MCTE 4327

Software Engineering

Week 03 Primitive data types

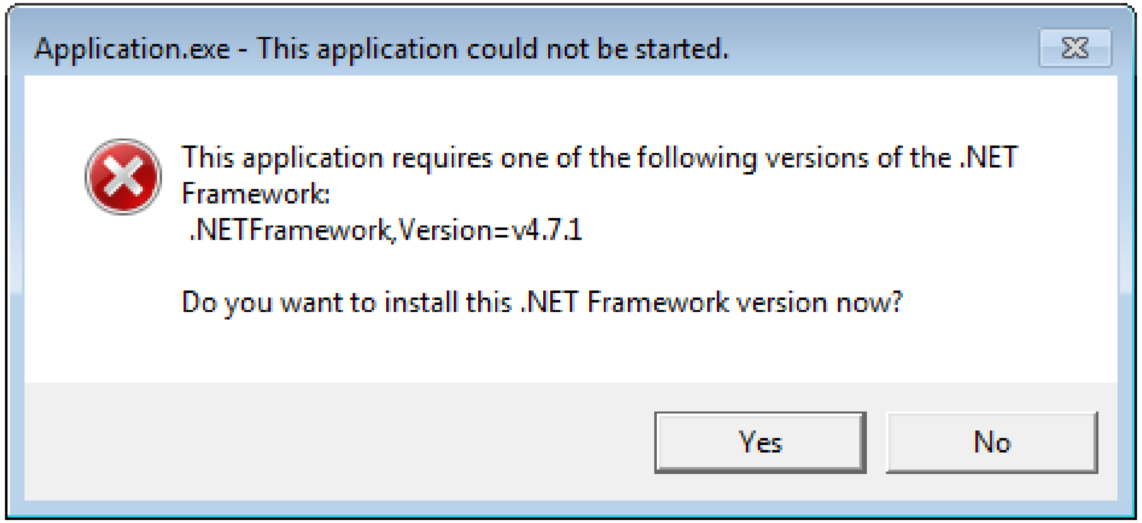
Outline

- Introduction to C#
- Data types
- Values vs References
- Namespaces

C#

- C# is a general-purpose, type-safe, object-oriented programming language. The goal of the language is programmer productivity. It has the same syntax as C++.
- The language balances simplicity, expressiveness, and performance.
- C# utilized .NET framework.
- C# programs are compiled into byte-code (not pure machine language). When the application gets run, the .NET framework performs actual compilation. It is called JIT (just in time compilation).
- JIT compilation allows C# programs to be run efficiently on any platform.
- The .NET framework performs automatic memory management.

- Historically C# was used almost entirely for writing code to run on Windows platform only.
- However now C# programs can be run on multiple desktop platforms (Linux, Mac OS) and mobile platforms (Android, IOS) using manifestations of .NET framework called Mono and Xamarin.
- C# programs need .NET framework its derivatives to be run.

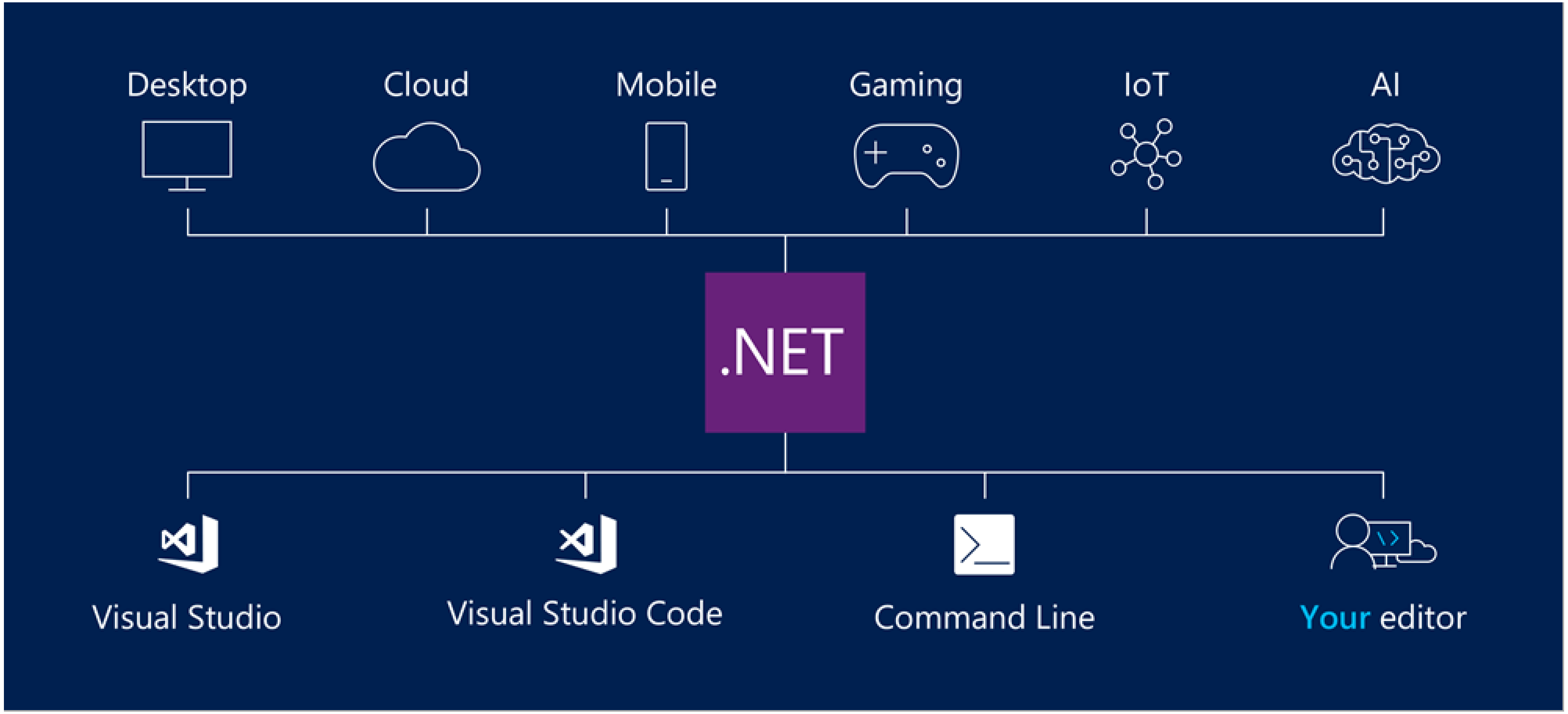


Application.exe - This application could not be started.

This application requires one of the following versions of the .NET Framework:
.NETFramework,Version=v4.7.1

Do you want to install this .NET Framework version now?

.NET Framework



IDE

Performance: C++ vs C# vs Python

C++ and C# code (same)

```
double x=0;
for (double i = 0.1; i < 100000; i++)
{
   for (double j = 0.1; j < 100000; j++)
   {
         double a = 3.2;
         double b = 4.4;
         double c = 13.8;

         x += (((a * b) + (a * b)) / (a * b)) * c + i + j;
   }
}
```

Python 3.6 code

```
x = 0
for i_ in range(0, 100000):
    for j_ in range(0, 100000):
        i = i_ + 0.1
        j = j_ + 0.1
        a = 3.2
        b = 4.4
        c = 13.8
        x = x + (((a * b) + (a * b)) / (a * b)) * c + i + j
```

Results on the same PC

Time taken to execute the previous code

C++:	39.976 seconds
C# :	30.101 seconds
Python:	6790.565 seconds (almost 2 hours)

C# is often faster than un-optimized C++ code. The .NET framework performs JIT compilation in an efficient way to run on a particular hardware.

C++ Hello World

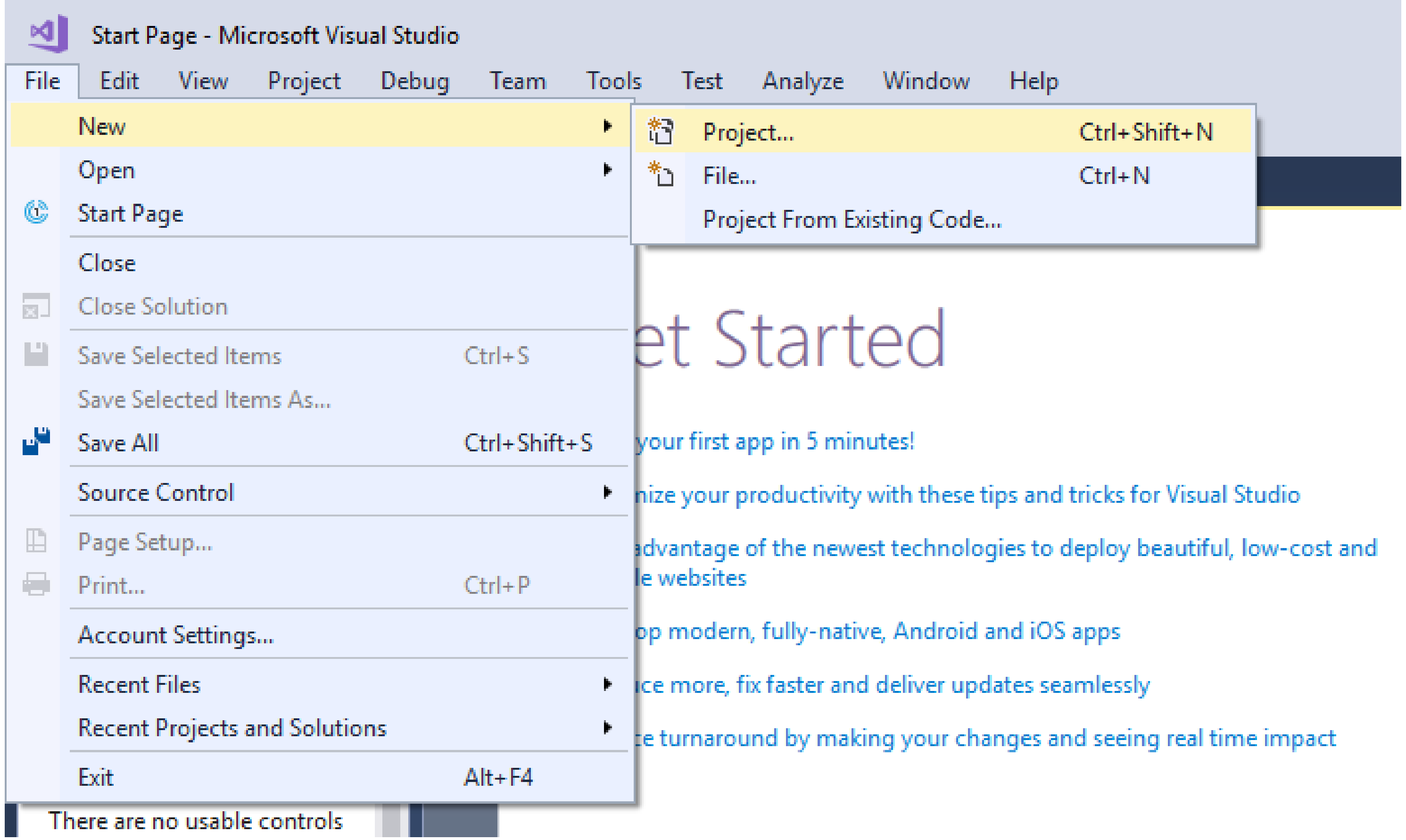
```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello from MCT 4237!";
    system("pause");
    return 0;
}
```

The cout function is stored inside the std namespace. The following code does not explicitly declare the namespace.

```
#include <iostream>

int main()
{
    std::cout << "Hello from MCT 4237!";
    system("pause");
    return 0;
}
```

C# Hello World

New -> Project



et Started

your first app in 5 minutes!

nize your productivity with these tips and tricks for Visual Studio

advantage of the newest technologies to deploy beautiful, low-cost and le websites

op modern, fully-native, Android and iOS apps

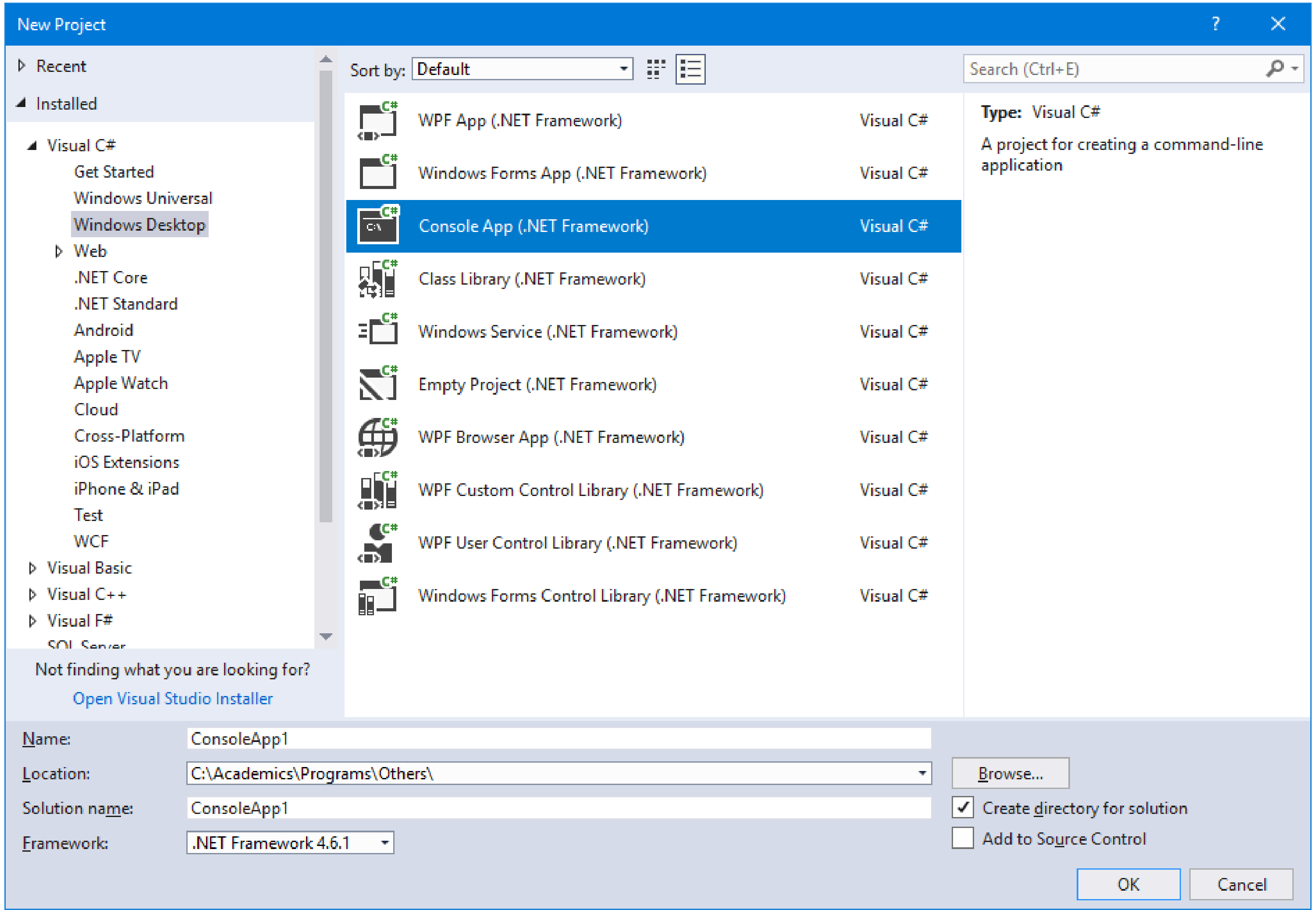
ice more, fix faster and deliver updates seamlessly

e turnaround by making your changes and seeing real time impact

There are no usable controls

C# Hello World

Select “Console App”



New Project

Sort by: Default

Search (Ctrl+E)

- Recent
- Installed
  - Visual C#
    - Get Started
    - Windows Universal
    - Windows Desktop
    - Web
    - .NET Core
    - .NET Standard
    - Android
    - Apple TV
    - Apple Watch
    - Cloud
    - Cross-Platform
    - iOS Extensions
    - iPhone & iPad
    - Test
    - WCF
  - Visual Basic
  - Visual C++
  - Visual F#
  - SQL Server

Not finding what you are looking for?

Open Visual Studio Installer

WPF App (.NET Framework)	Visual C#
Windows Forms App (.NET Framework)	Visual C#
Console App (.NET Framework)	Visual C#
Class Library (.NET Framework)	Visual C#
Windows Service (.NET Framework)	Visual C#
Empty Project (.NET Framework)	Visual C#
WPF Browser App (.NET Framework)	Visual C#
WPF Custom Control Library (.NET Framework)	Visual C#
WPF User Control Library (.NET Framework)	Visual C#
Windows Forms Control Library (.NET Framework)	Visual C#

Type: Visual C#

A project for creating a command-line application

Name: ConsoleApp1

Location: C:\Academics\Programs\Others\

Solution name: ConsoleApp1

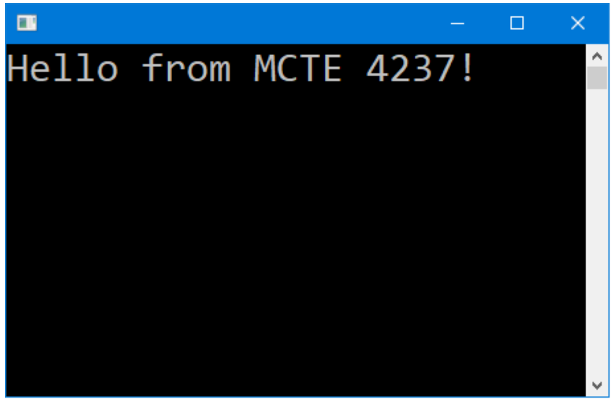
Framework: .NET Framework 4.6.1

☑ Create directory for solution

□ Add to Source Control

C# Hello World

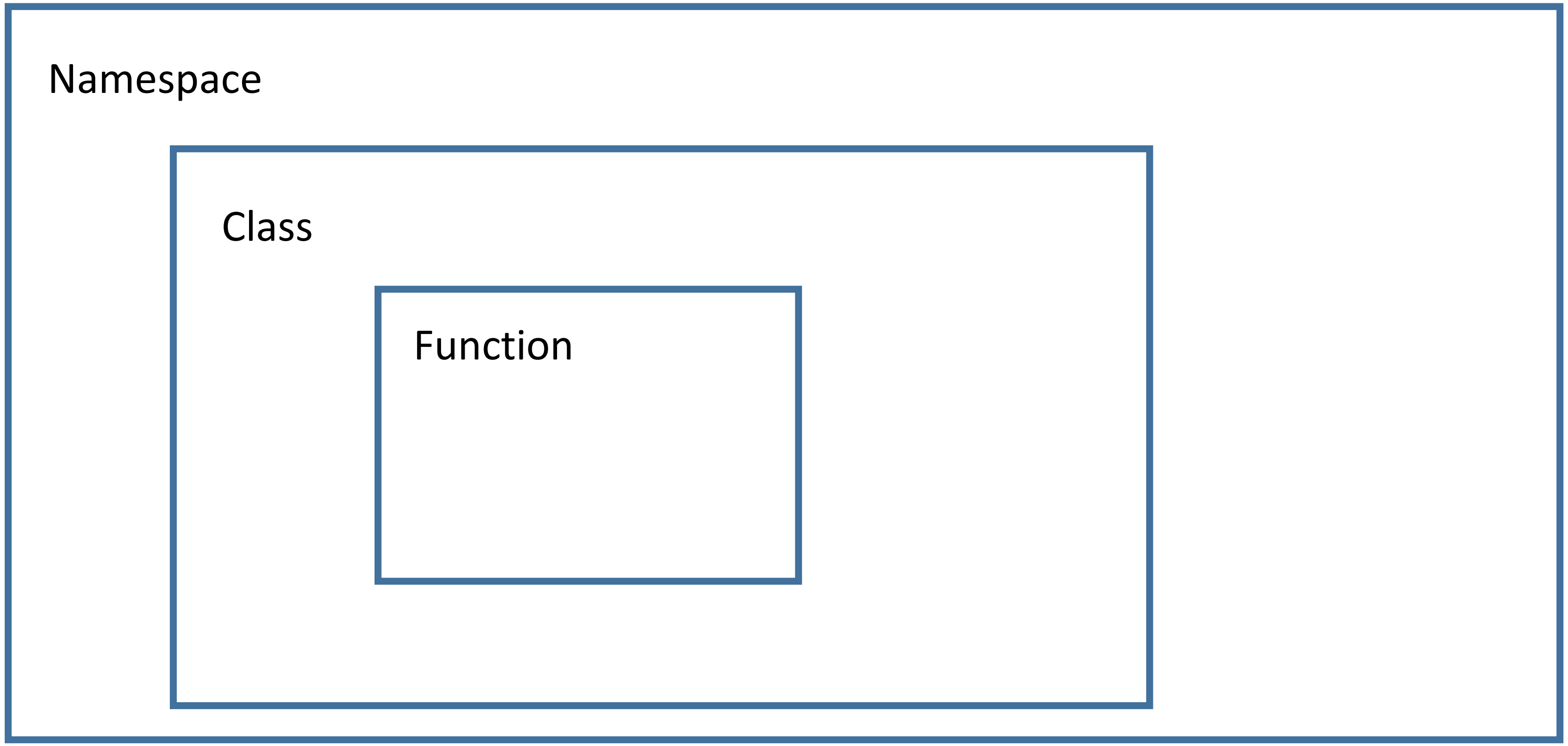
```
using System;   //This is the namespace that has "Console" object for reading and writing text

namespace HelloWorld  //This is just the namespace of your project (same name)
{
    class Program //According to OOP philosophy, every function must be inside a class
    {
        static void Main()  //The main function is static
        {
            Console.WriteLine("Hello from MCTE 4237!");
            Console.ReadLine();
        }
    }
}
```



```
Hello from MCTE 4237!
```

C# organization

- Every function must be inside a class.
- Every class must be inside a namespace.
- Variables can be inside a class or a function.



Console object

Console.WriteLine is equivalent to cout in C++

Console.ReadLine is equivalent to cin in C++

Primitive data types

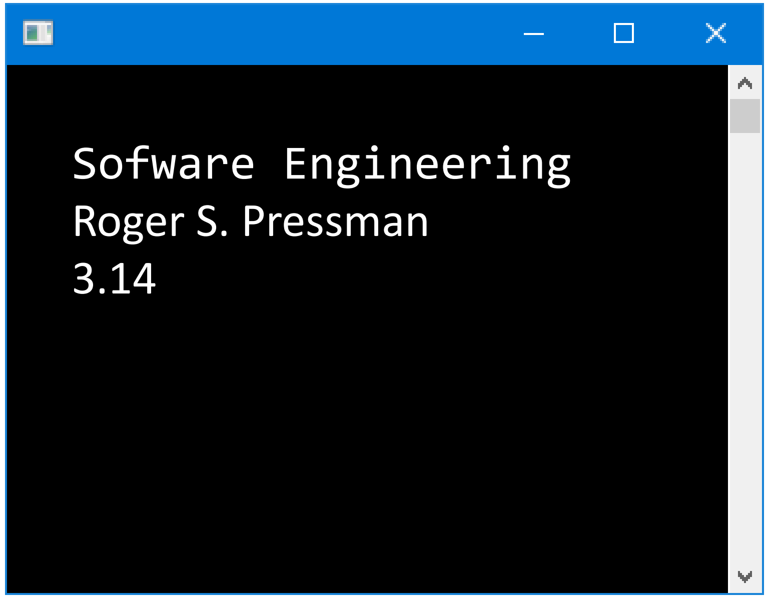
Data type	Description	Size (bits)	Range (values)
byte	Unsigned integer	8	0 to 255
sbyte	Signed integer	8	-128 to 127
short	Signed integer	16	-32,768 to 32,767
ushort	Unsigned integer	16	0 to 65,535
int	Signed integer	32	-2,147,483,648 to 2,147,483,647
uint	Unsigned integer	32	0 to 4294967295
long	Signed integer	64	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	Unsigned integer	64	0 to 18,446,744,073,709,551,615
float	Single-precision floating point type	32	-3.402823e38 to 3.402823e38
double	Double-precision floating point type	64	-1.79769313486232e308 to 1.79769313486232e308
decimal	Precise fractional with 29 significant digits	128	(+ or -)1.0 x 10e-28 to 7.9 x 10e28
char	A single Unicode character	16	Unicode symbols used in text
bool	Logical Boolean type	8	True or False
object/var	Base type of all other types	depends	
string	A sequence of characters	depends	
DateTime	Represents date and time	64	0:00:00am 1/1/01 to 11:59:59pm 12/31/9999

Struct

Struct is a composite data structure that can have many members.

```
struct Book
{
    public double price;
    public string title;
    public string author;
}

static void Main()  //The main function is static
{
    Book a; //variable a is of book type
     a.price = 3.14;
    a.title = "Sofware Engineering";
    a.author = "Roger S. Pressman";

    Console.WriteLine(a.title);
    Console.WriteLine(a.author);
    Console.Write(a.price);
    Console.Read();
}
```

Output



```
Sofware Engineering
Roger S. Pressman
3.14
```

Class

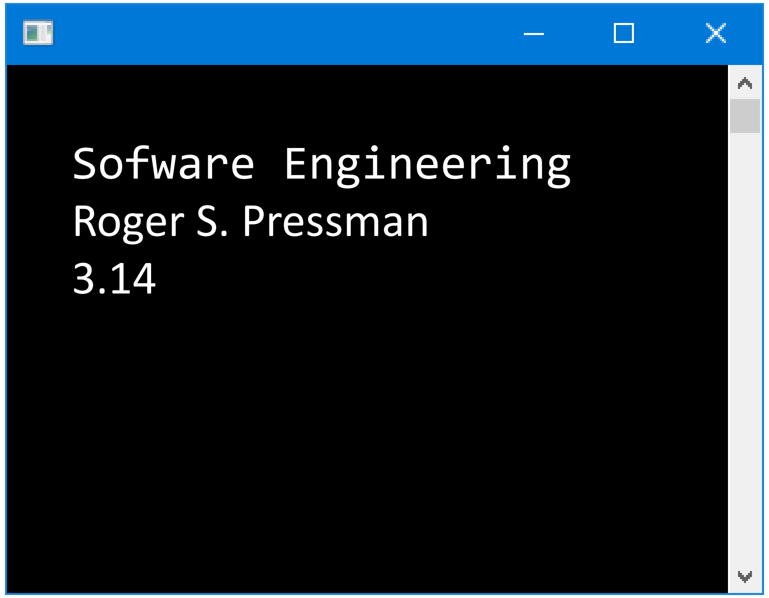
Class is like struct, but with some differences (we will discuss later)

```
class Book
{
    public double price;
    public string title;
    public string author;
}

static void Main()  //The main function is static
{
    Book a = new Book();
     a.price = 3.14;
    a.title = "Sofware Engineering";
    a.author = "Roger S. Pressman";

    Console.WriteLine(a.title);
    Console.WriteLine(a.author);
    Console.Write(a.price);
    Console.Read();
}
```

One difference:

To use a class, you need to use "new" keyword.

Output



```
Sofware Engineering
Roger S. Pressman
3.14
```

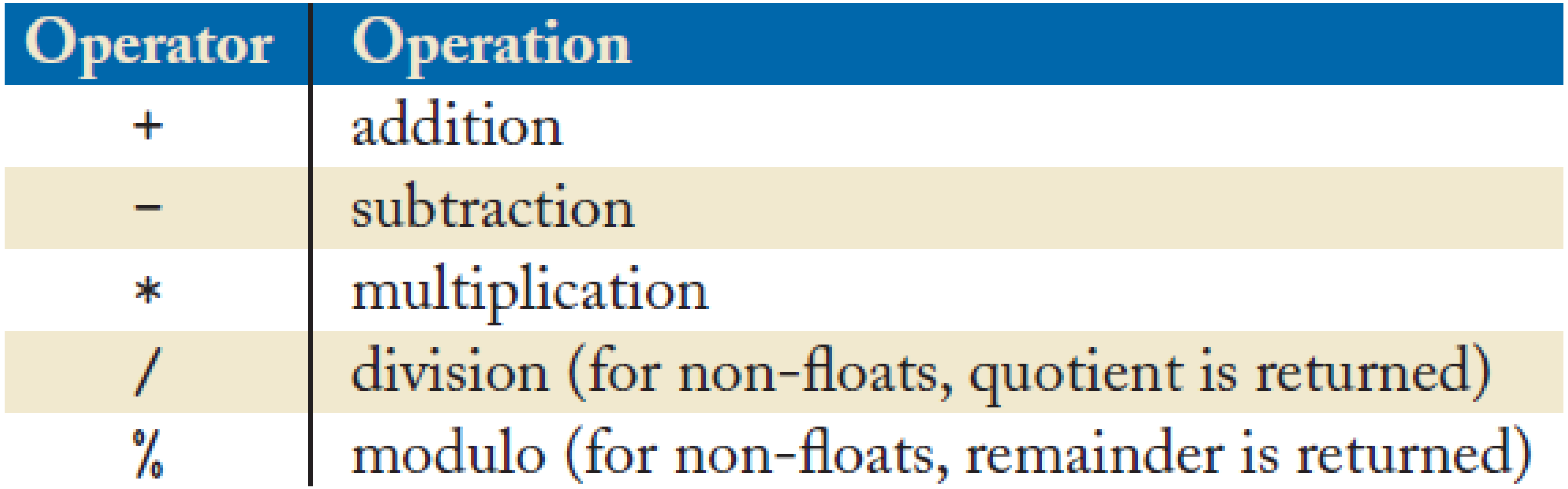
Enum

The enum keyword is used to declare an enumeration, a distinct type that consists of a set of named constants called the enumerator list.

```
enum Day { Sun, Mon, Tue, Wed, Thu, Fri, Sat };

static void Main()
{
    Day Today = Day.Sun;
    Day Yesterday = Day.Sat;
}
```

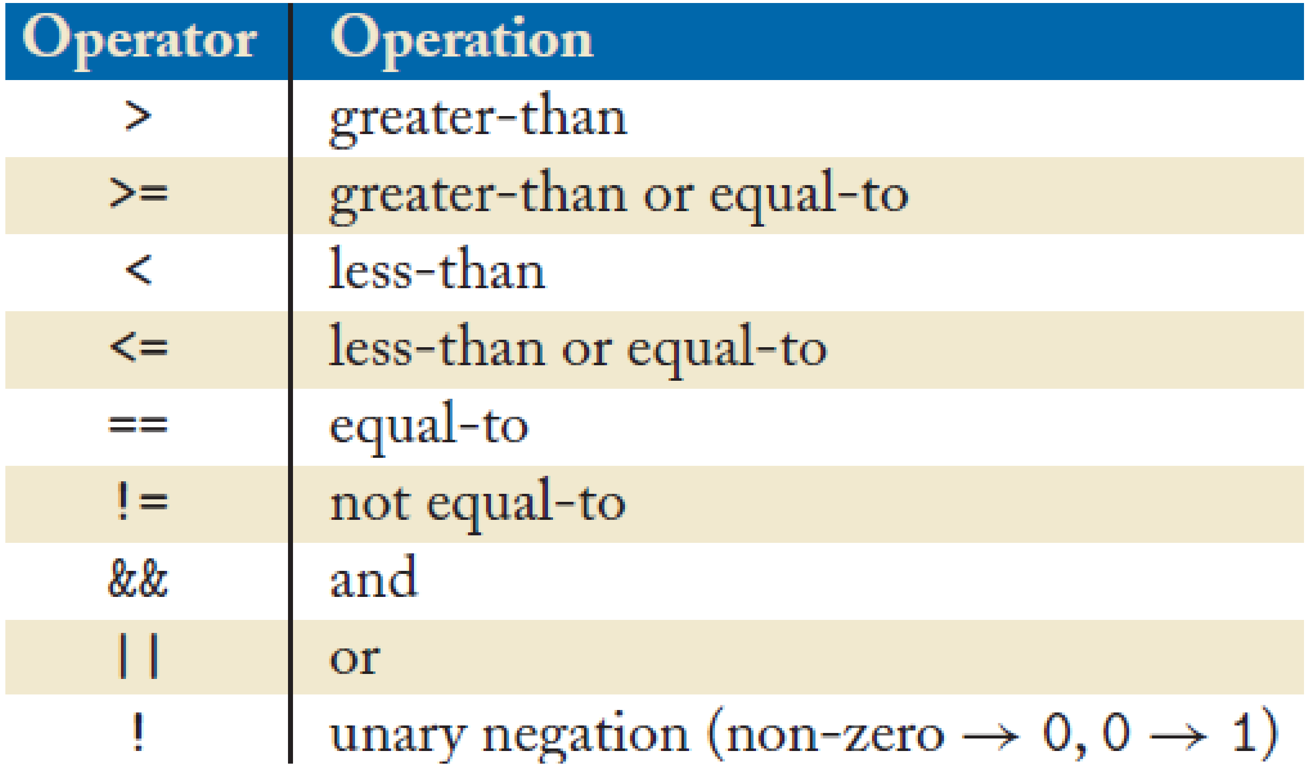
We use enum when we want the variables to be discrete (rather than continuous like numbers and strings)

Mathematical operators



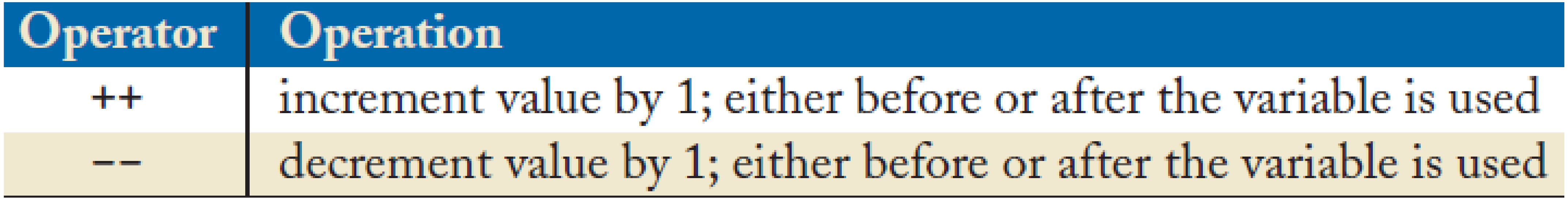
Operator	Operation
+	addition
–	subtraction
*	multiplication
/	division (for non-floats, quotient is returned)
%	modulo (for non-floats, remainder is returned)

Comparison operators



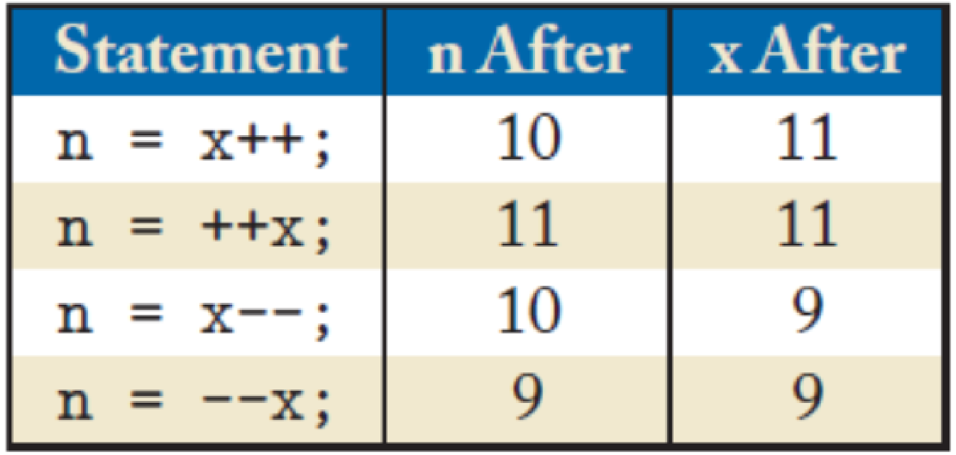
Operator	Operation
>	greater-than
>=	greater-than or equal-to
<	less-than
<=	less-than or equal-to
==	equal-to
!=	not equal-to
&&	and
||	or
!	unary negation (non-zero → 0, 0 → 1)

Increment operators



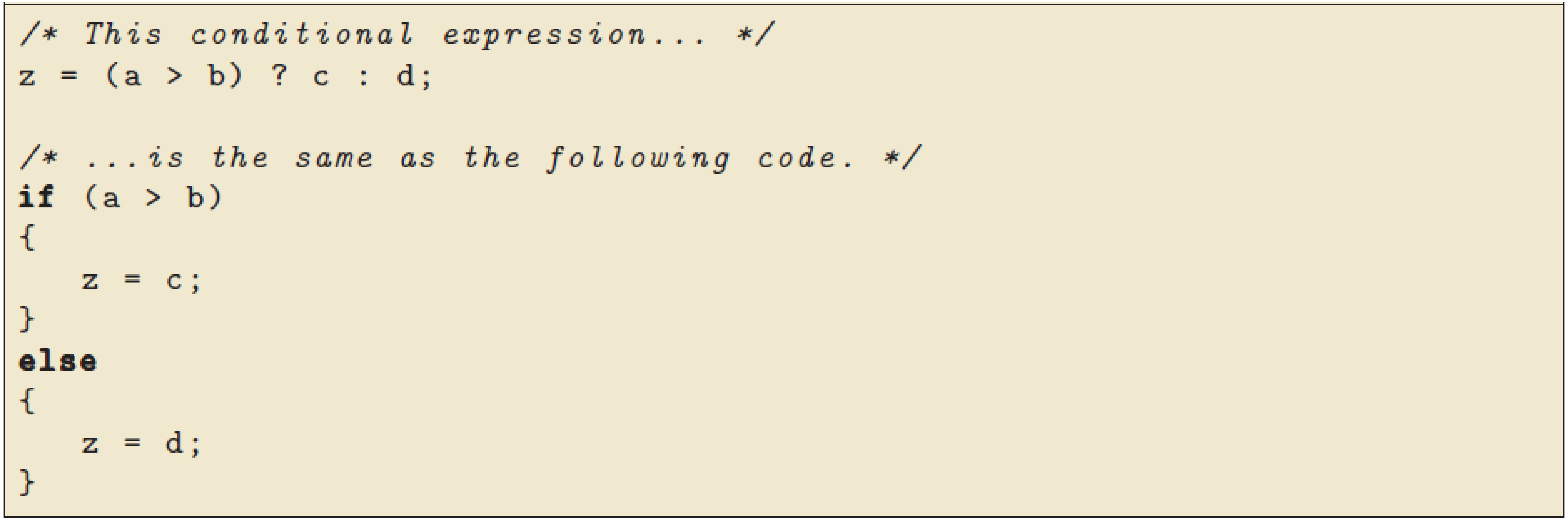
Operator	Operation
++	increment value by 1; either before or after the variable is used
--	decrement value by 1; either before or after the variable is used

Suppose x = 10 initially



Statement	n After	x After
`n = x++;`	10	11
`n = ++x;`	11	11
`n = x--;`	10	9
`n = --x;`	9	9

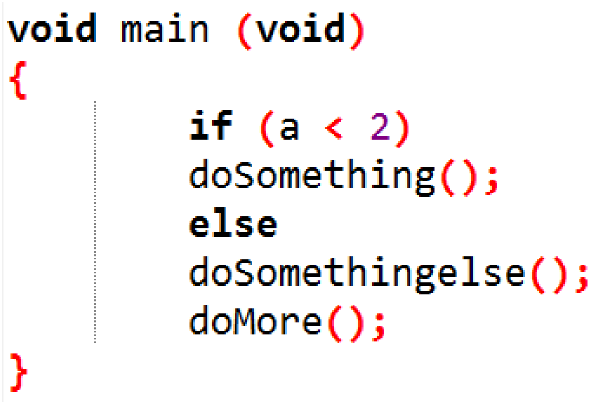
Conditional expression



```
/* This conditional expression... */
z = (a > b) ? c : d;

/* ...is the same as the following code. */
if (a > b)
{
   z = c;
}
else
{
   z = d;
}
```

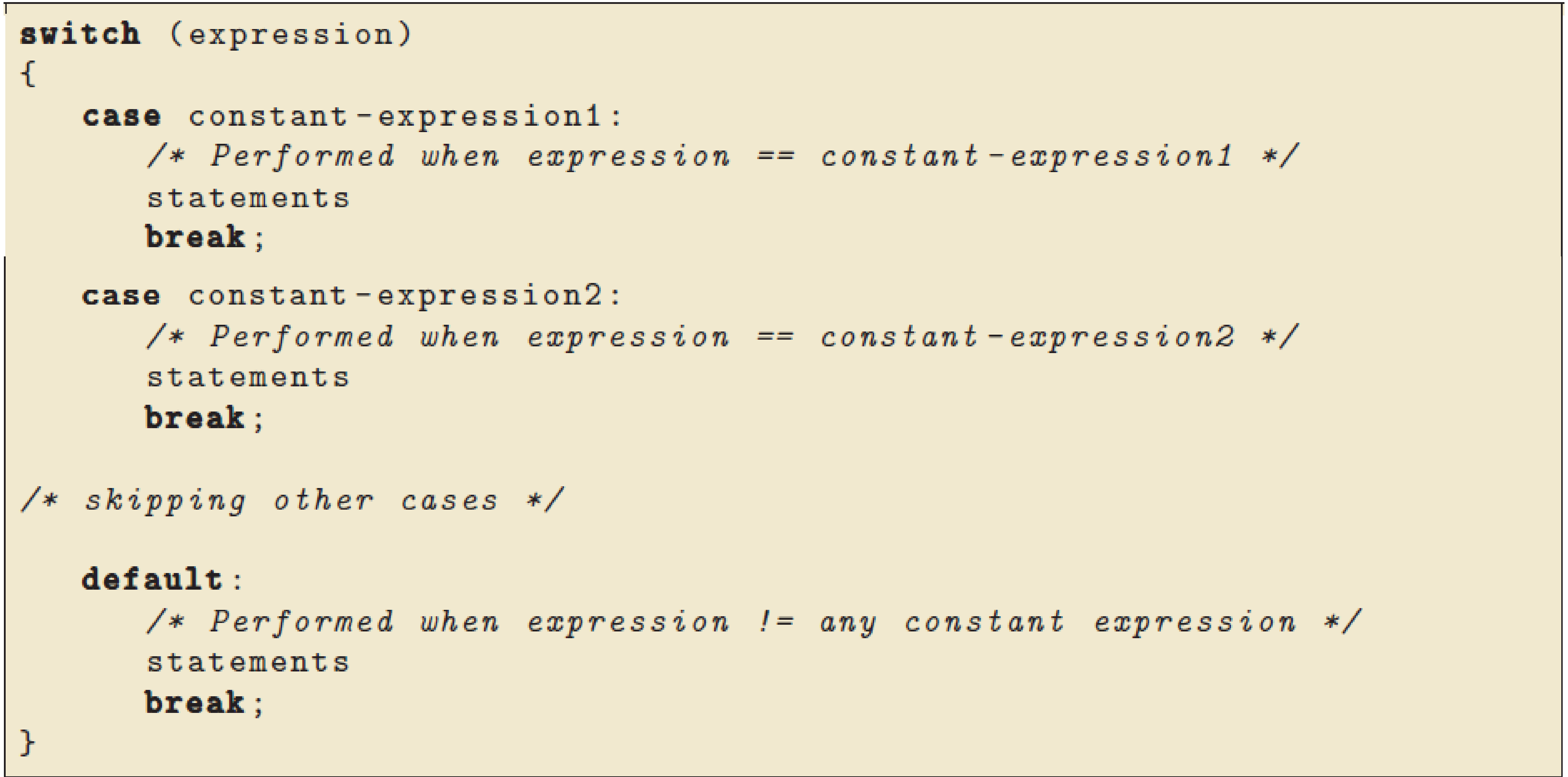
Note: it is a good practice to always put { } after if and else statements.

Avoid this kind of practice!



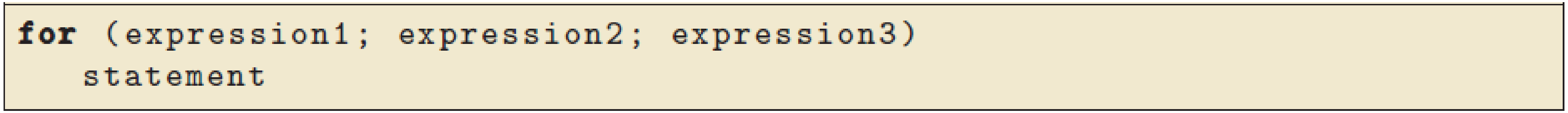
```
void main (void)
{
        if (a < 2)
        doSomething();
        else
        doSomethingelse();
        doMore();
}
```

Switch statement

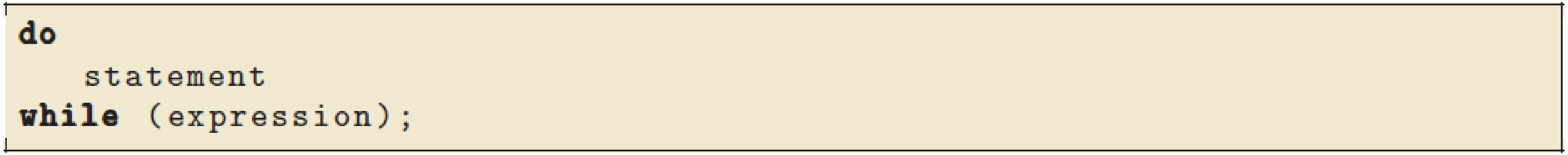


```
switch (expression)
{
   case constant-expression1:
      /* Performed when expression == constant-expression1 */
      statements
      break;

   case constant-expression2:
      /* Performed when expression == constant-expression2 */
      statements
      break;

/* skipping other cases */

   default:
      /* Performed when expression != any constant expression */
      statements
      break;
}
```

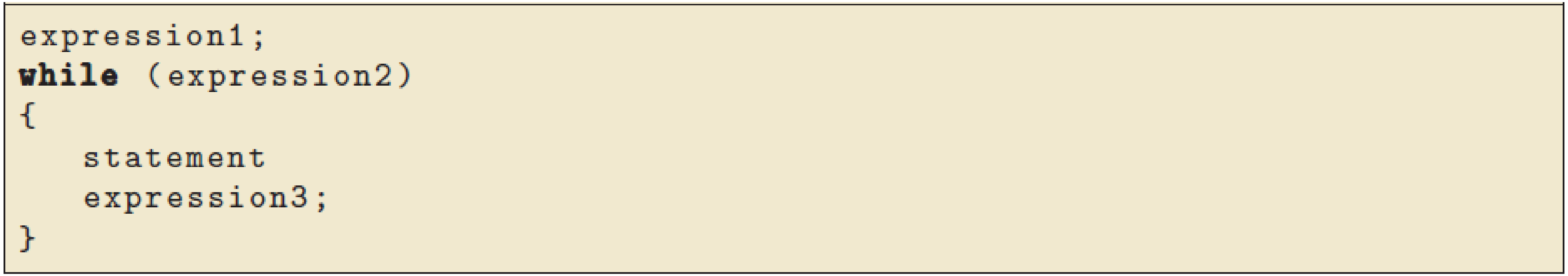
Loops



```
for (expression1; expression2; expression3)
   statement
```



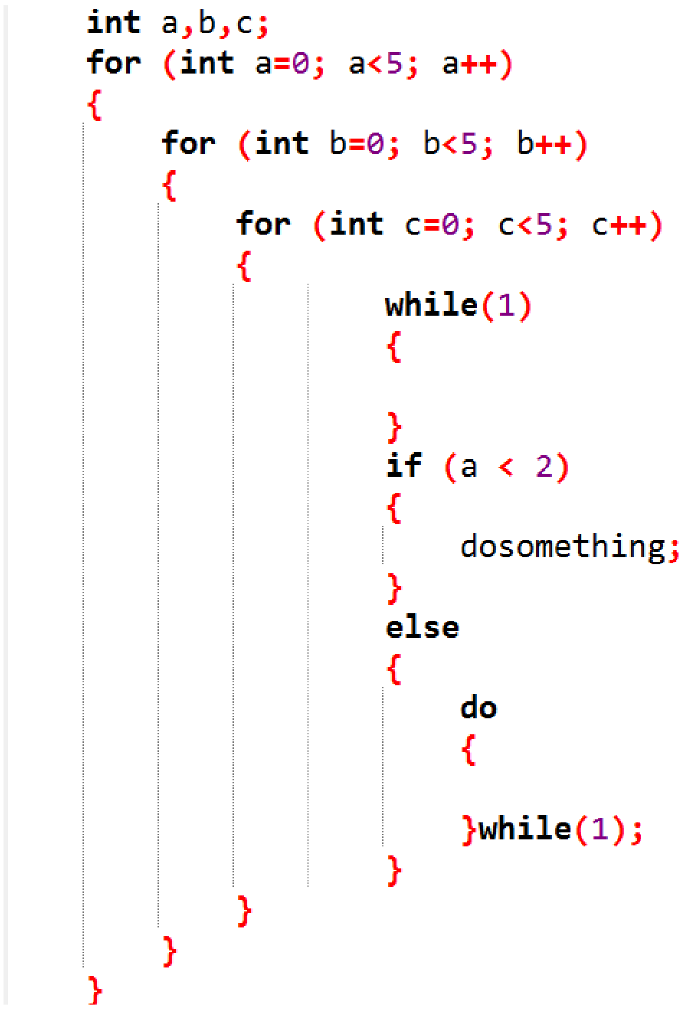
```
do
   statement
while (expression);
```



```
expression1;
while (expression2)
{
   statement
   expression3;
}
```

Good programming practice

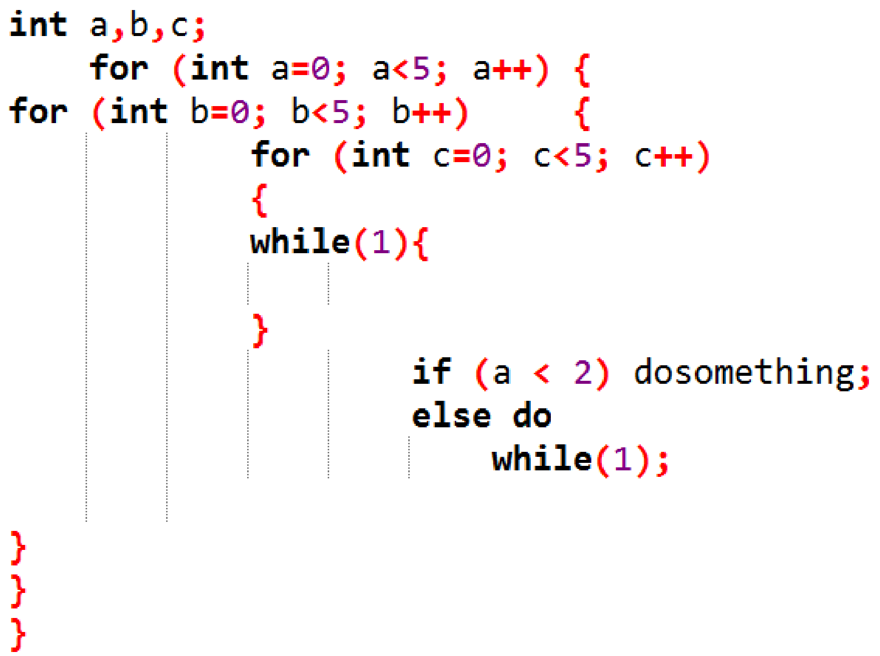
Good code



```
int a,b,c;
for (int a=0; a<5; a++)
{
    for (int b=0; b<5; b++)
    {
        for (int c=0; c<5; c++)
        {
                while(1)
                {

                }
                if (a < 2)
                {
                    dosomething;
                }
                else
                {
                    do
                    {

                    }while(1);
                }
        }
    }
}
```

- Always put {} where they belong.
- After { character, insert a new line and a tab character

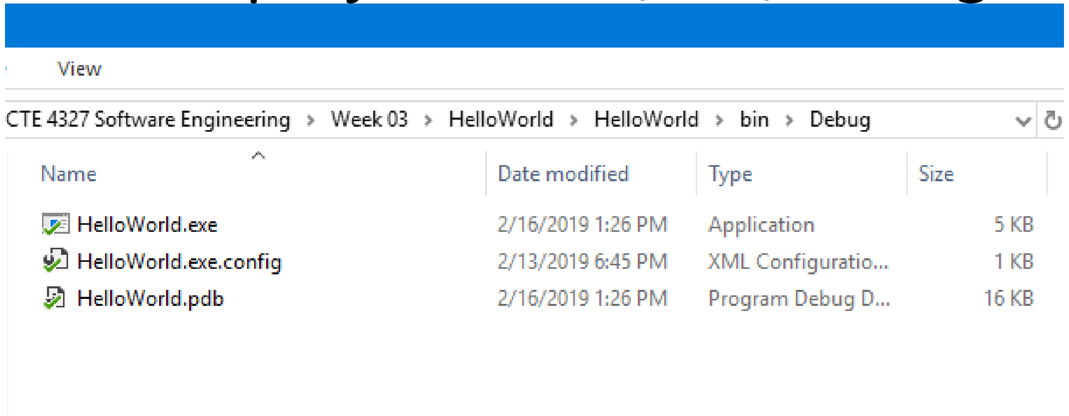
Bad code



```
int a,b,c;
    for (int a=0; a<5; a++) {
for (int b=0; b<5; b++)     {
            for (int c=0; c<5; c++)
            {
            while(1){

            }
                    if (a < 2) dosomething;
                    else do
                        while(1);
}
}
}
```

Compilation

- The exe file is inside projectfolder/bin/Debug

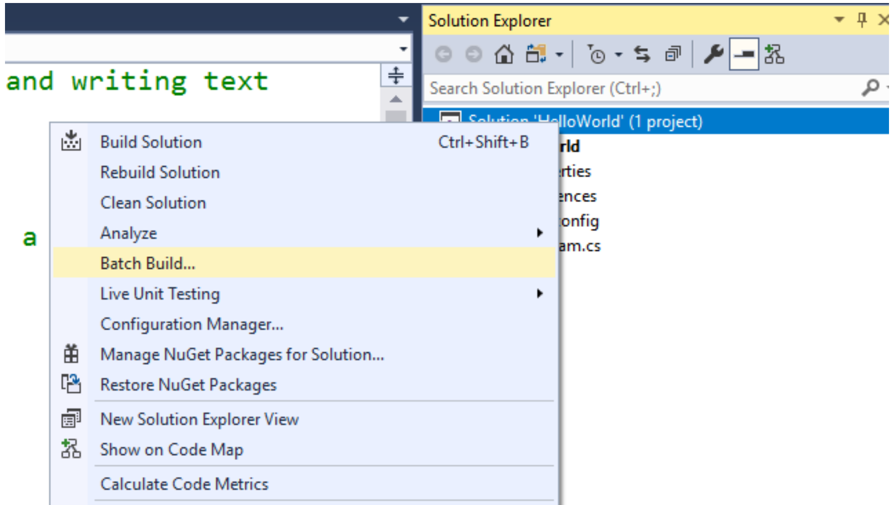


CTE 4327 Software Engineering › Week 03 › HelloWorld › HelloWorld › bin › Debug

Name	Date modified	Type	Size
HelloWorld.exe	2/16/2019 1:26 PM	Application	5 KB
HelloWorld.exe.config	2/13/2019 6:45 PM	XML Configuratio...	1 KB
HelloWorld.pdb	2/16/2019 1:26 PM	Program Debug D...	16 KB

- For final deployment, the Release version from projectfolder/bin/Release should be used.
- To compile the Release version, right-click

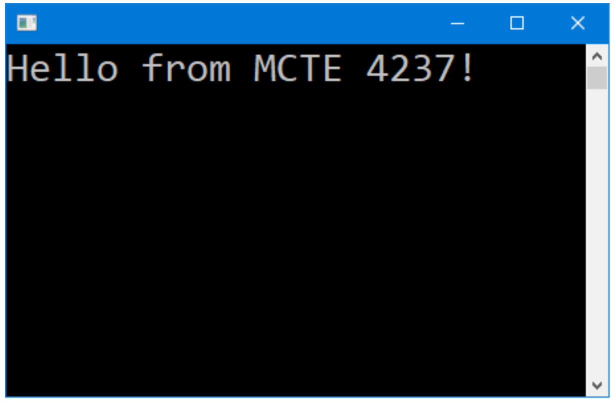
  on the solution, choose “batch build”



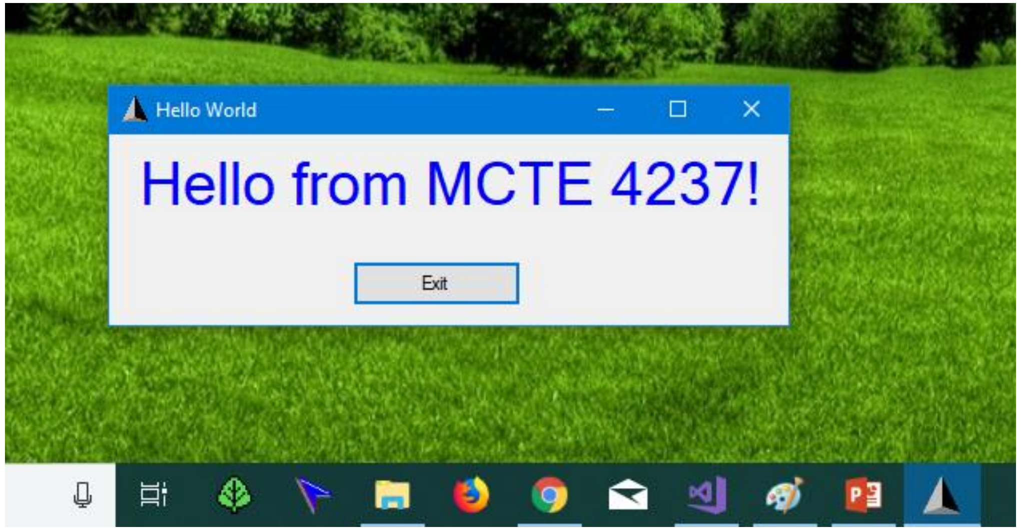
```
and writing text
```

Deploying on Windows

- Use can make an installer for your software or if it is just a single exe file without dependencies, you can just keep it portable.
- Windows has .NET framework built-in.
- If higher version of .NET framework is necessary, it will prompt to install.



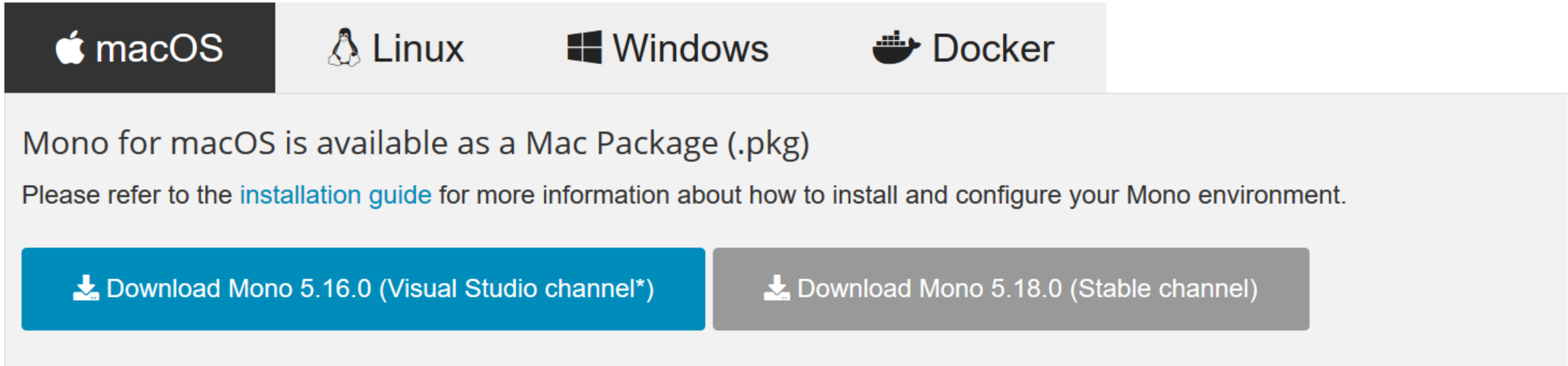
```
Hello from MCTE 4237!
```



Hello from MCTE 4237!

Deploying on Mac OS 10

- Install mono framework https://www.mono-project.com/download/stable/#download-mac



macOS
Linux
Windows
Docker

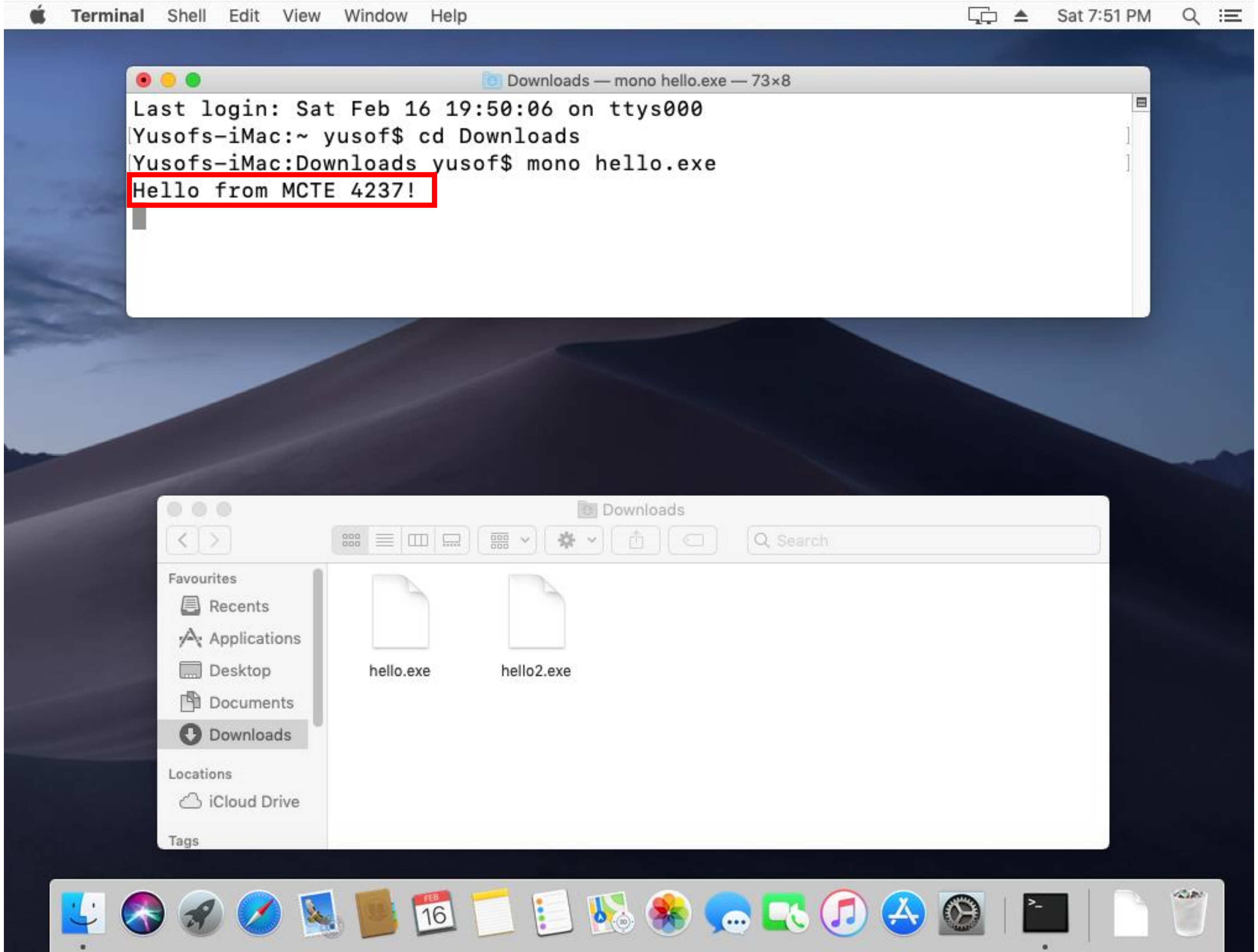
Mono for macOS is available as a Mac Package (.pkg)

Please refer to the installation guide for more information about how to install and configure your Mono environment.

Download Mono 5.16.0 (Visual Studio channel*)
Download Mono 5.18.0 (Stable channel)

Deploying on Mac OS 10

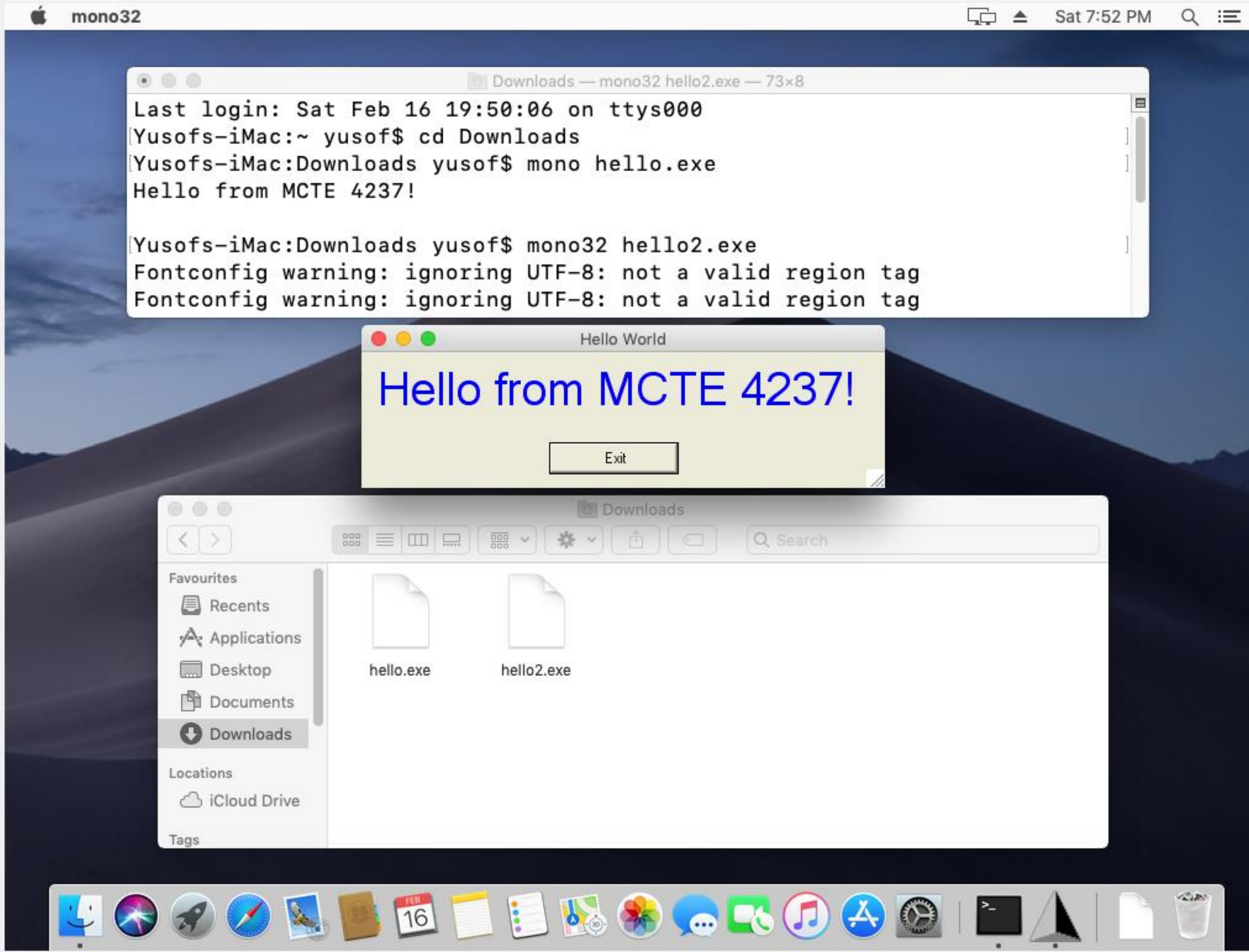
Command-line program



```
Last login: Sat Feb 16 19:50:06 on ttys000
Yusofs-iMac:~ yusof$ cd Downloads
Yusofs-iMac:Downloads yusof$ mono hello.exe
Hello from MCTE 4237!
```

Deploying on Mac OS 10

GUI-based program



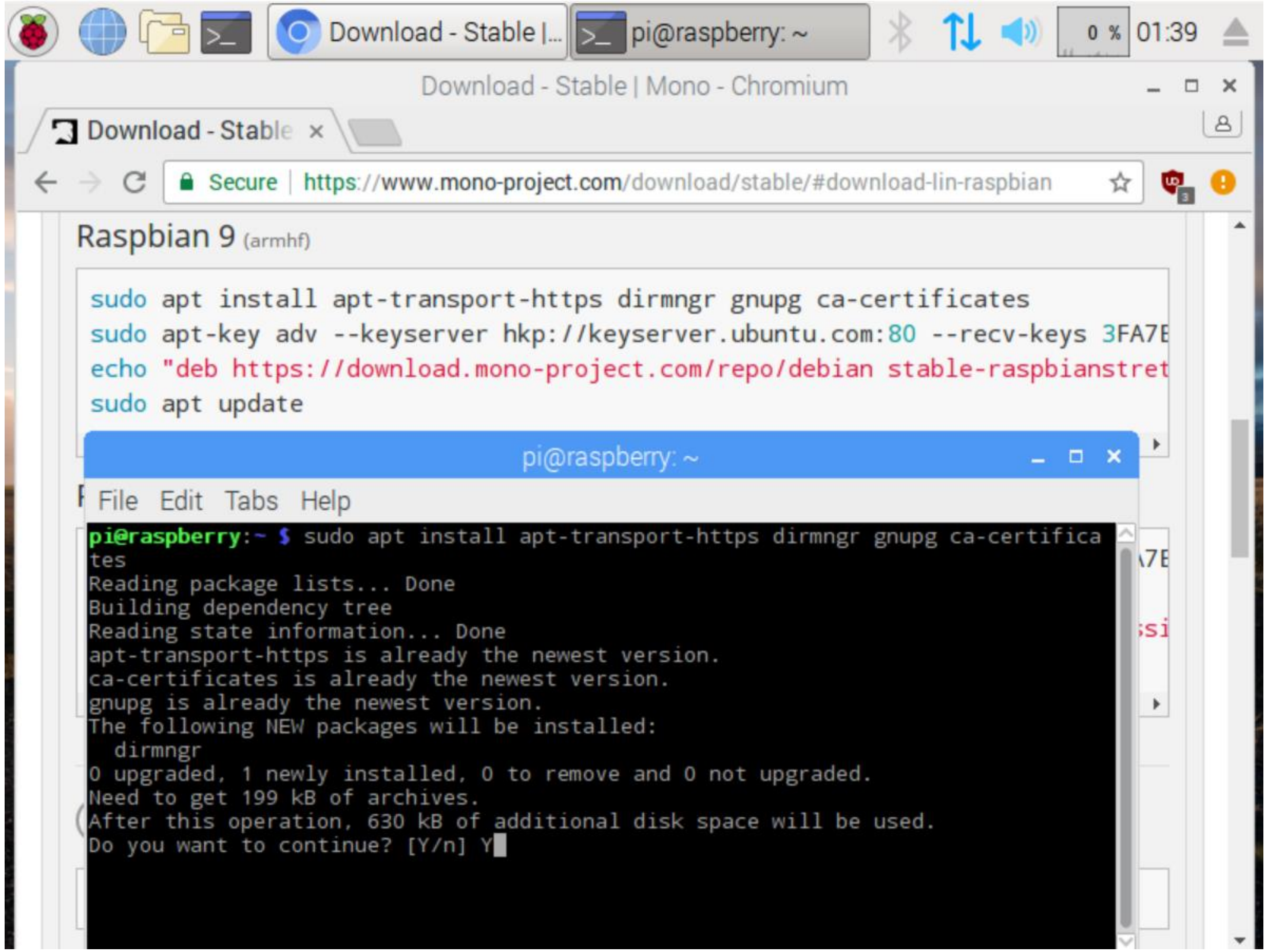
```
Last login: Sat Feb 16 19:50:06 on ttys000
Yusofs-iMac:~ yusof$ cd Downloads
Yusofs-iMac:Downloads yusof$ mono hello.exe
Hello from MCTE 4237!

Yusofs-iMac:Downloads yusof$ mono32 hello2.exe
Fontconfig warning: ignoring UTF-8: not a valid region tag
Fontconfig warning: ignoring UTF-8: not a valid region tag
```

Hello World

Hello from MCTE 4237!

Deploying on Raspberry Pi 3

- Install mono framework https://www.mono-project.com/download/stable/#download-lin-raspbian


Download - Stable |...
pi@raspberry: ~
0 %
01:39
Download - Stable | Mono - Chromium
Download - Stable
Secure
https://www.mono-project.com/download/stable/#download-lin-raspbian
pi@raspberry: ~
File
Edit
Tabs
Help

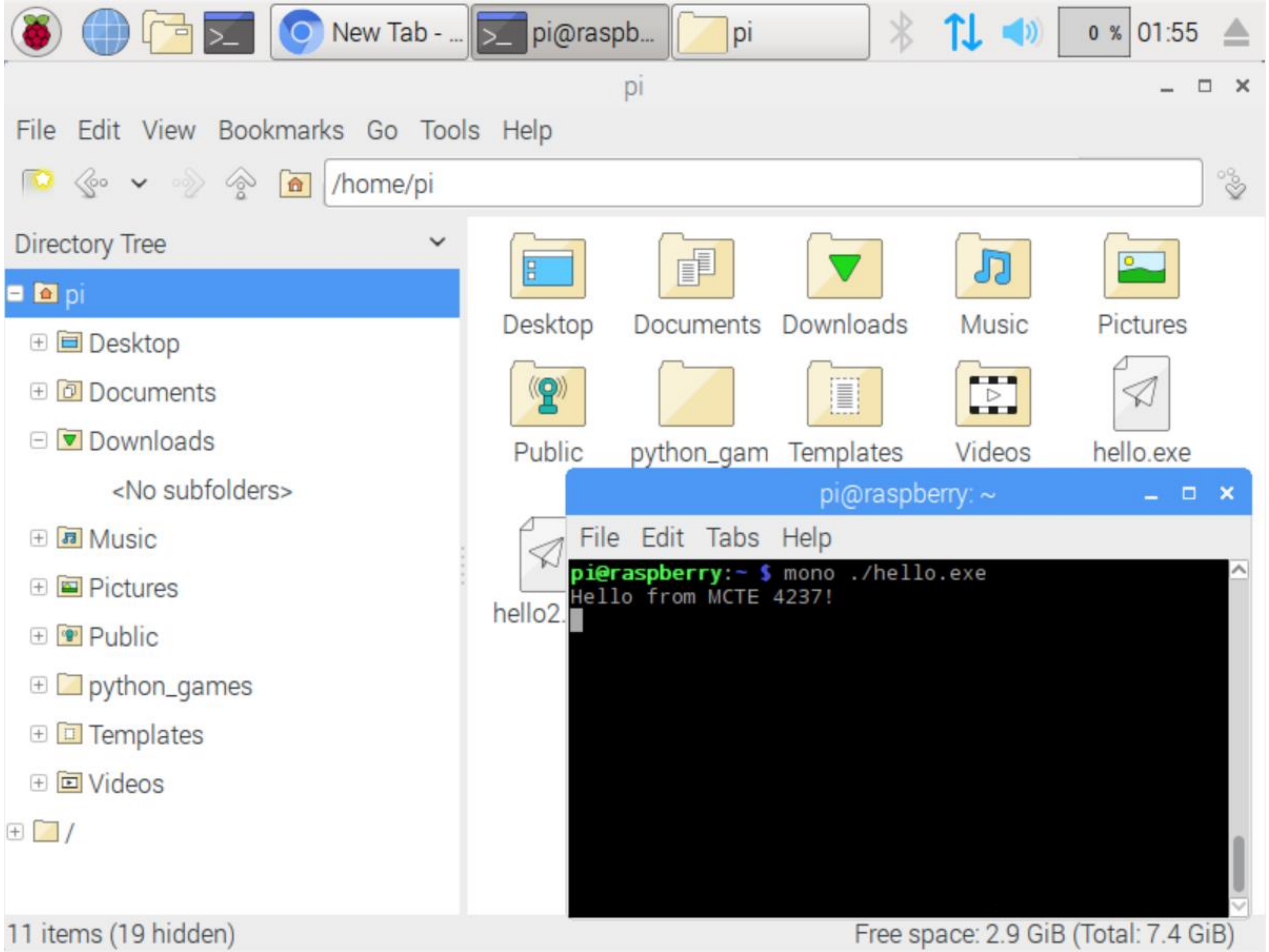
Raspbian 9 (armhf)

```
sudo apt install apt-transport-https dirmngr gnupg ca-certificates
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 3FA7E
echo "deb https://download.mono-project.com/repo/debian stable-raspbianstret
sudo apt update
```

```
pi@raspberry:~ $ sudo apt install apt-transport-https dirmngr gnupg ca-certifica
tes
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version.
ca-certificates is already the newest version.
gnupg is already the newest version.
The following NEW packages will be installed:
  dirmngr
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 199 kB of archives.
After this operation, 630 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Deploying on Raspberry Pi 3

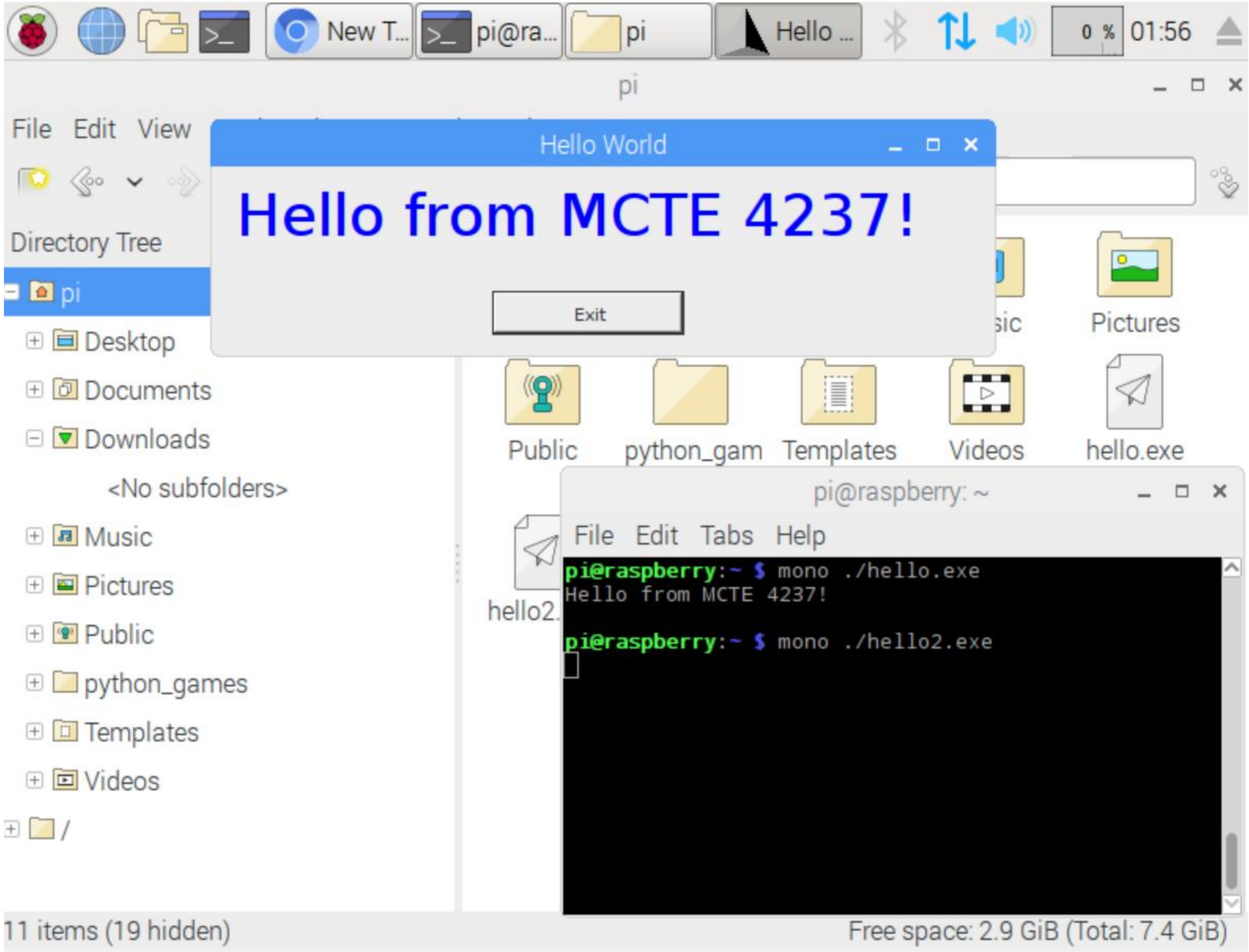
Command-line program



```
pi@raspberry:~ $ mono ./hello.exe
Hello from MCTE 4237!
```

Deploying on Raspberry Pi 3

GUI-based program

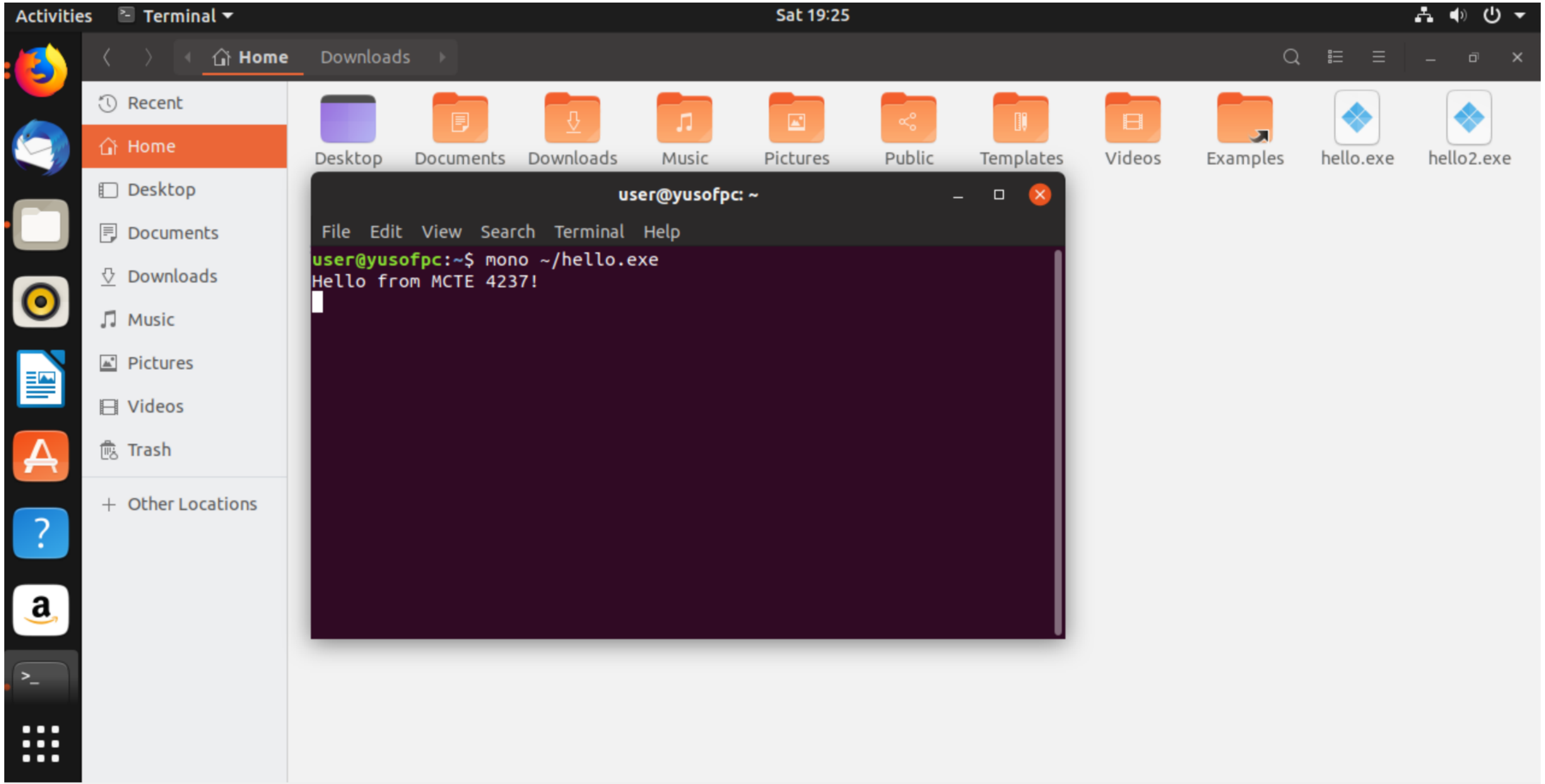


```
pi@raspberry:~ $ mono ./hello.exe
Hello from MCTE 4237!

pi@raspberry:~ $ mono ./hello2.exe
```

Deploying on Ubuntu

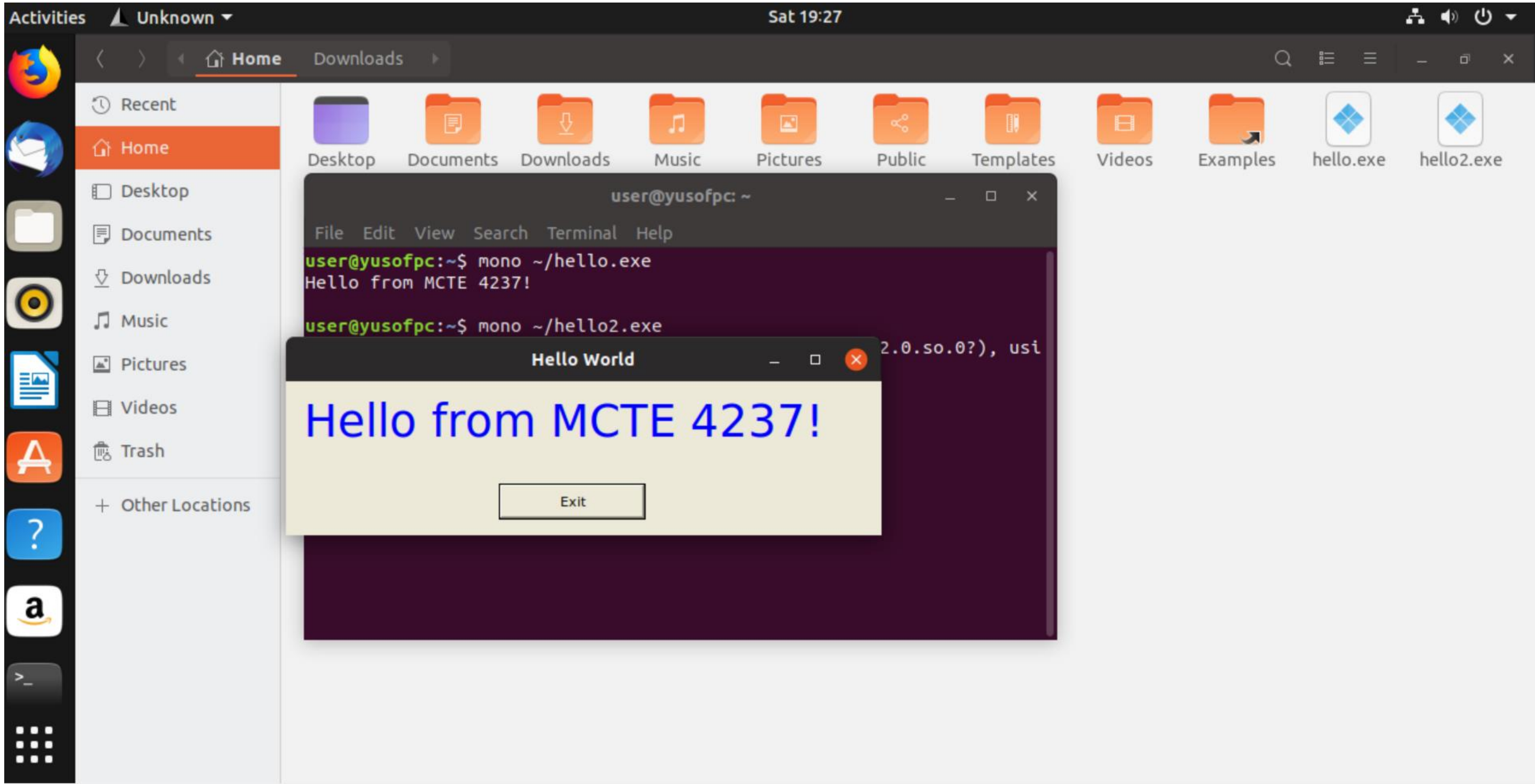
- Install mono framework https://www.mono-project.com/download/stable/#download-lin-ubuntu

Command-line program



```
user@yusofpc:~$ mono ~/hello.exe
Hello from MCTE 4237!
```

Command-line program

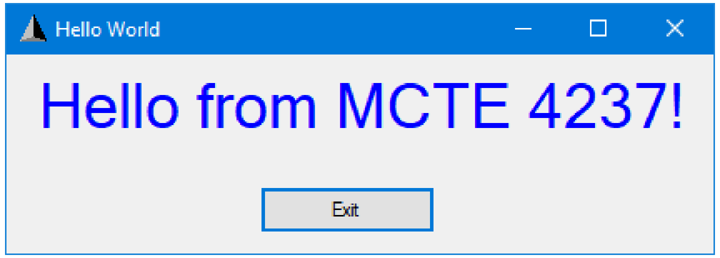


```
user@yusofpc:~$ mono ~/hello.exe
Hello from MCTE 4237!

user@yusofpc:~$ mono ~/hello2.exe
                                                          2.0.so.0?), usi
```

Hello from MCTE 4237!

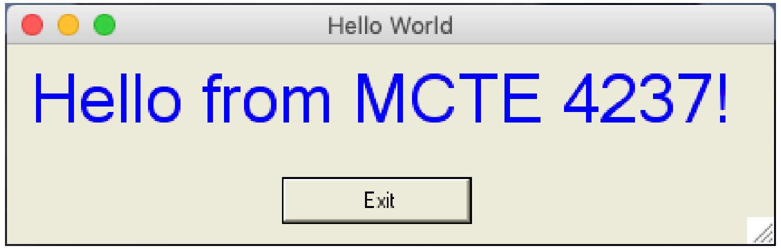
- The framework takes care of OS-specific chores. The developer just need to develop one type of coding.

Windows 10



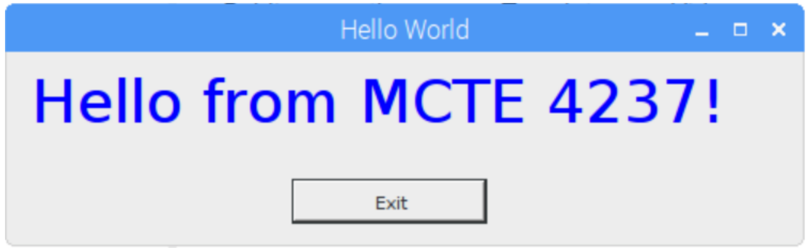
Hello from MCTE 4237!

Mac OS 10.12 Sierra



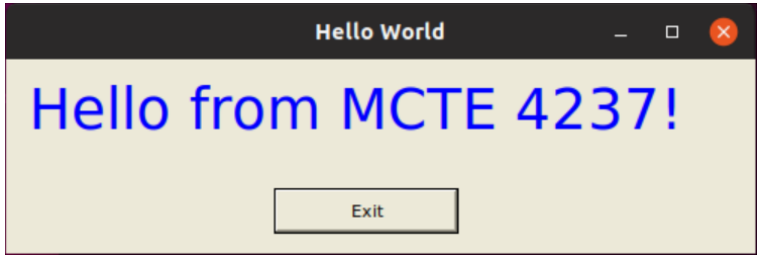
Hello from MCTE 4237!

Raspbian Jessie



Hello from MCTE 4237!

Ubuntu 18



Hello from MCTE 4237!

1-D Array

An array represents a fixed number of variables (called elements) of a particular type. The elements in an array are always stored in a contiguous block of memory, providing highly efficient access.

intarray
1
3
5

```
static void Main()
{
     var intarray = new int[5] { 1, 3, 5};

    Console.WriteLine(intarray[0]);
    Console.WriteLine(intarray[1]);
    Console.WriteLine(intarray[2]);
    Console.ReadLine();
}
```

Manual initialization

```
static void Main()
{
     int[] intarray = new int[5];

    intarray[0] = 1;
    intarray[1] = 3;
    intarray[2] = 5;


    Console.WriteLine(intarray[0]);
    Console.WriteLine(intarray[1]);
    Console.WriteLine(intarray[2]);
}
```

2-D array

```
static void Main()
{
    int[,] intarray = new int[2,2];

    intarray[0, 0] = 1;
    intarray[0, 1] = 5;
    intarray[1, 0] = 9;
    intarray[1, 1] = 3;

    Console.WriteLine(intarray[0,0]);
    Console.WriteLine(intarray[0,1]);
    Console.WriteLine(intarray[1,0]);
    Console.WriteLine(intarray[1,1]);
}
```

intarray	
1	5
9	3

Jagged array

Jagged arrays are arrays whose elements are also arrays.

```
static void Main()
{
    var myarray = new int[3][];

    myarray[0] = new int[4] { 1, 2, 3, 4 };
    myarray[1] = new int[2] { 5, 7 };
    myarray[2] = new int[3] { 7, 7, 7 };


    int retrieved = myarray[1][0]; //the value is 5
    int[] retrivedarray = myarray[1]; //the value is an array {5, 7}

}
```

for each loop

For each loop provides a convenient way of iterating elements of an array

```
int[] intarray = new int[5] { 1, 3, 5};
```

```
for (int i=0; i<intarray.Length; i++)
{
        Console.WriteLine(intarray[i]);
}
```

```
foreach (int element in intarray)
{
        Console.WriteLine(element);
}
```

```
  Output:

1
3
5
```

Value Types Versus Reference Types

- Value types (most primitive datatypes, struct, enum)
- Reference types (classes, arrays, delegate, string)
- Generic type parameters
- Pointer types

Value types

What is the output of the following program?

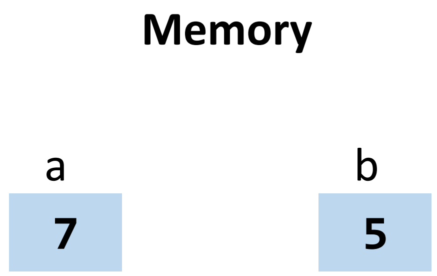
```
static void Main()
{
    int a = 5;
    int b = a;

    a = 7;

    Console.WriteLine(a);
    Console.WriteLine(b);

    Console.ReadLine();
}
```

int is a value type. a and b have independent memories.

The statement b = a simply copies the value of a to b.

a and b are separate entities.

```
Output:

7
5
```



Memory

a
7

b
5

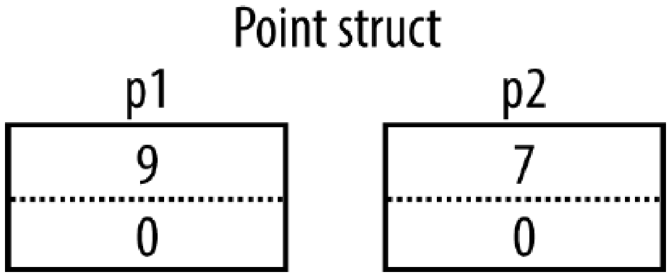
Value types

What is the output of the following program?

```
public struct Point
{
    public int X;
    public int Y;
}

static void Main()
{
    Point p1;
    p1.X = 7;
    Point p2 = p1;
    Console.WriteLine(p1.X);
    Console.WriteLine(p2.X);
    p1.X = 9;
    Console.WriteLine(p1.X);
    Console.WriteLine(p2.X);
    Console.ReadLine();
}
```

struct is also a value type

```
  Output:


7
7
9
7
```



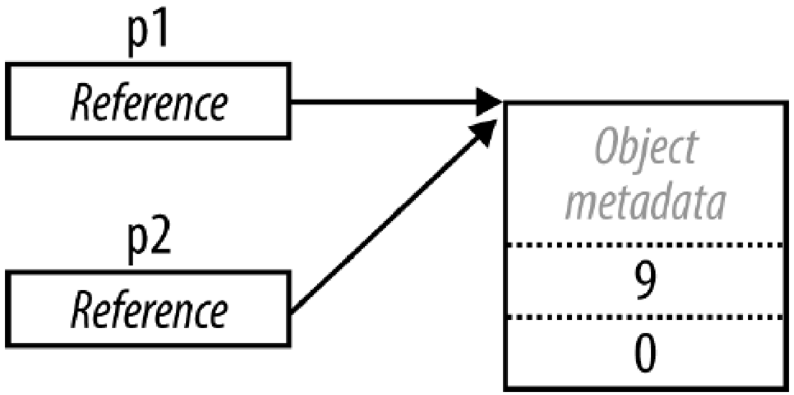
Reference types

What is the output of the following program?

```
public class Point
{
    public int X;
    public int Y;
}

static void Main()
{
    Point p1 = new Point();
    p1.X = 7;
    Point p2 = p1;

    Console.WriteLine(p1.X);
    Console.WriteLine(p2.X);
    p1.X = 9;
    Console.WriteLine(p1.X);
    Console.WriteLine(p2.X);
    Console.ReadLine();
}
```

- class is a reference type.
- p1 is actually a pointer that is pointing to some memory location holds data.
- The statement p2 = p1 makes p2 also points to the same memory location.
- Any changes made to p1 also affects p2 and vice versa.

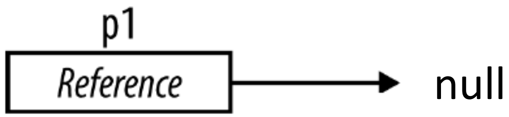
```
Output:

7
7
9
8
```



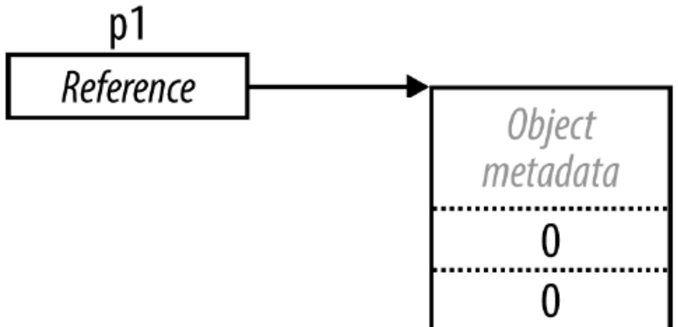
new keyword

```
public class Point
{
    public int X;
    public int Y;
}
```

```
Point p1; //this statement only declares a reference (to an empty memory location).
                // It is also called ‘null reference’
```

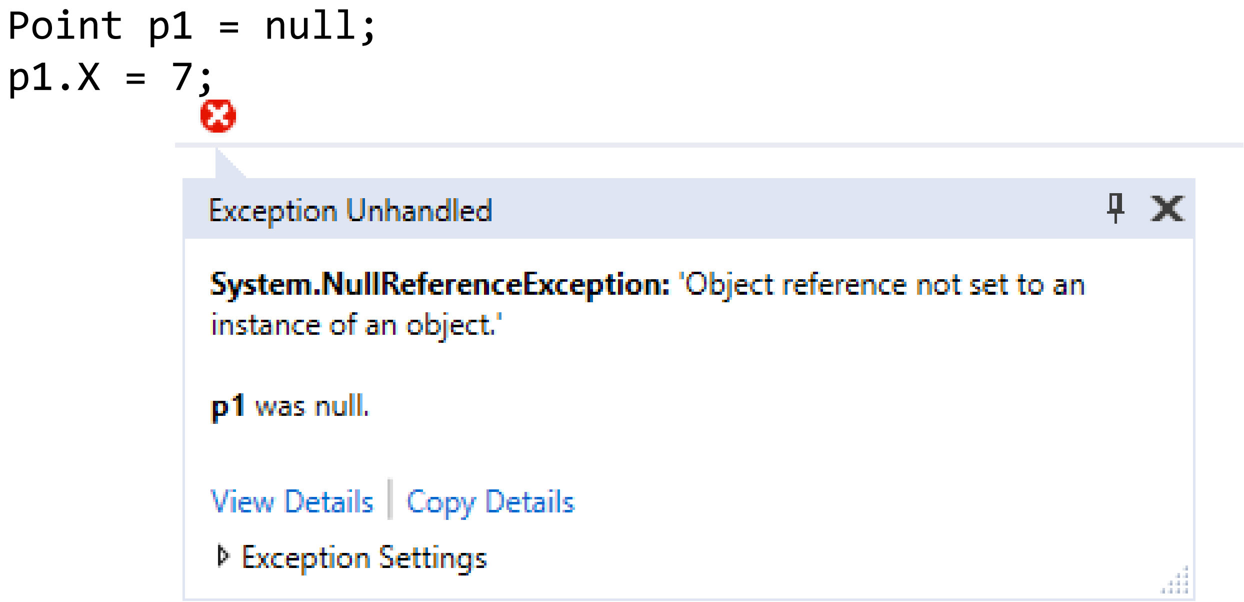


```
p1 = new Point(); //in order to make it usable, we need to use the keyword ‘new’.
                                                 //Only then it will create the memory location
```



null reference

- Using an object that is pointing to empty memory location (null reference) will raise a run-time error



```
Point p1 = null;
p1.X = 7;
```

Exception Unhandled

System.NullReferenceException: 'Object reference not set to an instance of an object.'

p1 was null.

Arrays = Value types or Ref types?

Arrays are reference types. That is why the “new” keyword is necessary.

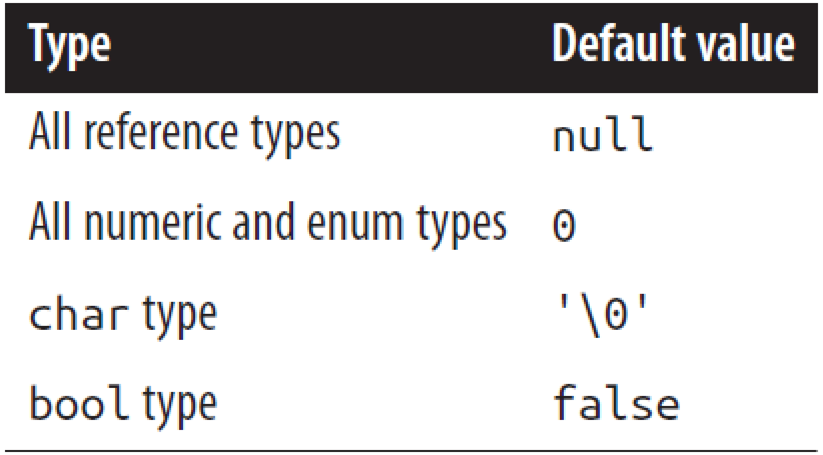
Here b and a are pointing to the same memory location.

```
static void Main()
{
    int[] a = new int[] { 1, 2, 3, 4 };

    int[] b = a;

    foreach (int element in b)
    {
        Console.WriteLine(element);
    }

    Console.ReadLine();

}
```

```
  Output:

1
2
3
4
```

Default values of variables

All type instances have a default value (if they are uninitialized)



Type	Default value
All reference types	`null`
All numeric and enum types	`0`
`char` type	`'\0'`
`bool` type	`false`

var data type

var is used to declare **implicitly** typed local variable means it tells the compiler to figure out the type of the variable at compilation time. A var variable must be initialized at the time of declaration.

```
var intarray = new int[5] { 1, 3, 5, 9, 5 };

 var str = "1";
 var num = 0;

 var P1 = new Point();
```

Same as this

```
int[5] intarray = new int[5] { 1, 3, 5, 9, 5 };

string str = "1";
int num = 0;

Point P1 = new Point();
```

Anonymous type

Anonymous type, as the name suggests, is a type that doesn't have any name. C# allows you to create an object with the *new* keyword without defining its class. Var is used to hold the reference of anonymous types.

Here student is of anonymous type. But has name, CGPA and age information.

```
var student = new { Name = "Ahmad", CGPA = 3.14, Age=22};
```

```
Console.WriteLine(student.Name);
Console.WriteLine(student.CGPA);
Console.WriteLine(student.Age);
```

Output:

```
Ahmad
3.14
22
```

Functions

- A function/method performs some meaningful task.
- A function should have a set of parameters and a return type.
- A function can call other functions including itself!
- A **static** function can only call other functions that are **static** (more about static later).

```
static void Main()
{
    Display("Ahmad", "Malaysia");
    Console.ReadLine();
}

static void Display(string name, string country, int age = 22)
{
    Console.WriteLine(name);
    Console.WriteLine(country);
    Console.WriteLine(age);
}
```

Parameters

Optional parameter with default value

Output:

```
Ahmad
Malaysia
22
```

Functions with multiple return values

out statement

```
static void Main()
{
    string Name, Country;
    int Age;

    LookupName(111111, out Name, out Country, out Age);
    Console.WriteLine(Name);
    Console.WriteLine(Country);
    Console.Write(Age);
    Console.Read();

}
static void LookupName(int matric, out string Name, out string Country, out int Age)
{
    string FoundName = "Ahmad";       //Dummy data
    string FoundCountry = "Malaysia";  //Dummy data
    int FoundAge = 22;            //Dummy data

    Name = FoundName;
    Country = FoundCountry;
    Age = FoundAge;
}
```

The out statement reverses the flow of information. It forces the function to output information through the parameters.

Output:

```
Ahmad
Malaysia
22
```

Functions with multiple return values

Using new datatype (struct)

```
struct Student
{
    public string Name;
    public string Country;
    public int Age;
}
static void Main()
{
    Student student = LookupName(111111);
    Console.WriteLine(student.Name);
    Console.WriteLine(student.Country);
    Console.Write(student.Age);
    Console.Read();

}
static Student LookupName(int matric)
{
    Student found;
    found.Name = "Ahmad";       //Dummy data
    found.Country = "Malaysia";  //Dummy data
    found.Age = 22;            //Dummy data
    return found;
}
```

Output:

```
Ahmad
Malaysia
22
```

Functions with multiple return values

Most elegant way (only with .NET Framework 4.7 and above)

```
static void Main()
{
    (string Name, string Country, int Age) = LookupName(111111);
    Console.WriteLine(Name);
    Console.WriteLine(Country);
    Console.Write(Age);
    Console.Read();

}

static (string, string, int) LookupName(int matric)
{
    string name = "Ahmad";       //Dummy data
    string country = "Malaysia";  //Dummy data
    int age = 22;            //Dummy data

    return (name, country, age);
}
```

Output:

```
Ahmad
Malaysia
22
```

Passing parameters by values

- By default, arguments with value types in C# are passed to functions by value

- This means a copy of the value is created when passed to the method.

- Assigning p a new value does not change the contents of x, since p and x reside in different memory locations.

```
static void Foo(int p)
{
    p = p + 1; // Increment p by 1
    Console.WriteLine(p); // Write p to screen
}
static void Main()
{
    int q = 8;
    Foo(q); // Make a copy of q
    Console.WriteLine(q); // q will still be 8
    Console.ReadLine();

}
```

```
  Output:

9
8
```

Passing parameters by reference

```
public class Point
{
    public int X;
    public int Y;
}

static void Foo(Point p)
{
    p.X = p.X + 1;
    Console.WriteLine(p.X);
}
static void Main()
{
    Point q = new Point();
    q.X = 8;
    Foo(q);
    Console.WriteLine(q.X);

    Console.ReadLine();
}
```

- Since q is now a reference type, it is passed to the function by reference.
- p and q have the same reference (point to the same memory location)
- Changes made in p reflect in q (and vice versa)

```
 Output:

9
9
```

Forcing to pass by reference

```
static void Foo(ref int p)
{
    p = p + 1; // Increment p by 1
    Console.WriteLine(p); // Write p to screen
}
static void Main()
{
    int q = 8;
    Foo(ref q);
    Console.WriteLine(q);
    Console.ReadLine();

}
```

We can force value types to be passed to functions by reference by using the keyword “ref”.

```
  Output:

9
9
```

String

```
int z = 538;
string s = z.ToString(); //Now s will be "538" in text

string a = "Hello ";
string b = "World ";
string c = "from MCT 4327";

string d = a + b + c;  //d will be "Hello World from MCT 4327";

int length = a.Length; //a will be 6 (including space);
string sub = a.Substring(2); //sub will be "llo ";
string sub2 = a.Substring(2, 2); //sub2 will be "ll";

string[] splitted = d.Split(' '); //Split d by space

foreach (string str in splitted)
{
    Console.WriteLine(str);
}
Console.ReadLine();
```

- Most data types have a function ToString() that converts them to strings.
- Please read https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/strings/

Output:

```
Hello
World
from MCT 4327
```

DateTime

```
static void Main()
{
    DateTime a = DateTime.Now;

    Console.WriteLine(a.ToString());
    Console.WriteLine(a.DayOfWeek);
    Console.WriteLine(a.Month);
    Console.WriteLine(a.Year);

    a = a.AddDays(1);
    Console.WriteLine(a.DayOfWeek);

    a = new DateTime(2012, 1, 27);
    Console.WriteLine(a.ToString());

    a = new DateTime(2012, 1, 27, 13, 5, 18);
    Console.WriteLine(a.ToString());
    Console.WriteLine(a.ToString("dd/MM/yyyy hh:mm tt"));

    Console.ReadLine();
}
```

- DateTime is a value type.
- Please go through https://www.dotnetperls.com/datetime

Output:

```
2/20/2019 7:46:32 PM
Wednesday
2
2019
Thursday
1/27/2012 12:00:00 AM
1/27/2012 1:05:18 PM
27/1/2012 01:05 PM
```

Note: it follows your PC’s date time format. Most PCs use American format, month/day/year instead of day/month/year.

Format	Description	Example
"y"	The year, from 0 to 99 without leading zero	5 19
"yy"	The year, from 00 to 99	05 19
"yyyy"	Year in full 4 digits	2019
"M"	The month, from 1 through 12 without leading zero	9
"MM"	The month, from 01 through 12	09
"MMM"	The abbreviated name of the month.	Sep
"MMMM"	The full name of the month.	September
"d"	The day of the month, from 1 through 31.	1
"dd"	The day of the month, from 01 through 31.	01
"ddd"	The abbreviated name of the day of the week.	Mon
"dddd"	The full name of the day of the week.	Monday
"h"	The hour, using a 12-hour clock from 1 to 12 without leading zero.	9
"hh"	The hour, using a 12-hour clock from 01 to 12.	09
"H"	The hour, using a 24-hour clock from 0 to 23 without leading zero	1 13
"HH"	The hour, using a 24-hour clock from 00 to 23.	01 13
"m"	The minute, from 0 through 59 without leading zero	9
"mm"	The minute, from 00 through 59.	09
"s"	The second, from 0 through 59 without leading zero	8
"ss"	The second, from 00 through 59.	08
"fff"	The milliseconds in a date and time value.	617
"tt"	AM or PM	AM

DateTime

Example:

```
Console.WriteLine(DateTime.Now.ToString("dd-MMMM-yyyy, ddddd, hh:mm:ss:ffff tt"));
```

```
                     Output:

20-February-2019, Wednesday, 08:10:13:0156 PM
```

Parsing

A parsing function converts string to a specific data type.

Examples:

int.Parse() converts string to int
DateTime.Parse() converts string to DateTime
float.Parse() convers string to flaot

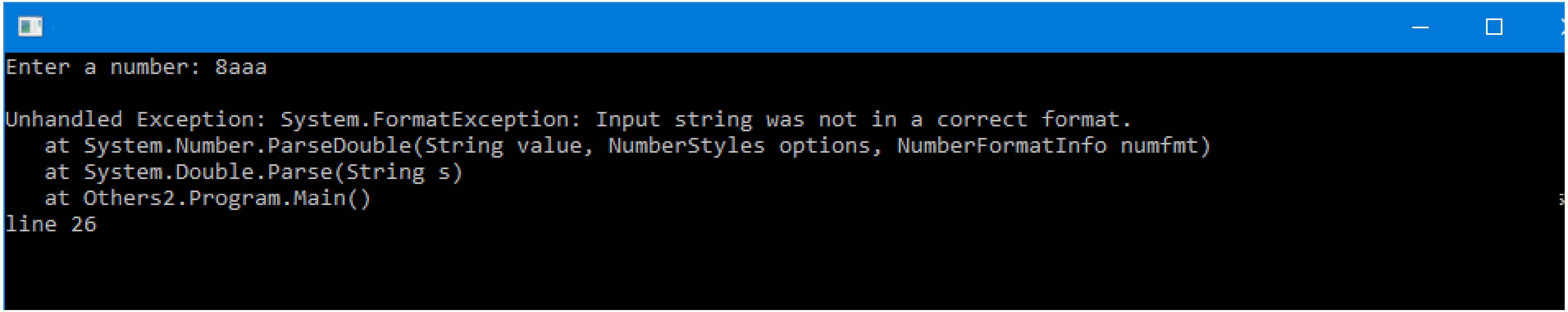
```
static void Main()
{
    Console.Write("Enter a number: ");
    string input = Console.ReadLine();
    int number = int.Parse(input); //Converts string to int
    int output = 1200 / number;
    Console.WriteLine("Output = " + output.ToString());

    Console.ReadLine();
}
```

Output:

```
Enter a number: 12
Output = 100
```

Exception handling

An exception is a runtime error that occurs during the execution of a program. For example, in the previous program, when the user enters non-numeric input, the program will experience an exception. If it is not handled, the program will crash. Every exception needs to be gracefully handled.



```
Enter a number: 8aaa

Unhandled Exception: System.FormatException: Input string was not in a correct format.
   at System.Number.ParseDouble(String value, NumberStyles options, NumberFormatInfo numfmt)
   at System.Double.Parse(String s)
   at Others2.Program.Main()
line 26
```

Exception handling

A try-catch block without arguments handles all the exception

```
static void Main()
{
    Console.Write("Enter a number: ");
    string input = Console.ReadLine();
    try
    {
        int number = int.Parse(input); //Converts string to double
        int output = 1200 / number;
        Console.WriteLine("Output = " + output.ToString());
    }
    catch
    {
        Console.WriteLine("Invalid input");
    }


    Console.ReadLine();
}
```

Output:

```
Enter a number: 8aaa
Invalid input
```

Because of division by 0 →

Output:

```
Enter a number: 0
Invalid input
```

Exception handling

Try-catch block can be parameterized to catch specific errors

```
static void Main()
{
    Console.Write("Enter a number: ");
    string input = Console.ReadLine();
    try
    {
        int number = int.Parse(input); //Converts string to double
        int output = 1200 / number;
        Console.WriteLine("Output = " + output.ToString());
    }
    catch (FormatException ex)
    {
        Console.WriteLine("The input must be a number");
    }
    catch (DivideByZeroException ex)
    {
        Console.WriteLine("Input cannot be 0");
    }
    catch (Exception ex)
    {
        Console.WriteLine("Sorry. Unexpected error: " + ex.Message);
    }
    Console.ReadLine();
}
```

```
Output:

Enter a number: 8aaa
Invalid input
```

```
Output:

Enter a number: 0
Invalid input
```

Because of division by 0

Random

```
static void Main()
{
    Random rnd = new Random();

    Console.WriteLine(rnd.Next(10)); //Generates a random number between 0 & 9
    Console.WriteLine(rnd.Next(10)); //Generates a random number between 0 & 9
    Console.WriteLine(rnd.Next(10)); //Generates a random number between 0 & 9


    Console.WriteLine(rnd.Next(50, 100)); //Generates a random number between 50 & 99
    Console.WriteLine(rnd.Next(50, 100)); //Generates a random number between 50 & 99
    Console.WriteLine(rnd.Next(50, 100)); //Generates a random number between 50 & 99

    Console.WriteLine(rnd.NextDouble()); //Generates a random between 0 and 1
    Console.WriteLine(rnd.NextDouble()); //Generates a random between 0 and 1
    Console.WriteLine(rnd.NextDouble()); //Generates a random between 0 and 1

    Console.ReadLine();
}
```

Output:

```
6
0
6
57
84
51
0.199591806717027
0.572778981445721
```

Reading small text files

Can declare
`using System.IO;`

```
static void Main()
{
    // Example #1
    // Read the file as one string.
    string text = System.IO.File.ReadAllText(@"C:\WriteText.txt");
    // Display the file contents to the console. Variable text is a string.
    System.Console.WriteLine("Contents of WriteText.txt = " + text);

    // Example #2
    // Read each line of the file into a string array. Each element
    // of the array is one line of the file.
    string[] lines = System.IO.File.ReadAllLines(@"C:\WriteLines2.txt");

    // Display the file contents by using a foreach loop.
    System.Console.WriteLine("Contents of WriteLines2.txt = ");
    foreach (string line in lines)
    {
        // Use a tab to indent each line of the file.
        Console.WriteLine(line);
    }
    // Keep the console window open in debug mode.
    Console.WriteLine("Press any key to exit.");
    System.Console.ReadKey();
}
```

Writing small text file

```
static void Main()
{
    // Example #1: Write an array of strings to a file.
    // Create a string array that consists of three lines.
    string[] lines = { "First line", "Second line", "Third line" };
    // WriteAllLines creates a file, writes a collection of strings to the file,
    // and then closes the file.  You do NOT need to call Flush() or Close().
    System.IO.File.WriteAllLines(@"C:\WriteLines.txt", lines);

    // Example #2: Write one string to a text file.
    string text = "A class is the most powerful data type in C#. Like a structure, " +
                   "a class defines the data and behavior of the data type. ";
    // WriteAllText creates a file, writes the specified string to the file,
    // and then closes the file.    You do NOT need to call Flush() or Close().
    System.IO.File.WriteAllText(@"C:\WriteText.txt", text);

    // Example #3: Append new text to an existing file.
    // The using statement automatically flushes AND CLOSES the stream and calls
    // IDisposable.Dispose on the stream object.
    using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"C:\WriteLines2.txt", true))
    {
        file.WriteLine("Fourth line");
    }
}
```

Other file operations

```
byte[] bytes = System.IO.File.ReadAllBytes(@"C:\test.exe"); //read binary file into byte array

System.IO.File.WriteAllBytes(@"C:\test.exe", bytes); //write byte array to binary file

System.IO.File.Delete(@"C:\test.exe");  //Delete

System.IO.Directory.CreateDirectory(@"C:\Test"); //Make new folder

System.IO.File.Move(@"C:\test.exe", @"C:\Test\test.exe");  //Move (can also use for renaming)

string[] filenames = System.IO.Directory.GetFiles(@"C:\");  //Get filenames from directory
```