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]

As DSL (Domain-specific Languages) são utilizadas para representação de conteúdo especifico tal como páginas Web, processamento matemático, bases de dados e sintetização de som. Alguns exemplos são HTML, MATLAB, SQL e Csound. As DSL são fáceis de aprender, no entanto têm limitações ao nível da aplicabilidade. A sua eficiência do processar é muito menor comparado com as GPL.

As GPL (General-Purpose Languages) são utilizadas para produzir software em vários tipos de contexto e não inclui estruturas de linguagem designadas em domínios específicos. Um exemplo de GPL é o C#. As GPL têm uma curva de aprendizagem reduzida, no entanto são uma linguagem demasiado abstrata para ser compreendida.

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

Editor – permite a edição do código fonte.

Refactoring – otimização do programa e resolução de erros.

Testes – automatização de testes a 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 frequentemente.

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

Fortemente tipificado, multiparadigma, simplicidade, tudo é um objecto, controlo de versões, suporte a código legacy e ambiente controlado.

1. Distingue IDE de Framework [6 pontos]

Os IDE (Integrated Development Environment) englobam um conjunto de ferramentas destinadas ao desenvolvimento de software tendo como principal objectivo reduzir a configuração necessária para a realização de todo o processo de desenvolvimento.

As Frameworks foram criadas de forma a facilitar o desenvolvimento, através de um conjunto de standards e módulos que podem ser reutilizados. Consistem em abstrações de software capazes de fornecer ao programador um conjunto de ferramentais tais como compiladores, bibliotecas e funcionalidades genéricas.

O IDE é utilizado para escrever e compilar código enquanto a Framework é um componente já existente que pode ser integrado, com vários standards e módulos, de forma a não ser necessário “reinventar a roda”.

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

O boxing consiste na atribuição de um valor de tipo primitivo a um tipo que permite valor nulo, para que este possa manter o valor.

int i = 123;

object o = i; //boxing

O unboxing é o processo oposto, em que se retorna o valor armazenado no objeto à variável.

(continuando o exemplo anterior)

int i = (int)o; //unboxing

O casting consiste na conversão de um tipo para outro através da associação a uma variável do tipo desejado. Efetuado entre tipos compatíveis entre si, número com número, texto com texto, etc.

double ab = 12.4;

int o = (int)ab; //informação é perdida ao converter de double para int (as casas decimais).

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

Através da Concatenação, que pode ser de 3 formas distintas:

Composição, utilizando o termo “@” e preenchendo elementos indexados através de chavetas.

Console.WriteLine(@”Valor {0}”, val);

Interpolação através do termo “$” e preenchendo diretamente os espaços entre chavetas.

Console.WriteLine($”Valor {val}”);

E simplesmente ir adicionando através do termo “+”.

Console.WriteLine(“Valor” + val);

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

Inicialmente utilizada com o intuito de desenvolver aplicações web, abrange agora também aplicações móveis, plugins e outros sistemas operativos (inicialmente exclusivo ao Windows). Exemplos: Web, Mobile, Desktop, Gaming, Machine Learning & AI e Internet of Things.

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

O breakpoint é uma técnica de debugging. Ao ser utilizado este permite correr o código passo a passo e entender onde se encontram os erros, sendo possível também testar e alterar os valores das variáveis. Por exemplo se um ciclo não está a dar o resultado pretendido, colocar um breakpoint será útil para tentar entender o porquê de o resultado não ser o desejado.

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

No topo da hierarquia encontra-se a nossa solução. Dentro desta podem se encontrar projectos, pastas e classes. Os projetos podem conter classes e pastas enquanto as pastas podem conter mais projetos e classes. A solução mantém as ligações entre projetos, configurações gerais dos projetos e outros ficheiros que se encontrem relacionados com a solução. As pastas servem para aglomerar uma série de elementos sobre um nome característico em a necessidade de qualquer produção de código fonte. Os projectos contêm todos os ficheiros que são compilados em executáveis, bibliotecas e 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]**



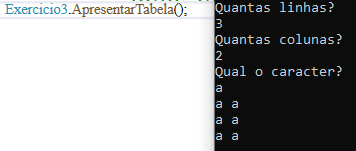
* 1. Cria uma função que calcule a soma de dois valores inteiros. Caso estes sejam iguais, calcula o triplo da soma. **[4 pontos]**



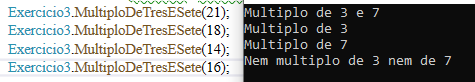
* 1. Cria uma função que apresente a potência de um número. Deverá requisitar o número e a potência. **[4 pontos]**



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]**



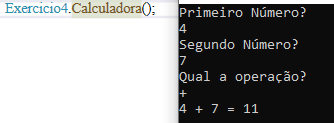
* 1. Cria uma função que verifica se um número é múltiplo de 3 e/ou 7. **[10 pontos]**



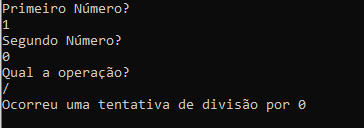
* 1. Cria uma função que apresente o fatorial de um número. **[5 pontos iterativa / 8 recursiva]**

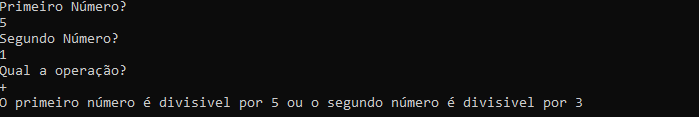
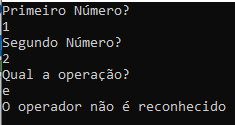


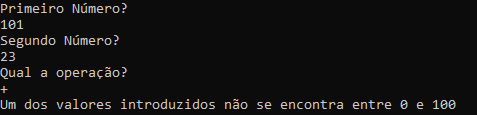
1. Na biblioteca GrupoIII, no ficheiro Exercicio4.cs:
   1. Cria uma função que solicite dois números, a e b, e um caracter op. De acordo com ‘op’ deverão realizar o conjunto de operações de soma, subtração multiplicação, divisão e resto de divisão e apresentar o seu resultado. **[10 pontos]**

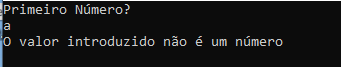


* 1. No mesmo método, deverão fazer a validação dos valores que foram introduzidos. Tal validação pode ser realizada através de uma função que se encontra no ficheiro, chamado ValidarInput. Esta função deverá ser invocada a seguir às instruções que solicitam os números e o caracter. Ao realizar esta 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]**