Teste

Programação C# (Parte 2)

**Regras**

Este teste 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 (40 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 (100 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 o teste, 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 o teste.

Boa sorte!

# **Grupo I – Verdadeiros e Falsos (40 pontos)**

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

* 1. É possível, através de um índice, alterar um caracter de uma string.
  2. Os vetores são de dimensão fixa.
  3. A class Path permite criar e eliminar pastas.
  4. Na especificação das funções que arredondam valores, o floor e o ceiling arrendondam de forma diferente com base no seu sinal
  5. Os métodos de extensão são utilizados para adicionar novas funcionalidades a classes existentes.
  6. As exceções são lançadas com o launch
  7. Os tipos genéricos permitem o desenvolvimento de funções que podem ser utilizadas em diferentes tipos, que serão posteriormente especificados.
  8. Os enums permitem valores repetidos

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

1. Descreve o namespace. Apresenta exemplos. **[10 pontos]**

O namespace serve para organizar a solução. Segue uma forma hierárquica. Temos o namespace built-in, o global e o local.

O built-in é onde ficam as soluções, ou seja, se tiver um namespace como “Empresa.Alimentacao”, é nele que serão colocadas todas as soluções que sejam relacionadas com a alimentação de uma certa empresa. O global é relacionada com os projetos, ou seja, dentro do built-in teria ainda tudo organizado por meses, sendo que “Empresa.Alimentacao.Janeiro” contem tudo o que for relacionado com alimentação no mês de janeiro. E por fim, dentro dos globais temos ainda os locais que correspondem aos nomes das classes, ou seja, “Empresa.Alimentacao.Janeiro.Custo” por exemplo.

1. Descreve o atributo. Apresenta exemplos **[4 pontos]**

Os atributos são informação adicional associada ao código. Por exemplo, se eu tiver um método “sum(int num1, int num2)” que faz a soma entre dois números inteiros e tentar por 2 números com casas decimais vou ter um erro. O método devia aceitar números com casas decimais também. Se criar outro método “sumAll(double num1, double num2)” e colocar em cima do método inicial o atributo “[Obsolete(“Este método já não é utilizado. Para permitir números com casas decimais utilize o método sumAll(double num1, double num2)”)]”, sempre que chamar a função “sum”, vou ter um aviso que o “sum” já não é usado e que o “sumAll” permite casas decimais também.

1. Distingue delegate, event, action e func. **[8 pontos]**

Delegate – é um tipo que representa referências a métodos com parâmetros e tipo de retorno igual.

Event – é uma variante do delegate que não pdoe ser evocado fora da classe onde foi declarado.

Action – Funções representadas por um tipo e um nome que não têm um valor de retorno.

Funcs -- Funções representadas por um tipo e um nome com valor de retorno. O ultimo tipo é o tipo de retorno.

1. Distingue listas de arrays. Exemplifica **[8 pontos]**

As listas têm uma dimensão dinâmica. Não precisamos de especificar a quantidade de índices.

Por exemplo, se tiver uma lista para guardar o nome de pessoas inscritas para um evento, posso ir adicionado os registos que quiser através do método “add”. Enquanto se tivesse um array, teria de indicar a quantidade de pessoas que se iriam inscrever. Como é impossível fazer uma previsão certa, a lista permite adicionar quantas pessoas necessárias.

1. Descreve o tipo enum. Exemplifica **[6 pontos]**

Serve para declarar uma lista de valores constantes com um nome associado a cada valor. Desta maneira é possível tratar valores numéricos através de nomes.

Exemplo:

enum Semana

{

Segunda = 0,

Terca = 1,

Quarta = 2,

Quinta = 3,

Sexta = 4,

Sabado = 5,

Domingo = 6,

}

Console.WriteLine(Semana.Terca); // Terca

Console.WriteLine((int)Semana.Terca); // 1

1. Distingue classes de structs. Demonstra **[8 pontos]**

As classes são o “template” para criar objetos. Contêm propriedades e ações.

As structs são referencias dor valor em vez de referencia em memoria. E também não podem ser alteradas, não podem conter construtores vazios, etc…

1. O que é açúcar sintático? Dá exemplos do seu uso. **[10 pontos]**

São características das linguagens de programação que facilitam o desenvolvimento.

Por exemplo, utilizar o tipo “var” em vez de um tipo certo. E os getters e setters, que permite obter o valor e modifica-lo, ou seja, se tiver a classe “Pessoa” com a propriedade “nome”, consigo obter o valor ao chamar “Pessoa.nome” e muda-lo ao fazer “Pessoa.nome = “Teste””.

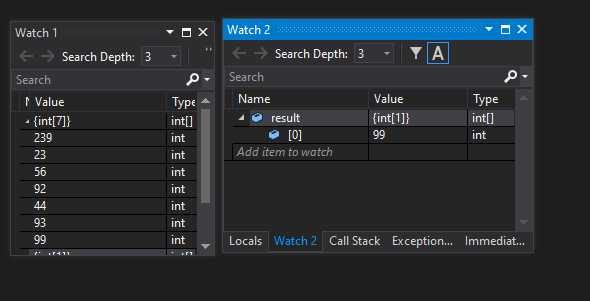
1. Descreve o LINQ. Exemplifica **[6 pontos]**

Language Integrated Query. Server para efetuar operações de pesquisa sobre dados. Se tiver uma lista “lista” com 10 números e quiser obter os números pares fazia algo como “lista.Select(x => x % 2 == 0)”.

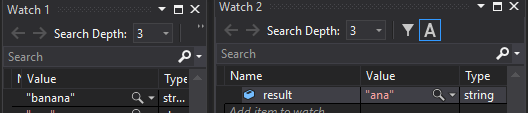
# **Grupo III – Desenvolvimento (100 pontos)**

1. Avaliação do código em geral **[10 pontos]**
   * Disposição de elementos nos ficheiros adequados
   * Respeito às normas de programação definidas
   * Uso do conteúdo lecionado de forma correta e responsável
   * Legibilidade e açúcar sintático
2. Os exercícios deverão ser colocados na pasta FirstExercise:
   1. No ficheiro OneA, sem recorrer a LINQ, cria uma função que receba um conjunto de inteiros e retorne apenas os valores > 50, ímpares e divisíveis por 9; **[6 array / 4 list]**

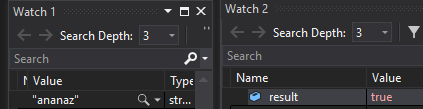
No seguinte exemplo, temos um conjunto original com 7 numeros à esquerda, e o resultado da operação à direita.



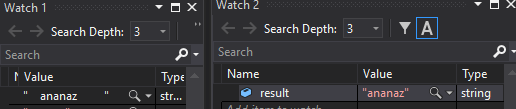
* 1. No ficheiro OneB cria funções que recebam uma string e: **[5 pontos]**
     1. Retorne apenas a metade (arredondada para baixo) da string



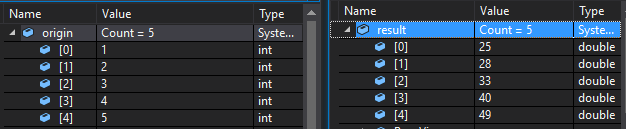
* + 1. Valide se a string começa com “a” e acaba com “z”



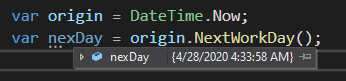
* + 1. Remova os espaços à volta da string e retorne a string “limpa”



* 1. No ficheiro OneC cria uma função que receba uma lista de inteiros e retorne a lista com o resultado da operação 22+8\*3 **[3 pontos / 2 pontos LINQ]**



* 1. Cria um método de extensão que permita obter a próxima data útil (se amanhã for sábado, ou domingo, retorna a data da próxima segunda). **[15 pontos]**



1. Os exercícios deverão ser colocados na pasta SecondExercise:

Tenho uma gaveta com várias divisórias. Como o seu interior é sensível, não quero colocar objetos que sejam afiados. Por vezes, quando a tento abrir fica presa, e assim não consigo aceder ao conteúdo.

* 1. Cria uma interface que represente algo seja afiado, capaz de cortar **[6 pontos]**
  2. Cria uma classe que identifique uma faca, através de propriedades e métodos que aches necessários **[10 pontos]**
  3. Cria um atributo chamado Stuck, que apenas pode ser aplicado a classes, não podendo existir, explicitamente, mais que um destes atributos associados a uma classe. Aplica-o à classe Drawer. **[7 pontos]**
  4. Cria sete classes distintas, onde três das quais devem implementar a interface produzida. Não é necessário elaborar muito! **[7 pontos]**
  5. Altera a classe Drawer, para que, através de um indexador seja possível organizar os objetos em divisórias. Não deverá permitir que sejam introduzidos objetos afiados **[15 pontos]**
  6. Cria uma exceção que indique que a gaveta está presa. Implementa uma forma de lançar a exceção quando é necessário aceder a uma divisória da gaveta, com base num valor aleatório. **[7pontos]**

# **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. Altera os namespaces dos projetos de forma a que estes sejam representativos da solução e dos projetos. **[10 pontos]**
4. Na biblioteca Extras completa o código da classe ContentSerializer. Altera apenas o código na zona indicada pelos comentários. **[15 pontos]**
5. Os exercícios deverão ser colocados na pasta ThirdExercise:
   1. Cria uma classe que te permita realizar as operações básicas (soma, subtração, multiplicação e divisão) a uma calculadora. Esta deverá manter apenas o valor em memória, que é atualizado após realizar uma operação. Deverás aplicar os conceitos que deste de forma a organizar o teu código da melhor forma. **[10 pontos]**
   2. Atualiza a tua classe para adicionares os mesmos métodos, sem que seja necessário um objeto. **[5 pontos]**