Exercise 11 page 107


Form1
File
Options
Help
B
I