UPskill – C#

Introdução (C# procedimental)

C#

- O C# é uma linguagem desenvolvida pela Microsoft e que é amplamente utilizada (principalmente a nível empresarial);
- Não confundir com a linguagem C!
- Tal como a linguagem Java, a linguagem C# abstrai o utilizador de várias especificidades relacionadas com a comunicação com o hardware, tornando-a mais acessível.

Compilador

- A linguagem C# é uma linguagem compilada;
- Podemos escrever um programa utilizando somente um editor de texto!
- No entanto, o ficheiro criado irá fazer parte de um processo de compilação do programa;

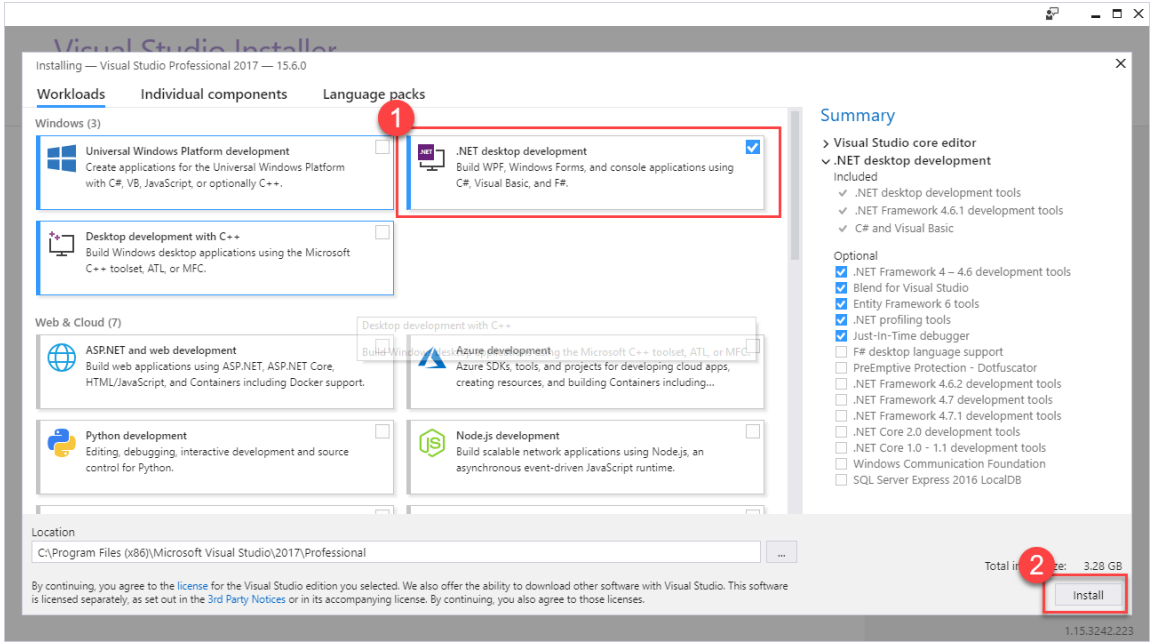
Criar programas em C#

- A Microsoft disponibiliza uma ferramenta chamada de Visual Studio que incorpora o compilador, editor e um debugger;
- Existe uma versão gratuita disponível:
  - https://visualstudio.microsoft.com/pt-br/vs/express/?rr=https%3A%2F%2Fwww.google.com%2F
- É necessária a instalação da Framework Microsoft .NET (incluída no instalador do Visual Studio).

Criar programas em C#

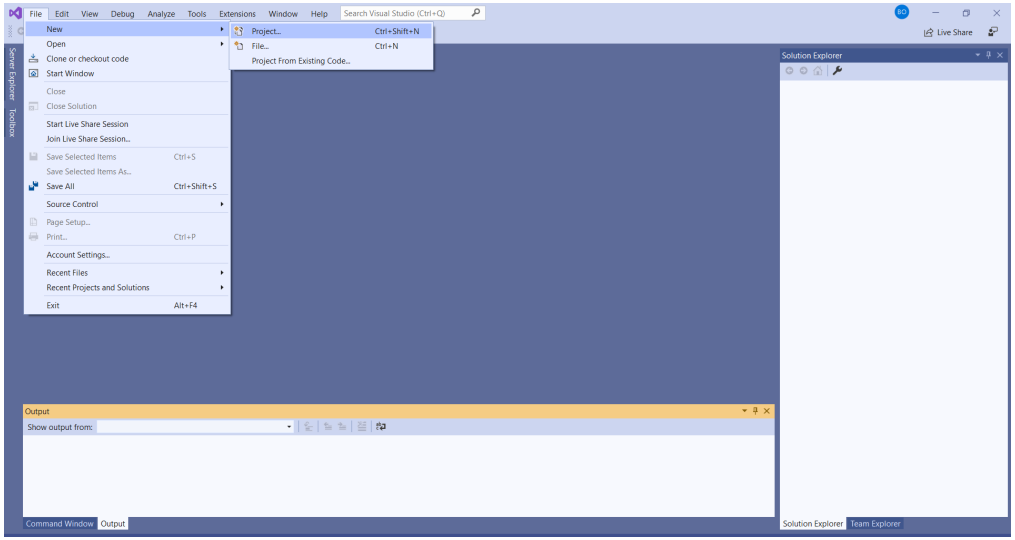
UPskill
Digital Skills & Jobs

- No processo de instalação, garantir que a framework .NET está selecionada:



Criar programas em C#

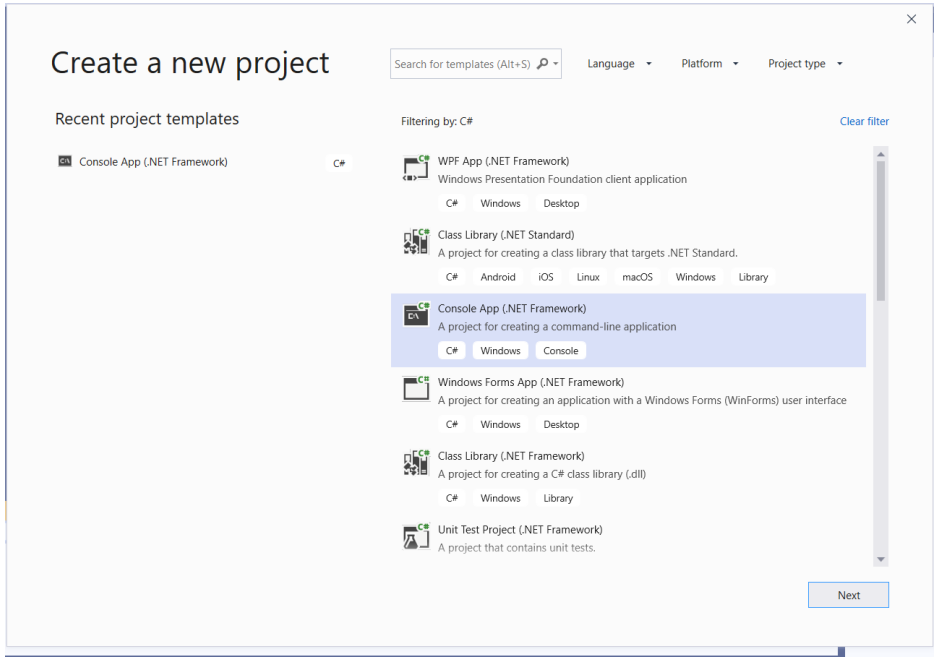
- Podemos criar novas aplicações em File -> New -> Project



Criar programas em C#

UPskill
Digital Skills & Jobs

- No próximo passo temos de escolher: Console application (aplicação que funcionará na consola/linha de comandos do sistema operativo, juntamente com a localização e nome do projeto.



Criar programas em C#

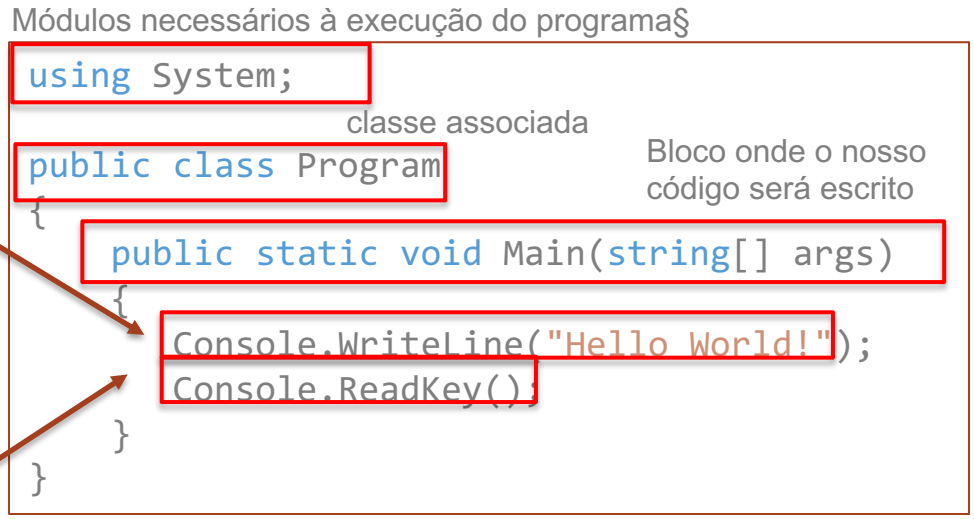
- Neste exemplo, podemos observar todos os artefactos para executar a aplicação de consola;
- O programa principal encontra-se no ficheiro: Program.cs (é o ficheiro que por defeito será executado);

Criar programas em C#

- Como alternativa ao Microsoft Visual Studio, Podemos utilizar um editor mais simples, o Microsoft Visual Studio Code:
  - https://docs.microsoft.com/en-us/dotnet/core/tutorials/with-visual-studio-code
- Podemos criar um projeto por linha de comandos e executar (respetivamente):
  - Dotnet new console
  - Dotnet run

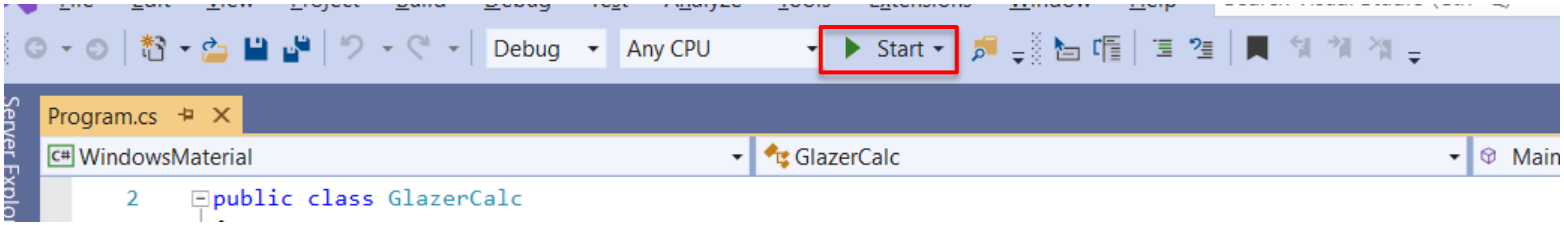
Um primeiro programa

- Este pequeno exemplo imprime na consola a mensagem: "Hello World";
- De seguida fica à espera que o utilizador pressione uma qualquer tecla para terminar.



Um primeiro programa

- Para executar (e compilar) o programa temos de utilizar o Start:



Um primeiro programa

- Se o código for introduzido corretamente, a seguinte mensagem será apresentada:

Um exemplo completo

UPskill
Digital Skills & Jobs

- Exemplo:

```
1   using System;
2   public class GlazerCalc {
3        public static void Main() {
4               double width, height, woodLength, glassArea;
5               string widthString, heightString;
6               widthString = Console.ReadLine();
7               width = double.Parse(widthString);
8               heightString = Console.ReadLine();
9               height = double.Parse(heightString);
10              woodLength = 2 * ( width + height ) * 3.25;
11
12              glassArea = 2 * ( width * height );
13              Console.WriteLine ( "The length of the wood is " + woodLength + " feet" );
14              Console.WriteLine( "The area of the glass is " + glassArea + " square metres" );
15                Console.ReadKey();
16       }
17  }
```

Vamos decompor o programa!

UPskill
Digital Skills & Jobs

- `(1) using System;`
  - É uma instrução C# que indica que pretendemos utilizar funcionalidades relacionadas com o grupo: System.
  - Permite-nos utilizar funcionalidades específicas da linguagem C#.

Vamos decompor o programa!

UPskill
Digital Skills & Jobs

- `(2) class GlazerCalc`
  - Um programa em C# é constituído por uma ou mais classes.
  - Podemos olhar para uma classe como uma caixa que contém dados e código (instruções) para responder a uma tarefa específica.
  - No nosso caso, a classe contém apenas um método (`public static void Main()`) para responder às necessidades mas podem existir vários métodos numa classe.

Vamos decompor o programa!

UPskill
Digital Skills & Jobs

- (3) `public static void main`
  - Esta linha representa a declaração de um método;
  - `public` significa que qualquer classe adicionalmente criada tem acesso a este método (mais informação no futuro);
  - `static` garante que o método está sempre presente na classe (não pode mudar) - mais informação no futuro;
  - `void` significa "nada" e significa que após o método ser executado, nada será devolvido. Isto é, é invocado e simplesmente termina. No futuro veremos métodos que efetivamente devolvem "algo";
  - `main` é o nome do método. É na realidade um método especial (Main é uma palavra reservada) que representa o método que será inicialmente executado na execução do programa. Sem este método, o sistema não sabe como vai executar o programa!

Vamos decompor o programa!

- (2) ()
  - Par de parêntesis sem conteúdo;
  - Indica ao compilador que o método nada recebe para suportar a sua execução!
  - Um parâmetro permite providenciar algo para o método conseguir executar;

Vamos decompor o programa!

- (2) (3) { e (17)(18) }
  - Par de chavetas;
  - São utilizadas aos pares: uma chaveta de início e uma chaveta de fim;
  - Permitem definir blocos, permitindo agrupar várias instruções;
  - Neste caso, estamos a agrupar um conjunto de instruções associadas a um bloco do método Main e à classe criada.

Vamos decompor o programa!

- (4) `double width, height, woodLength, glassArea`
  - double significa: "double precision floating point number“, e é um tipo de dados;
  - No nosso exemplo, necessitamos de armazenar informação que o programa pergunta ao utilizador e ainda os novos valores calculados pelo programa de forma a conseguirmos apresentar resultados;

Vamos decompor o programa!

UPskill
Digital Skills & Jobs

- (4);
  - O ponto e vírgula é obrigatório e indica o fim da instrução;
  - Todas as instruções do programa terminam com ;
  - Sem o ; , o compilador não sabe onde a instrução termina, originando erros de compilação;

Vamos decompor o programa!

UPskill
Digital Skills & Jobs

- (5) `string widthString, heightString;`
  - A leitura de dados a partir do teclado é realizada através de texto;
  - Após a leitura, convertemos os valores para números;
  - Estas variáveis vão conter a representação textual dos números introduzidos;

Vamos decompor o programa!

- (6) `widthString = Console.ReadLine();`
  - Instrução de atribuição que permite a alteração de valor de uma variável;
  - Neste caso, procedemos à leitura de uma linha de texto do teclado e colocamos o valor lido na variável: widthString.
  - `Console` indica uma classe de sistema (`System`) que possui um método (utilizamos o `.` para aceder a um método de uma classe): `ReadLine()` que é responsável por proceder à leitura de texto do teclado, retornando o valor lido para a variável: widthString;

Vamos decompor o programa!

- (7) `width = double.Parse(widthString);`
  - O método Parse tem a tarefa de converter o texto lido do teclado para um valor real;
  - Para cada caracter da string é realizada uma conversão para a representação decimal do valor;
  - De notar que entre parêntesis, o método possui a variável: widthString, indicando que o método recebe um parâmetro que representa um valor com que terá de trabalhar;
  - O método retorna o número real correspondente para a variável widthString;
  - As linhas 8 e 9 representam instruções similares mas para a altura da janela.

Vamos decompor o programa!

UPskill
Digital Skills & Jobs

- (10) `woodLength = 2 * ( width + height ) * 3.25;`
- (12) `glassArea = 2 * ( width * height );`
  - Atribuição de valores resultantes de um cálculo.
  - São utilizados os operadores de soma e multiplicação;
  - Os parêntesis são utilizados para definir a ordem das operações.

Vamos decompor o programa!

- (13) e (14) `Console.WriteLine`
  - Similar à invocação do método ReadLine;
  - No entanto, este método é utilizado para imprimir valores para o terminal/consola;
- (13) `"The length of the wood is "`
  - Representa um literal de string;
  - O texto é colocado entre aspas;
  - Este texto representa um valor e não uma instrução;
  - O texto representa o output que irá surgir no terminal de execução do programa.

Tipos de dados - inteiros

- C# possui uma grande variedade de tipos inteiros:

sbyte	8 bits	-128 to 127
byte	8 bits	0 to 255
short	16 bits	-32,768 to 32,767
ushort	16 bits	0 to 65,535
int	32 bits	-2,147,483,648 to 2,147,483,647
uint	32 bits	0 to 4,294,967,295
long	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
ulong	64 bits	0 to 18,446,744,073,709,551,615
char	16 bits	0 to 65,535

Tipos de dados: números reais

- float:
  - float averageIceCreamPriceInPence;
- double:
  - double univWidthInInches;
- Para uma maior precisão:
  - decimal rodsOvercraft;
- Float literal:
  - 2.5f
- Double literal:
  - 3.5
- Decimal literal:
  - 3.5M

Tipos de dados: Texto

- C# suporta o tipo char e string;
- Um literal do tipo char utilizer plicas (‘’) e do tipo string aspas (“”)
- Escape sequences:

Character	Escape Sequence name
\'	Single quote
\"	Double quote
\\	Backslash
\0	Null
\a	Alert
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical quote

- Verbatim string:@”exemplo” - ignora escape characters e mudanças de linha.

Tipos de dados: boolean

UPskill
Digital Skills & Jobs

- Boolean:
  - bool networkOK;
- Boolean literal:
  - networkOK = true;

Estrutura condicional IF

```
(...)
int selection;
Console.WriteLine("Window Type:\n 1 - Casement \n 2 - standard \n 3 - pation door");
selection = Int32.Parse(Console.ReadLine());
          if (selection == 1){
                //type 1
          }else{
                if (selection == 2){
                    //type 2
                }else{
                    if (selection == 3){
                        //type 3
                    }else{
                        Console.WriteLine("Invalid number");
                    }
                }
          }
(...)
```

The switch construction

```
(...)
int selection;
Console.WriteLine("Window Type:\n 1 - Casement \n 2 - standard \n 3 - pation door");
selection = Int32.Parse(Console.ReadLine());
          switch (selection){
                case 1: //type 1
                break;
                case 2: //type 2
                break;
                case 3: //type 3
                break;
                default: Console.WriteLine("Invalid number");
                break;
        }
(...)
```

do -- while loop

```
do{
      //statement or block
}while (condition);
```

while loop

```
while (condition){
        //statement or block
}
```

for loop

```
(…)
int i ;
i = 1 ;
while ( i < 11 ) {
      Console.WriteLine ( "Hello mum" ) ;
      i = i + 1 ;
}
(…)
```

Arrays

- Example

```
using System;

class Program
{
    static void Main()
    {
        int[] scores = new int[11];
        for(int i = 0; i < 11; i++)
        {
            scores[i] = int.Parse(Console.ReadLine());
        }
    }
}
```

Matrizes

- Exemplo:

```
int [,] board = new int [3,3];
board [1,1] = 1;
```

UPskill – C#

Introdução (C# procedimental)