Ficha de avaliação de capacidades

Programação C# (Parte 1)

**Regras**

Esta ficha tem como intuito avaliar o teu conhecimento e capacidades acerca das bases de programação em C#. Deverás, até as 12h50 de hoje, entregar a sua resolução por email, ou até por repositório até ao final, ou não será contabilizada. Durante este período, poderás consultar os materiais lecionados (slides), tendo sempre em consideração o tempo que te resta até à entrega. Se sentires que consegues escrever mais depressa em papel, poderás então escrever as tuas respostas numa folha, tirar foto ou digitalizar, enviando juntamente com a solução. Caso tenhas algum problema notifica-o no Slack. Um dos teus colegas poderá estar na mesma situação! Assim que entregares, notifica-me.

**Atenção :** o email enviado com a solução para a ficha deverá conter apenas um anexo (zip) com o nome FICHA\_CSHARP\_PrimeiroNome\_UltimoNome (ex: FICHA\_CSHARP\_Fabio\_Jesus.zip)

A ficha é composta por 4 (quatro) grupos:

1. **Verdadeiros e Falsos (50 pontos)**

As respostas deverão ser colocadas nos respetivos campos da grelha.

1. **Desenvolvimento (60 pontos)**

A resposta deve ser colocada abaixo da pergunta, ou se escreveres a resposta numa folha, marca apenas o número da questão. (ex: 1))

1. **Prático (90 pontos)**

As respostas devem ser colocadas no código fonte. Caso ocorra algum erro que cause o teu projeto a funcionar, respira fundo, comenta o código e continua. Não deixes que um erro mínimo te impeça de continuar a ficha, pois todo o código comentado será avaliado, e caso esteja parcialmente correto, será atribuída essa pontuação. Perguntas que tenham cotações diferentes apenas totalizam a pontuação total se forem apresentadas soluções para cada uma (ex: [iterativa 5pts / recursiva 10pts] resulta em 15 pontos se entregares ambas).

1. **Extras (50 pontos)**

São pontos extra, por isso, tal como nas outras fichas e testes, nunca contam para além de demonstrar o teu esforço. Aconselho-te a resolver estes exercícios assim que acabares a ficha.

Boa sorte!

# **Grupo I – Verdadeiros e Falsos (50 pontos)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| **F** | **V** | **V** | **F** | **F** | **F** | **V** | **V** | **V** | **V** |

* 1. A framework une ferramentas tais como o editor e o compilador para facilitar o desenvolvimento de software.
  2. As linguagens de programação são classificadas como DSL ou GPL.
  3. A leitura do conteúdo introduzido pelo utilizador pode ser realizada através da função Console.ReadLine();
  4. A negação de uma proposição é realizada através do operador **?** (ponto de interrogação)
  5. “1nome” é um nome válido para uma variável
  6. O tratamento de exceções permite corrigir problemas causados pelo programador.
  7. Após terminar a execução de um âmbito, todas as variáveis declaradas no seu interior ficam inacessíveis.
  8. A arquitetura .NET Framework é composta por Sistema Operativo, Modelos de aplicação, FCL e CLR.
  9. O *debugger* é útil para, durante em tempo de execução, detetar anomalias.
  10. De momento, o Visual Studio é apenas uma janela, onde podem ser instalados vários plugins que podem ser ancorados para uso posterior.

# **Grupo II – Desenvolvimento (60 pontos)**

1. Distingue as DSL de GPL. [6 pontos]

A DSL são linguagens de programação criadas com o objetivo de solucionar problemas em um domínio específico de aplicações, possibilitando ao usuário construir estruturas que modelem de forma clara e concisa funcionalidades específicas de um determinado domínio. As DSLs têm sido amplamente utilizadas como ferramenta de comunicação entre os analistas de negócio e desenvolvedores, pois permitem tratar soluções no nível de abstração do domínio.  
As GPLs são voltadas para produzir software em vários tipos de contexto e não incluem estruturas de linguagem designadas em domínios específicos.

1. Descreve as nove ferramentas de um IDE [9 pontos]

Editor: edita código fonte

Refactoring: Otimização do programa e resolução de erros (antes da execução)

Tests: Automatização de testes e algoritmos e programas

Deploy: Cria um pacote distribuível por diversos meios

Debugger: Assiste no processo de deteção e correção de erros em execução

Compiler: Compila os diferentes elementos desenvolvidos numa única aplicação ou biblioteca

Linker: Efetua a conexão entre os diferentes elementos de código

Modeling : Permite modelar elementos tais como classes e propriedades

Gerador de código : Gera código repetitivo que é utilizado com frequência

1. Identifica as sete características do C# [7 pontos]

Fortemente tipada – restringi os valores ao tipo da variável (variável tem um tipo bem definido e precisa ser informado no momento de sua declaração)

Multiparadigma – suporta o programador na realização de tarefas que exijam diferentes paradigmas de programação, sendo a orientação a objetos e standard

Simples – as instruções podem ser realizadas de varias formas

Tudo é um objeto – todas as classes criadas com esta linguagem derivam do tipo objeto

Controlo de versões – cada aplicação gerada contém informações acerca da versão de código, de forma que a biblioteca com o mesmo nome e versões diferentes podem coexistir

Suporte a código legacy – permite importar bibliotecas desenvolvidas em outras linguagens

Ambiente controlado – os programas desenvolvidos utilizam um conjunto de ferramentas de gestão de recursos que reduzem a possibilidade de ocorrer um erro por parte do programador

1. Distingue IDE de Framework [6 pontos]

IDE é um conjunto de ferramenta que permite o desenvolvimento de softwares, enquanto a FrameWork fornece um conjunto de ferramentas que auxiliam no desenvolvimento de softwares, como bibliotecas e compiladores.

1. Descreve boxing, unboxing e casting. Apresenta exemplos para cada processo. [6 pontos]

Boxing consiste em colocar um valor primitivo em uma referência de memória.

Unboxing retorna o valor armazenado no objeto à variável ou outa do mesmo tipo.

Casting – consiste na conversão de um tipo para outro, desde que sejam compatíveis entre si, através da associação a uma variável do tipo desejado. Para tipos de dados nos quais não seja possível realizar operações de casting, utiliza-se métodos próprios do tipo (Parse e TryParse), ou a classe Convert.

1. Descreve as formas de juntar texto com variáveis, apresentando exemplos. [6 pontos]

Ex. Console.WriteLine(“Introduza seu nome”);  
Var nome = Console.ReadLine();

Concatenação 🡪 Console.WriteLine(“Olá” + “nome”);

Composição: utilizando o termo “@” e preenchendo elementos indexados através de chavetas 🡪Console.WriteLine(@“Olá” + {0}, nome);

* Interpolação através do termo “$” e preenchendo diretamente os espaços entre chavetas🡪Console.WriteLine($“Olá” + {nome});

1. Identifica o que pode ser realizado com a .NET Framework. [6 pontos]

Web

Gaming

Mobile

Machine learning & AI

Desktop

Internet of things

1. Descreve a utilidade de um *breakpoint.* Apresenta um exemplo. [6 pontos]

É um ponto de parada no seu programa, ou seja, você define algum breakpoint na sua aplicação e quando estiver executando você poderá analisar linha por linha de código, facilitando encontrar possíveis erros que possa esta dando na aplicação.

1. Descreve a hierarquia de uma aplicação [8 pontos]

Solução – “recipiente” que contém pastas e projetos e outros ficheiros que se encontrem relacionados a solução (extensao.sln)

Pastas – estejam estas sob soluções ou projetos, o seu único intuito é aglomerar uma serie de elementos sobre um nome característico sem a necessidade de qualquer produção de código fonte

Projetos – contém todos os ficheiros que são compilados em:

* + Executáveis – aplicações de desktop (ficheiros \*.exe)
  + Bibliotecas – contém código fonte reutilizável (ficheiros \*.dll)
  + Aplicações web

# **Grupo III – Desenvolvimento (90 pontos)**

1. Na solução em anexo com o enunciado, criar uma biblioteca chamada Grupo3. Nesta biblioteca deverá ser criada uma função chamada GoodbyeWorld que apresente, na consola, “Good bye world!”. **[10 pontos]**



1. Na biblioteca GrupoIII, no ficheiro Exercicio2.cs:
   1. Cria uma função que solicite o nome do utilizador, e apresente “Olá” seguido do nome

**[4 pontos]**



* 1. Na função “QuantasPatas”, implementa o algoritmo necessário para devolver o número de patas de um conjunto de vacas, porcos e galinhas. **[4 pontos]**



* 1. Cria uma região de código **[2 ponto]**
  2. Cria uma função que verifique se um número introduzido é par ou ímpar. Se criaste uma região, coloca a função dentro da região **[4 pontos]**



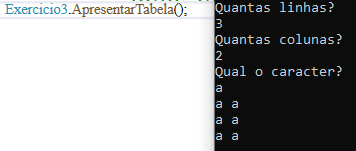
* 1. Cria uma função que calcule a soma de dois valores inteiros. Caso estes sejam iguais, calcula o triplo da soma. **[4 pontos]**



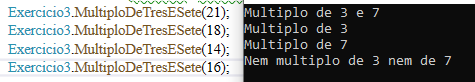
* 1. Cria uma função que apresente a potência de um número. Deverá requisitar o número e a potência. **[4 pontos]**



1. Na biblioteca GrupoIII, no ficheiro Exercicio3.cs:
   1. Cria uma função que solicite ao utilizador dois números inteiros (m e n) e um caracter, e que apresente uma tabela mxn preenchida com o caracter. **[10 pontos]**



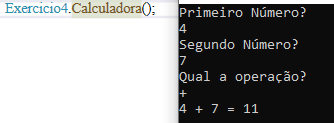
* 1. Cria uma função que verifica se um número é múltiplo de 3 e/ou 7. **[10 pontos]**



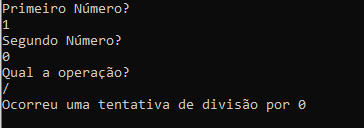
* 1. Cria uma função que apresente o fatorial de um número. **[5 pontos iterativa / 8 recursiva]**

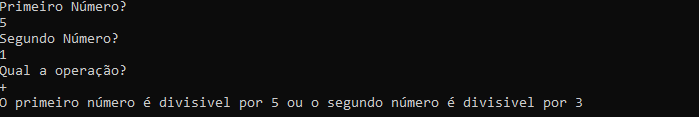
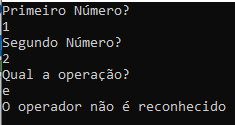


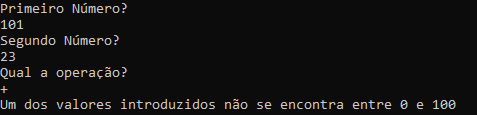
1. Na biblioteca GrupoIII, no ficheiro Exercicio4.cs:
   1. Cria uma função que solicite dois números, a e b, e um caracter op. De acordo com ‘op’ deverão realizar o conjunto de operações de soma, subtração multiplicação, divisão e resto de divisão e apresentar o seu resultado. **[10 pontos]**

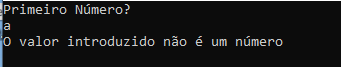


* 1. No mesmo método, deverão fazer a validação dos valores que foram introduzidos. Tal validação pode ser realizada através de uma função que se encontra no ficheiro, chamado ValidarInput. Esta função deverá ser invocada a seguir às instruções que solicitam os números e o caracter. Ao realizar esta validação, são lançadas exceções referentes ao tipo de problema que ocorreu. Deve ser realizado o tratamento para cada uma destas exceções. Caso alguma das exceções ocorram, deve surgir uma mensagem descritiva do que ocorreu. **[15 pontos]**







# **Grupo IV – Extras (50 pontos)**

1. Às 13h00, coloca a tua resolução num repositório do GitHub, enviando o link juntamente com a resolução. **[5 pontos]**
2. Documenta todas as funções com as quais interagiste durante o teste **[5 pontos]**
3. Pedra, Papel, Tesoura. Na solução encontrarás um projeto chamado GrupoIV. Nesse projeto existe uma classe chamada PedraPapelTesoura. Nela:
   1. A função Start é o ponto principal para a execução do jogo. Nela deverás requisitar o número de jogadas necessárias para um jogador vencer (à melhor de…). Esta função deverá ser também responsável por, assim que um jogador vença, apresentar o resultado. **[5 pontos]**
   2. Deverá ser solicitado o nome do jogador. **[2 pontos]**
   3. Deverá ser validado se um jogador ou o computador venceu. **[10 pontos mediante eficiência]**
   4. Deverá ser selecionada aleatoriamente a jogada do computador. Tal poderá ser feita recorrendo à função “Aleatório”, onde é gerado um número entre um mínimo e um máximo. **[5 pontos]**
   5. Após solicitar a jogada do utilizador, e gerar a jogada do computador, deverá recorrer à função Display para apresentar a simulação da jogada em tempo real. Nesta função, são solicitados os argumentos p1 e p2, ou seja, a jogada de cada jogador (r, p, s). O terceiro parâmetro, *rounds* é apenas a quantidade de vezes que as mãos sobem e descem (o suspense). **[5 pontos]**
   6. Após a jogada terminar é necessário identificar o vencedor. **[8 pontos]** Não esquecer que:
      1. Pedra vence contra tesoura
      2. Tesoura vence contra papel
      3. Papel vence contra pedra
   7. Após o fim da jogada, os números de jogadas vencidas por cada participante devem ser atualizados. **[5 pontos]**