FACET
PÓS-GRADUAÇÃO

Projeto de Banco de Dados e OO .NET

Revisão C#

Prof: Foad Mobini Kesheh
foadmk@gmail.com

Console

- Console.Write()
- Console.WriteLine()
- string nome = Console.ReadLine()

Tipos Básicos

- int
- float
- double
- decimal
- string
- bool
- var

Conversão de Tipo

- int someNumber = 1000
- string someString = someNumber.ToString()
- float = int
- int = float (erro)
- int = Convert.ToInt32(float) //arredonda
- int = (int) float //trunca
- int = int.Parse(string)
- Int.TryParse(string, out int x)

DateTime

- DateTime myDate = new DateTime() //zerado
- DateTime myDate = DateTime.Now;
- DateTime myDate = new DateTime(1999,10,5)
- myDate.AddDays(-5)
- string formattedDate = string.Format(“Date is {0:dd/MM/yyyy HH:mm:ss}”, myDate)
- DateTime loadedDate = DateTime.ParseExact(loadedString, "dd/MM/yyyy", null);

Exercício 1 – C#

- Teste o console.
- Teste os tipos básicos
- Teste a variável DateTime.
- Faça um programa para ler uma data entrada pelo usuário e depois mostre essa data na tela usando a tipo DateTime.

Operações

- Adição +
- Subtração –
- Multiplicação *
- Divisão /
- Resto da Divisão %
- Ordem: () */ +- =

Operadores condicionais

```
if (condição) {código}
else if (condição) {código}
else {código}

switch(input) {
        case “caso1”: { código }; break;
        case “caso2”: { código }; break;
        default: {código}
}
```

Comparações

- Igual: ==
- Diferente: !=
- Maior: >
- Menor: <
- Maior Igual: >=
- Menor Igual: <=

Operadores Lógicos

- Operador E: &&
- Operador OU: ||
- Operador Inversão(Negado): !

Constantes

- const double PI = 3.14;

Exercício 2 – C#

- Crie um programa que solicite 3 números: A, B e C
- Imprima os números em ordem crescente.

Manipulação de Strings

- string b = “teste “+ a + “ teste”
- string b = string.Format(“teste {0} teste”, a)
- string b = $“teste {a} teste”

- b[0] -> “t”
- b.Trim() -> remove os espaços no início e no fim
- b.ToUpper() -> maiúsculas
- b.ToLower() -> minúsculas
- b.Length -> tamanho da string
- b.Replace(“teste”,”ola”) ->substituir

Manipulação de Strings

- b.IndexOf(“val”) => encontra a primeira ocorrência
- b.LastIndexOf(“val”) => encontra a última ocorrência
- b.Substring(4, 5) => gera uma string de tamanho 5 a partir do 4º caractere
- b.Remove(4, 5) => remove da string b 5 caracteres a partir do 4º caractere
- b.Insert(6, “ Teste “) => insere na posição 6 a string “ Teste “

Escaping

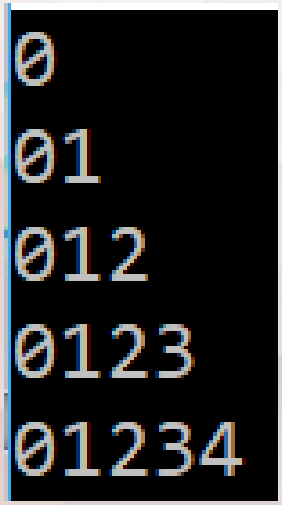
- “estava \“barato\”” => estava “barato”
- “D:\\” => D:\
- \t => caracter tab
- \n => nova linha
- \a => beep

Loops

- for([valor inicial]; [condição]; [incremento])
- while([condição]) => roda enquanto a condição for verdadeira
- do {código} while ([condição]); =>roda uma vez mesmo se a condição for falsa
- break => quebra o loop

Exercício 3 – C#

- Faça um programa que crie uma saída como a abaixo:



Math

- Math.Ceiling(x) => arredonda para cima
- Math.Floor(x) ou Math.Truncate(x) => arredonda para baixo
- Math.Round(x, digitos) => arredonda para o mais próximo
- Math.Max(x,y) => maior entre x e y
- Math.Min(x,y) => menor entre x e y
- Math.Sqrt(x) => Raiz de x
- Math.Pow(x,y) => x elevado a y
- Math.PI => valor de PI

Número Aleatórios

- Random rand = new Random();
- int x = rand.Next(min, max);
- double x = rand.NextDouble(); =>entre 0 e 1

Enum

```
enum DiasDaSemana {
  Segunda,
  Terca,
  Quarta,
  Quinta,
  Sexta,
  Sabado,
  Domingo= 6  //definir numero
};

DiasDaSemana dia = DiasDaSemana.Sabado;
Int index = (int) dia; // valor 5
string dia = DiasDaSemana.Sabado.ToString();
```

Struct

Similar a classe, exceto: não pode ser estendido, não pode definir construtor, dados não podem ser inicializados

```
struct Carro
{
  public string marca;
  public string motor;
  public int potencia;
  public decimal motorLitros;
}
```

Exercício 4 – C#

- Crie uma struct chamada triangulo com parâmetros base e altura e tipo de triângulo (Isósceles, Equilátero, Retângulo), inicialize os parâmetros com valores aleatórios, o tipo de triângulo deve ser uma enum.

Arrays

- int[] lista = new int[5];
- int[] lista = new int[5] { 1, 4, 6, 7, 8 };
- int[] lista = { 1, 4, 6, 7, 8 };
- foreach (int l in lista)
- lista.Length => tamanho do array

Collections - ArrayList

```
ArrayList array = new ArrayList();
array.Add(“First”);
array.Add(4.5f);
array.Insert(1, “Sec”); //insere no index 1
var x = array[2];
array.Count //tamanho
array.Sort();
array.Reverse();
array.Remove(“Sec”);
array.RemoveAt(1);
array.RemoveRange(0,2);
array.Contains(“First”);
array.GetRange(0,2);
array.Clear();
```

Collections - Hashtable

```
Hashtable hash = new Hashtable();
hash.Add(“Quadrado”, 4);
hash.Add(“Triangulo”, 3);
hash.Add(“Pentagono”,5)
new ArrayList(hash.Keys);
new ArrayList(hash.Values);
hash.Contains(“key”);
hash.ContainsValue(3);
hash.Remove(“key”);
foreach (DictionaryEntry item in hash)
```

Collections - SortedList

- Similar ao HashTable, porém, sempre ordenada pelo Key
- sortedList.GetKey(i)
- sortedList.GetByIndex(i)

Listas Com Tipo Definido (Generic)

- ArrayList => List<T>
- HashTable => Dictionary<T,T>
- SortedList => SortedList<T,T>

Queue, Stack

- Stack ou Stack<T> => Push e Pop – Last In First Out
- Queue ou Queue<T> => Enqueue e Dequeue – First In First Out

Exercício 5 – C#

- Crie uma lista de triângulos gerados com valores aleatórios. Teste o software utilizando List com tipo definido e ArrayList