



Classes de Coleções

? EXERCÍCIO 1: ORDENAÇÃO E INVERSÃO DE UM VETOR

```
using System;
namespace CC1
{public class MaisMetodosParaArrays
    {public static void ImprimirVectorOrd(int[] Numeros, string
ordem)
        {Console.WriteLine(ordem);
        foreach (int I in Numeros)
            {Console.Write("{0} ", I); }
        Console.WriteLine();}}

public class Ordenacoes
    {static void Main(string[] args)
        {int[] Numeros = { 1, 12, 5, 6, 7, 2, 3, 10, 9, 11 };
        Array.Sort(Numeros);
        MaisMetodosParaArrays.ImprimirVectorOrd(Numeros, "Números
por ordem crescente");
        Array.Reverse(Numeros);
        MaisMetodosParaArrays.ImprimirVectorOrd(Numeros, "Números
por ordem inversa");}}
```

? EXERCÍCIO 2: INDEXADOR DE LISTA

```
using System;
namespace CC2
{public class MaterialEscolar
    {private string NomeDonoDoMaterial;
    private string[] Material;
    private int Ultindice = 0;
    public MaterialEscolar(string N, params string[] Descricao)
    {NomeDonoDoMaterial = N;
    Material = new string[10];
    foreach (string M in Descricao)
        Material[Ultindice++] = M;}
    public string PNome
    {get
        {return NomeDonoDoMaterial; }}
    public int PIndice
    {get
        {return Ultindice; }}
```

```
public string this[int I]
{
    get
    {
        try
        {
            return Material[I];
        }
        catch
        {
            return "Erro! Índice menor que zero ou maior do que a capacidade máxima da lista";
        }
    }
    set
    {
        try
        {
            Material[I] = value;
            if (Ultindice < I + 1)
                Ultindice = I;
        }
        catch
        {
            Console.WriteLine("Erro! A capacidade da lista foi ultrapassada");
        }
    }
}

public class ListaDeMaterialEscolar
{
    static void Main(string[] args)
    {
        MaterialEscolar ME = new MaterialEscolar("Inês", "Caneta", "Lápis");
        ME[2] = "Caderno";
        ME[3] = "Borracha";
        ME[4] = "Afiador";
        ME[6] = "Livro";
        Console.WriteLine("Material para {0}", ME.PNome);
        for (int I = 0; I <= ME.PIndice; I++)
        {
            Console.WriteLine("{0}-{1}", I+1, ME[I]);
        }
        Console.WriteLine(ME[11]);
    }
}
```

? EXERCÍCIO 3: LISTA ORDENADA DE UTENTES

```
using System;
using System.Collections;
namespace CC3
{
    public class ListaOrdenada
    {
        public static void Main()
        {
            ArrayList Utentes = new ArrayList();
            Console.WriteLine("Nome do utente <ZZZ para terminar> ");
            string nome = Console.ReadLine();
            while (nome.ToUpper().CompareTo("ZZZ") != 0)
            {
                Utentes.Add(nome);
                Console.WriteLine("Nome do utente <ZZZ para terminar> ");
                nome = Console.ReadLine();
            }
            Utentes.Sort();
            Console.WriteLine("Lista de utentes:");
            foreach (string Ele in Utentes)
            {
                Console.WriteLine(Ele);
            }
        }
    }
}
```

```
Console.WriteLine("{0}", Ele);}}}
```

? EXERCÍCIO 4: SUBLISTA DE NOMES

```
using System;
using System.Collections;
namespace CC4
{public class SubListaDeNomes
{public static void Main()
{ArrayList Utentes = new ArrayList();
Console.Write("Nome do utente <ZZZ para terminar> ");
string nome = Console.ReadLine().ToUpper();
while (nome.CompareTo("ZZZ") != 0)
{Utentes.Add(nome);
Console.Write("Nome do utente <ZZZ para terminar> ");
nome = Console.ReadLine().ToUpper();}
Utentes.Sort();
int Inicio, Fim;
try
{Inicio = Utentes.IndexOf("BEATRIZ", 0);
Fim = Utentes.IndexOf("DÍLIA", 0);
Console.WriteLine("Sublista de nomes");
foreach (string Ele in Utentes.GetRange(Inicio, Fim - Inicio
+ 1))
Console.WriteLine(Ele);}
catch
{Console.WriteLine("Vazia ou não contém BEATRIZ ou DÍLIA");}}}}
```

? EXERCÍCIO 5: INSERÇÃO DE UMA LISTA NOOUTRA

```
using System;
using System.Collections;
namespace CC5
{public class InsercaoLista
{public static void Main()
{ArrayList Frutos = new ArrayList();
ArrayList FrutosTropicais = new ArrayList();
string[] F = {"Maçã", "Laranja", "Pêra", "Cereja", "Tangerina"};
string[] FT = {"Banana", "Papaia", "Anona", "Manga"};
foreach (string Ele in F)
Frutos.Add(Ele);
foreach (string Ele in FT)
FrutosTropicais.Add(Ele);
Frutos.InsertRange(Frutos.Count, FrutosTropicais);
```

```
Frutos.Sort();
foreach (string Ele in Frutos)
    Console.WriteLine("{0}", Ele);
Console.WriteLine("{0} espécies de frutos", Frutos.Count);}}}
```

? EXERCÍCIO 6: PESQUISA E REMOÇÃO DE UM ELEMENTO

```
using System;
using System.Collections;
namespace CC6
{
    public class PesqElimina
    {
        public static void Main()
        {
            string[] Localid = { "Porto", "Lisboa", "Braga", "Vila Real",
                                "Faro", "Coimbra" };
            ArrayList Localidade = new ArrayList();
            foreach (string Ele in Localid)
                Localidade.Add(Ele);
            Localidade.Sort();
            Console.Write("Digite o nome da localidade a eliminar ");
            string X = Console.ReadLine();
            if (Localidade.BinarySearch(X) >= 0)
                Localidade.Remove(X);
            else
                Console.WriteLine("{0} não existe no conjunto", X);
            foreach (string Ele in Localidade)
                Console.WriteLine(Ele);}}}
```

? EXERCÍCIO 7: LISTA DE OBJETOS DE UMA CLASSE

```
using System;
using System.Collections;
namespace CC7
{
    public class Utente
    {
        private string Nome;
        private int AnoNasc;
        public Utente()
        {
            Nome=" ";
            AnoNasc=0;
        }
        public Utente(string N, int A)
        {
            Nome=N;
            AnoNasc=A;
        }
        public string PNome
        {
            get
            {
                return Nome;
            }
        }
    }
}
```

```
        public int Idade(int AnoCorrente)
        {return AnoCorrente - AnoNasc;}}

public class ListaDeObjetos
{public static void Main()
{ArrayList SPA = new ArrayList();
  Utente UT=new Utente();
  Console.Write ("Nome do utente (ZZZ para terminar) ");
  int AnoUt=0;
  string NomeUt = Console.ReadLine();
  while (NomeUt.ToUpper().CompareTo("ZZZ") != 0)
  {Console.Write("Ano de nascimento do utente ");
    AnoUt = Convert.ToInt16(Console.ReadLine());
    UT = new Utente(NomeUt, AnoUt);
    SPA.Add(UT);
    Console.Write("Nome do utente (ZZZ para terminar) ");
    NomeUt = Console.ReadLine();}
  Console.WriteLine("{0}{1,15}", "Utente", "Idade");
  foreach (Utente U in SPA)
  {Console.WriteLine("{0} {1, 10}", U.PNome.PadRight(8),
    U.Idade(System.DateTime.Today.Year));
    Console.WriteLine("{0} utentes", SPA.Count);
  }}}}
```

? EXERCÍCIO 8: PERCURSO COM ITERADOR

```
using System;
using System.Collections;
namespace CC8
{public class Iterando
{public static void Main()
{ArrayList Frutos = new ArrayList();
  Frutos.Add("Laranjas");
  Frutos.Add("Papaia");
  Frutos.Add("Manga");
  Frutos.Add("Morangos");
  IEnumerator Item = Frutos.GetEnumerator();
  Item.MoveNext();
  for (int i=0; i<=Frutos.Count-1; i++)
  {Console.WriteLine(Item.Current.ToString());
    Item.MoveNext();}}}}
```

➡ EXECUÇÃO

Laranjas
Papaia



Manga
Morangos

? EXERCÍCIO 9: FILAS DE ESPERA

```
using System;
using System.Collections;
namespace CC9
{public class PessoasEmFila
    {public static void ColocarNaFila(Queue Q)
        {string Nome;
        Console.WriteLine("Digite o nome da pessoa (ZZZ para terminar)");
        Nome = Console.ReadLine();
        while ( Nome.ToUpper()!="ZZZ")
        {Q.Enqueue(Nome);
        Console.WriteLine("Digite o nome da pessoa (ZZZ para
        terminar)");
        Nome = Console.ReadLine();}}
    public static void ImprimirFila(Queue Q)
        {Console.WriteLine("Estado da fila");
        IEnumerator Ele = Q.GetEnumerator();
        while (Ele.MoveNext())
        {Console.WriteLine(Ele.Current);
        Console.WriteLine("Estão na fila {0} pessoas", Q.Count);}
    public static void Main(string[] args)
        {Queue Q = new Queue();
        ColocarNaFila(Q);
        ImprimirFila(Q);}}
```

? EXERCÍCIO 10: FILA DE ESPERA AO MINUTO X

```
using System;
using System.Collections;
namespace CC10
{public class RegistoFila
    {private string Nome;
    private int EntraServico;
    public RegistoFila(string N, int E)
        {Nome = N;
        EntraServico=E;}
    public string PNome
        {get
        {return Nome;}}
```

```
public int PEntraServico
{
    get
    {
        return EntraServico;
    }
}

public class FilaAoMinutoCerto
{
    static void Main(string[] args)
    {
        Queue Q = new Queue();
        string[] Nomes = { "Carla", "Rui", "Rosa", "Abel", "Maria" };
        int[] Atendimento = { 10, 5, 7, 6, 12 };
        ColocarNaFila(Q, Nomes, Atendimento);
        FilaAoMinutoX(Q);
    }

    public static void ColocarNaFila(Queue Q, string[] Nomes,
        int[] Atendimento)
    {
        int Entra=0;
        for (int I = 0; I <= Nomes.GetLength(0) - 1; I++)
        {
            RegistoFila EleFila = new RegistoFila(Nomes[I], Entra );
            Entra += Atendimento[I];
            Q.Enqueue(EleFila);
        }
    }

    public static void FilaAoMinutoX(Queue Q)
    {
        Console.WriteLine("Minuto a que pretende conhecer a fila ");
        int X = Convert.ToInt32(Console.ReadLine());
        RegistoFila EleFila;
        EleFila = (RegistoFila)Q.Peek();
        int Entra = EleFila.PEntraServico;
        while (X>=Entra)
        {
            Q.Dequeue();
            EleFila = (RegistoFila)Q.Peek();
            Entra= EleFila.PEntraServico;
        }

        Console.WriteLine("Estado da fila");
        IEnumerator ele = Q.GetEnumerator();
        while (ele.MoveNext())
        {
            EleFila = (RegistoFila)ele.Current;
            Console.WriteLine(EleFila.PNome + "--- Começa a ser
            atendido no minuto "+ EleFila.PEntraServico);
        }
        Console.WriteLine("Estão na fila {0} pessoas", Q.Count);
    }
}
```

? EXERCÍCIO 11: PILHA DE PROCESSOS

```
using System;
using System.Collections;
namespace CC11
{
    public class PilhaDeProcessos
    {
        static void Main(string[] args)
        {
            Stack Pilha = new Stack();
            Empilhar(Pilha);
        }
    }
}
```

```
OrdemResolucao(Pilha);}
public static void Empilhar(Stack Pilha)
{string Codigo;
 Console.Write("Código do processo(ZZZ para terminar) ");
 Codigo = Console.ReadLine();
 while (Codigo.ToUpper() != "ZZZ")
 {Pilha.Push(Codigo);
  Console.Write("Código do processo(ZZZ para terminar) ");
  Codigo = Console.ReadLine();}}
public static void OrdemResolucao(Stack Pilha)
{Console.WriteLine("Ordem de resolução dos processos:");
 while (Pilha.Count > 0)
 {Console.WriteLine(Pilha.Peek());
  Pilha.Pop();}}}
```

? EXERCÍCIO 12: PILHAS DE COMPONENTES

```
using System;
using System.Collections;
namespace CC12
{public class Componentes
 {private int Cod;
  private int TempoP;
  public Componentes(int C, int T)
  {Cod = C;
   TempoP = T;}
  public int Pcod
  {get
   {return Cod;}}
  public int PTempoP
  {get
   {return TempoP;}}}
public class PilhaDeComponentes
{static void Main(string[] args)
 {Stack S1 = new Stack();
  Stack S2 = new Stack();
  int[,]Subcomp={{1,10},{2,15},{3,5},{4,15},{5,10},{6,15},{7,30}
  ,{8,12}};
  EmpilharSingular(Subcomp, S1);
  EmpilharComposta(S1,S2);
  ImprimirComposta(S2);}
public static void EmpilharSingular (int[,] Subcomp, Stack S1)
 {for (int i = 0; i <= Subcomp.GetLength(0) - 1; i++)
  {Componentes C1 = new Componentes(Subcomp[i,0],Subcomp[i,1]);
   S1.Push(C1);}}
```



```
public static void EmpilharComposta (Stack S1, Stack S2)
{Componentes NovaComponente, C1, C2;
 while (S1.Count>0)
 {C1 = (Componentes)S1.Pop();
  try
  {C2 = (Componentes)S1.Pop();
   NovaComponente = new Componentes(C1.Pcod+C2.Pcod,C1.PTempoP
   +C2.PTempoP);}
  catch
  {NovaComponente = new Componentes(C1.Pcod , C1.PTempoP);}
  S2.Push(NovaComponente);}}
public static void ImprimirComposta (Stack S2)
{Componentes NovaComponente;
 IEnumerator ele = S2.GetEnumerator();
 while (ele.MoveNext())
 {NovaComponente = (Componentes)ele.Current;
  Console.WriteLine("{0} é processada em {1} segundos",
  NovaComponente.Pcod, NovaComponente.PTempoP );}}}
```

? EXERCÍCIO 13: DICIONÁRIO DE CAPITALS

```
using System;
using System.Collections;
namespace CC13
{public class CapitaisEuropeias
 {static void Main(string[] args)
  {Hashtable Ht = new Hashtable();
   InserirRegistos(Ht);
   ProcurarCapitais(Ht);}
 static void InserirRegistos(Hashtable Ht)
 {string Capital;
  Console.Write("Digite o país (ZZZ para terminar) ");
  string Pais = Console.ReadLine();
  while (Pais.ToUpper() != "ZZZ")
  {Console.Write("Digite a capital ");
   Capital = Console.ReadLine();
   Ht.Add(Pais, Capital);
   Console.Write("Digite o país (ZZZ para terminar) ");
   Pais = Console.ReadLine();}}
 static void ProcurarCapitais (Hashtable Ht)
 {Console.WriteLine("Capitais Europeias:");
  Console.Write("Digite o país cuja capital procura (ZZZ para
  terminar) ");
  string Aprocurar = Console.ReadLine();
  do
  {if (Ht[Aprocurar]!=null)
```



```

Console.WriteLine("{0} --- {1}", Aprocurar, Ht[Aprocurar]);
else
    Console.WriteLine("{0} não foi registado.", Aprocurar);
Console.Write("Digite o país cuja capital procura (ZZZ para
terminar) ");
Aprocurar = Console.ReadLine();}
while (Aprocurar.ToUpper() != "ZZZ");}}}
```

? EXERCÍCIO 14: ATUALIZAÇÃO DO DICIONÁRIO DE CAPITALS

```

using System;
using System.Collections;
namespace CC14
{public class AtualizacaoPaises
    {static void Main(string[] args)
        {Hashtable Ht = new Hashtable();
        InserirPaises(Ht);
        RemoverPaises(Ht);
        ImprimirPaises(Ht);}
    public static void InserirPaises(Hashtable Ht)
        {string Capital, Pais;
        Console.WriteLine("Inserção de países");
        Console.Write("País a inserir (ZZZ para terminar) ");
        Pais = Console.ReadLine();
        while (Pais.ToUpper() != "ZZZ")
            {try
                {Console.Write("Digite a capital ");
                Capital = Console.ReadLine();
                Ht.Add(Pais, Capital);}
            catch
                {Console.WriteLine("Chave repetida"); }
                Console.Write("País a inserir (ZZZ para terminar) ");
                Pais = Console.ReadLine();}}
    public static void RemoverPaises(Hashtable Ht)
        {string Pais;
        Console.WriteLine("Remoção de países");
        Console.Write("País a remover (ZZZ para terminar) ");
        Pais = Console.ReadLine();
        while (Pais.ToUpper() != "ZZZ" && Ht.Count > 0)
            {Ht.Remove(Pais);
            Console.Write("País a remover (ZZZ para terminar) ");
            Pais = Console.ReadLine();}}
    public static void ImprimirPaises(Hashtable Ht)
        {Console.WriteLine("Capitais registadas:");
        ICollection Chaves = Ht.Keys;
        foreach (string C in Chaves)
```

```
Console.WriteLine("{0} --- {1}", C, Ht[C]);}}}
```

? EXERCÍCIO 15: LISTA SEMPRE ORDENADA

```
using System;
using System.Collections;
namespace CC15
{public class Freguesia
    {private string Cod;
      private string Nome;
      public Freguesia()
      {Cod = "";
       Nome = "";}
      public Freguesia(string C, string Nf)
      {Cod = C;
       Nome = Nf;}
      public string PCod
      {get
       {return Cod;}}
      public string PNome
      {get
       {return Nome;}}}
public class ListaDeFreguesias
    {static void Main(string[] args)
      {SortedList Lord = new SortedList();
       InserirFreguesias(Lord);
       ImprimirLista(Lord);
       ProcurarCodigos(Lord);}
      static void InserirFreguesias(SortedList Lord)
      {string CFreg, DFreg;
       Freguesia F = new Freguesia();
       Console.Write("Código da freguesia (ZZZ para terminar) ");
       CFreg = Console.ReadLine();
       while (CFreg.ToUpper().CompareTo("ZZZ") != 0)
       {try
        {Console.Write("Nome da freguesia ");
         DFreg = Console.ReadLine();
         F = new Freguesia(CFreg, DFreg);
         Lord.Add(F.PCod, F.PNome);}
        catch
        {Console.WriteLine("Chave repetida. O último registo foi
         ignorado.");}
         Console.Write("Código da freguesia (ZZZ para terminar) ");
         CFreg = Console.ReadLine();}}
      static void ImprimirLista(SortedList Lord)
      {string CFreg, DFreg;
```



```
Console.WriteLine("{0}{1,10}", "Código", "Freguesia");
for (int I = 0; I < Lord.Count; I++)
{
    CFreg = (string)Lord.GetKey(I);
    DFreg = (string)Lord.GetByIndex(I);
    Console.WriteLine("{0} {1}", CFreg.PadRight(8), DFreg);
}
static void ProcurarCodigos(SortedList Lord)
{
    Console.Write("Código da freguesia a procurar (ZZZ para
terminar) ");
    string CFreg = Console.ReadLine();
    while (CFreg.ToUpper().CompareTo("ZZZ") != 0)
    {
        if (Lord.ContainsKey(CFreg) == true)
        {
            Console.WriteLine("Índice de {0} registo de {1}",
                Lord.IndexOfKey(CFreg), Lord[CFreg]);
        }
        else
        {
            Console.WriteLine("{0} não existe na lista", CFreg);
        }
        Console.Write("Código da freguesia a procurar (ZZZ para
terminar) ");
        CFreg = Console.ReadLine();
    }
}
```