Laboratório 2 – Tipos de Dados Básicos em C#

Este laboratório visa introduzir os tipos de dados básicos de C#, mostrar como funciona a conversão entre tipos e introduzir as operações básicas de manipulação de strings.

1 Trabalhando com tipos-valor fundamentais byte, int e long

1. Crie um novo projeto de console com o nome “Laboratorio2”.

2. Dentro do método Main, crie uma variável local do tipo byte com o nome b. No início do método Main, digite o seguinte:

```
byte b;
```

3. Coloque o valor inicial da variável como o valor máximo que pode ser armazenado em um byte. Faça isso como demonstrado abaixo:

```
b = byte.MaxValue;
```

4. Mostre o resultado na janela de console:

```
Console.WriteLine("Valor maximo de byte: " + b);
```

5. Dentro do método Main, crie uma variável local do tipo int com o nome i. Digite o seguinte:

```
int i;
```

6. Coloque o valor inicial da variável como o valor máximo que pode ser armazenado em um int. Faça isso como demonstrado abaixo:

```
i = int.MaxValue;
```

7. Mostre o resultado na janela de console:

```
Console.WriteLine("Valor maximo de int: " + i);
```

8. Dentro do método Main, crie uma variável local do tipo long com o nome l**.** Digite o seguinte:

```
long l;
```

9. Coloque o valor inicial da variável como o valor máximo que pode ser armazenado em um long. Faça isso como demonstrado abaixo:

```
l = long.MaxValue;
```

10. Mostre o resultado na janela de console:

```
Console.WriteLine("Valor maximo de long: " + l);
```

2 Trabalhando com strings

1. Crie duas strings como mostrado abaixo:

```
string strPrimeira = "Alo ";
string strSegunda = "Mundo";
```

2. Crie uma terceira string e a inicialize com o resultado da concatenação das duas strings anteriores:

```
string strTerceira = strPrimeira + strSegunda;
```

3. Mostre o resultado na janela de console:

```
Console.WriteLine(strTerceira);
```

4. Calcule o tamanho da string strTerceira utilizando o seguinte código:

```
int cchTamanho = strTerceira.Length;
```

5. Crie uma string para guardar o resultado da seguinte forma:

```
string strQuarta = "Tamanho = " + cchTamanho.ToString(); Console.WriteLine(strQuarta);
```

6. Chame o método String.Substring para remover uma parte de uma string. Mostre o resultado:

```
Console.WriteLine(strTerceira.Substring(0, 5));
```

3 Trabalhando com objetos do Framework

1. Para criar um objeto DateTime é necessário chamar o operador new, como no seguinte código:

```
DateTime dt = new DateTime(2015, 04, 23);
```

2. Como os demais tipos, a classe DateTime possui um método ToString para converter o objeto para uma string, como mostrado abaixo:

```
string strQuinta = dt.ToString();
```

3. Mostre o resultado da seguinte maneira:

```
Console.WriteLine(strQuinta);
```

4 Exercícios

1. Além de tipos inteiros, C# também suporta *float*, *double*, e *decimal* o qual garante uma boa precisão para trabalhos com valores monetários. Escreva algum código para testar variáveis destes novos tipos e mostrar o resultado na tela do console.

2. Busque na documentação da biblioteca de classes do .Net Framework ou .Net Core, novas operações da classe String. Faça experiências e mostre os resultados na tela.

3. Busque na documentação da biblioteca de classe do .Net Framework ou .Net Core, novas operações da classe *DateTime*. Faça experiências e mostre os resultados na tela.

4. Converter dados em C# se dá de forma implícita ou explícita. Digite o seguinte exemplo e note que a conversão para um tipo "maior" é feita de forma implícita e, para um tipo "menor" de forma explícita.

```
static void Main(){
        int i = 10;
        float f = 0;
        f = i; //conversão implícita, sem perda de dados.
        System.Console.WriteLine(f);
        f = 0.5F;
        i = (int) f; //conversão explícita, com perda de dados.
        System.Console.WriteLine(i);
}
```

5. Além das conversões realizadas pela linguagem C#, podemos utilizar um mecanismo fornecido pelo próprio Framework, que é independente da linguagem utilizada. Este mecanismo é a classe *System.Convert*. Escreva um programa para testar os diferentes métodos de conversão. Como exemplo, utilize as seguintes linhas de comandos:

```
string stringInteiro = "123456789";
int valorStringInteiro = Convert.ToInt32(stringInteiro);
Console.WriteLine(valorStringInteiro);
Int64 valorInt64 = 123456789;
int valorInt = Convert.ToInt32(valorInt64);
Console.WriteLine(valorInt);
```

Adicione agora as seguintes linhas ao seu programa:

```
string stringInteiroGrande = "99999999999999999999999999999999999999999999999";
int valorStringInteiroGrande = Convert.ToInt32(stringInteiroGrande);
```

Note que ao tentar converter uma string com um valor que irá estourar a capacidade máxima que o tipo Int32 suporta a seguinte exceção será gerada:

```
System.OverflowException: Value was either too large or too small for an Int32.    at
System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)    at
<programa>.<classe>.Main(String[] args) in <diretorio do programa\arquivo.cs>:<linha>
```

Da mesma forma ao tentar converter uma string contendo letras ao invés de algarismos outra exceção será gerada. Para que isto não aconteça em uma conversão de tipos, o .NET Framework 2.0 trouxe o método TryParse para os tipos valor, que retorna um boolean com o valor true caso a operação tenha sido bem sucedida, ou com o valor false caso contrario.

6. Escreva um programa para testar o método TryParse. Como exemplo, utilize as seguintes linhas de comandos:

```
string stringInteiro = "123456789";
int valorStringInteiro;
bool conversao1 = Int32.TryParse(stringInteiro, out valorStringInteiro);
Console.WriteLine("Conversão efetuada:" + conversao1 + " Valor: " + valorStringInteiro);

string stringInteiroGrande = "99999999999999999999999999999999999999999999999";
int valorStringInteiroGrande;
bool conversao2 = Int32.TryParse(stringInteiroGrande, out valorStringInteiroGrande);
Console.WriteLine("Conversão efetuada:" + conversao2 + " Valor: " + valorStringInteiroGrande);
string stringLetras = "abc";
double valorStringLetras;
bool conversao3 = Double.TryParse(stringLetras, out valorStringLetras);
Console.WriteLine("Conversão efetuada:" + conversao3 + " Valor: " + valorStringLetras);
```

7. Devemos ter muito cuidado com os métodos de conversão de valores, pois arredondamentos são executados de formas diferentes. Verifique o comportamento do seguinte trecho de código.

```
double valorFracionado = 4.7;
int valorInteiro1 = (int) valorFracionado;
int valorInteiro2 = Convert.ToInt32(valorFracionado);
Console.WriteLine("Conversao explicita = " + valorInteiro1);
Console.WriteLine("Conversao metodo Convert = " + valorInteiro2);
```

8. O comando *WriteLine* tem uma sintaxe específica para a exibição de valores de várias variáveis. Teste o trecho abaixo. Explique como o mesmo funciona.

```
int x = 10;
double y = 3.4;
Console.WriteLine("{0}  {1}",x,y);
```