BÀI 3:
LẬP TRÌNH C# CƠ BẢN (tt)

Thời gian: 180 phút

Giảng viên: PHẠM PHÚ KHƯƠNG

Email: phamphukhuong@dtu.edu.vn

Điện thoại:0905635421

DUYTAN edu.vn

NỘI DUNG

1. Mảng 1 chiều, đa chiều
2. Collections
3. Kiểu enumeration

Video

https://www.youtube.com/watch?v=RcFaJxtAWDI

MẢNG

- Chứa các biến có cùng kiểu dữ liệu.
- Truy xuất phần tử thông qua chỉ số (index)
- Chỉ số bắt đầu bằng 0.

```
Datatype[ ]   array-name
```

- VD
  - int[] myInteger = new int[5];
  - string[] myString = {"abc","def" };

MẢNG

- Lấy kích thước mảng qua thuộc tính Length
  - int Size = myArray.Length;
- Nếu thành phần của mảng là kiểu định trước, có thể dùng hàm Sort của lớp **Array** để sắp xếp
  - Array.Sort(myArray);
- Dùng hàm Reverse của Array để đảo thứ tự các phần tử trong mảng
  - Array.Reverse(myArray);

```
public static int Main()
{
    string[] artists = { "Leonardo", "Monet", "Van Gogh", "Klee" };

    Array.Sort(artists);
    foreach (string name in artists)
        Console.WriteLine(name);

    Array.Reverse(artists);
    foreach (string name in artists)
        Console.WriteLine(name);

    Console.ReadLine();
    return 0;
}
```

Dùng phương thức tĩnh Sort của lớp Array để sort artists

Dùng phương thức tĩnh Reverse của lớp Array để đảo thứ tự artists

Datatype[,] array-name

- Khai báo mảng int 2 dòng 3 cột
  - int[,] myMatrix = new int[2, 3];
- Có thể khởi gán
  - int[,] matrix1 = new int[,] {{1,2},{3,4},{5,6},{7,8}};
  - int[,] matrix2 = {{1,2},{3,4},{5,6},{7,8}};
  - string[,] beatleName = {{"Lennon","John"},
    {"McCartney","Paul"},
    {"Harrison","George"},
    {"Starkey","Richard"}};

MẢNG ĐA CHIỀU

```
static void Main(string[] args)
{
    double[,] matrix = new double[10, 10];
    int count = 0;
    for (int i = 0; i < matrix.GetLength(0); i++)
        for (int j = 0; j < matrix.GetLength(1); j++)
            matrix[i, j] = ++count;

    foreach (double d in matrix)
        Console.WriteLine(d);

    Console.ReadLine();
}
```

Truy cập tuần tự theo kiểu mảng 1 chiều

```
static void Main(string[] args)
{
    double[,] matrix = new double[10, 10];
    int count = 0;
    for (int i = 0; i < matrix.GetLength(0); i++)
        for (int j = 0; j < matrix.GetLength(1); j++)
            matrix[i, j] = ++count;

    for (int i = 0; i < matrix.GetLength(0); i++)
        for (int j = 0; j < matrix.GetLength(1); j++)
            Console.WriteLine(matrix[i, j]);

    Console.ReadLine();
}
```

Truy cập theo dạng dòng cột qua chỉ mục i và j

MẢNG Jagged

Datatype[ ][ ] array-name

- Jagged là mảng mà mỗi phần tử là một mảng có kích thước khác nhau
- Những mảng con này phải được khai báo riêng
- Khai báo mảng 3 dòng, mỗi dòng là một mảng 1 chiều

```
int[][] a = new int[3][];
    a[0] = new int[4];
    a[1] = new int[3];
    a[2] = new int[1];
```

Khai báo số dòng, hàng

Khai báo số cột riêng cho từng dòng

MẢNG Jagged

```
public static int Main()
{
    string[][] softwares = new string[3][];

    softwares[0] = new string[] {
        "Bitdefender", "Karperky", "NAV");
    softwares[1] = new string[] {
        "IE", "Mozilla", "Opera", "Avant");
    softwares[2] = new string[] {
        "MS Word", "OpenOffice");

    for (int i = 0; i < softwares.GetLength(0); i++)
        for (int j = 0; j < softwares[i].GetLength(0); j++)
           Console.WriteLine(softwares[i][j]);

    return 0;
}
```

Truy cập theo dòng, cột

```
public static int Main()
{
    string[][] softwares = new string[3][];

    softwares[0] = new string[] {
        "Bitdefender", "Karperky", "NAV");
    softwares[1] = new string[] {
        "IE", "Mozilla", "Opera", "Avant");
    softwares[2] = new string[] {
        "MS Word", "OpenOffice");

    foreach (string[] srr in softwares)
        foreach (string s in srr)
            Console.WriteLine(s);

    return 0;
}
```

Truy cập dùng foreach

DUYTAN edu.vn

TẬP HỢP

- C# hỗ trợ mạnh mẽ việc thao tác trên tập hợp
- Collection là enumerable data structures thông qua index hoặc key.
- Namespace
  - System.Array
  - System.Collections
- System.Collections cung cấp các lớp, phương thức, thuộc tính để tương tác với nhiều cấu trúc dữ liệu khác nhau.

TẬP HỢP

- Các giao diện được định nghĩa trong namespace

IEnumerable

IEnumerator **ICollection**

IList **IDictionary**

- Các lớp thực thi ICollection
  - System.Collections.Stack
  - System.Collections.Queue
  - System.Collections.BitArray
  - System.Collections.NameValueCollection

TẬP HỢP

- IDictionary thể hiện các collection theo dạng name-value.
- Các lớp thực thi giao diện này
  - System.Collections.SortedList
  - System.Collections.Hashtable
  - System.Collections.Specialized.HybridDictionary
  - System.Collections.Specialized.ListDictionary

TẬP HỢP

- IList thể hiện collection chỉ có giá trị.
- Những lớp thực thi giao diện này
  - System.Array
  - System.Collections.ArrayList
  - System.Collections.Specialized.StringCollection

- Giao diện IEnumerable
  - Cung cấp khả năng duyệt forward, read-only các item của đối tượng.
  - Phương thức chính GetEnumerator() trả về đối tượng thực thi giao diện IEnumerator.

```
string[] name = { "Ha Giang", "Ha Nam", "Ha Noi", "Ha
Tay" };
for(IEnumerator e = name.GetEnumerator(); e.MoveNext();
                Console.WriteLine(e.Current));

string[] name = { "Ha Giang", "Ha Nam", "Ha Noi", "Ha
Tay" };
foreach(string s in name)
     Console.WriteLine(s);
```

TẬP HỢP

- ArrayList: mảng động những đối tượng không cùng kiểu dữ liệu.
- Array chỉ chứa các đối tượng cùng kiểu dữ liệu
- ArrayList có khả năng chứa các đối tượng khác kiểu dữ liệu, nhưng được chứa dưới dạng kiểu Object
- VD: một đối tượng ArrayList có thể chứa các item kiểu string, int, long, float…
- ArrayList sử dụng indexer để xác định các item

TẬP HỢP

- Thuộc tính Count cho biết số lượng item được lưu trữ trong collection.
- Thuộc tính Capacity cho phép get/set số lượng item mà ArrayList có thể lưu trữ.
- Các item được thêm vào qua phương thức Add và xóa qua Remove

- Minh họa ArrayList

```
int i = 100;
float f = 99.99f;
ArrayList arraylist = new ArrayList();
arraylist.Add("Ha Nam");
arraylist.Add(i);
arraylist.Add(f);
for (int index = 0; index < arraylist.Count;
index++)
    Console.WriteLine(arraylist[index]);
```

TẬP HỢP

- StringCollection thực thi giao diện IList và tương tự như ArrayList các String

```
StringCollection strlist = new
StringCollection();
strlist.Add("Ha Giang");
strlist.Add("Ha Nam");
strlist.Add("Ha Noi");
foreach(string str in strlist)
      Console.WriteLine(str);
```

TẬP HỢP

- StringDictionary là dạng hashtable có khóa và dữ liệu cùng dạng string
- Hashtable chứa dữ liệu trong các khóa của nó

```
StringDictionary strList = new
StringDictionary();
strList.Add("A", "Ha Nam");
strList.Add("B", "Ha Giang");
strList.Add("C", "Ha Noi");

foreach(string s in strList.Values) // duyệt qua
từng giá trị
        Console.WriteLine(s);
```

TẬP HỢP

- Duyệt qua từng item trong StringDictionary
- Sử dụng kiểu DictionaryEntry

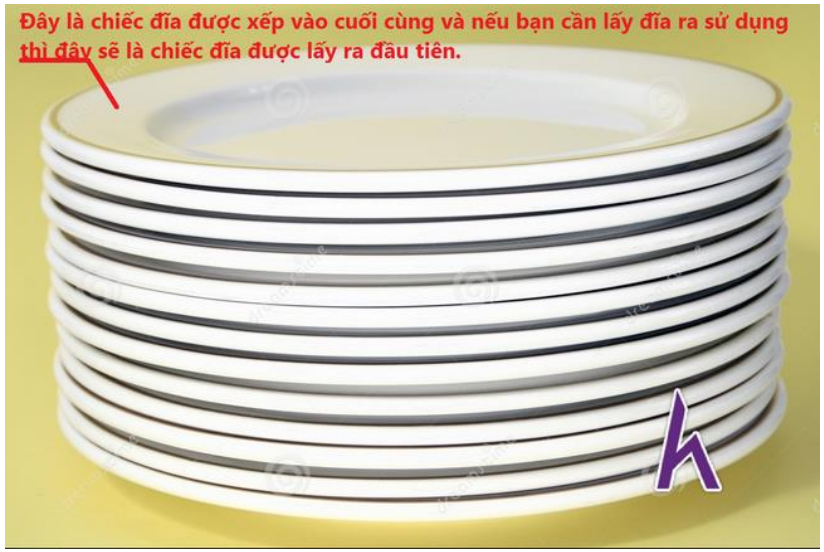
```
StringDictionary strList = new
StringDictionary();
strList.Add("A", "Ha Nam");
strList.Add("B", "Ha Giang");
strList.Add("C", "Ha Noi");

foreach(DictionaryEntry d in strList)
      Console.WriteLine("Key: {0} ==> Value:
{1}",d.Key, d.Value);
```

TẬP HỢP

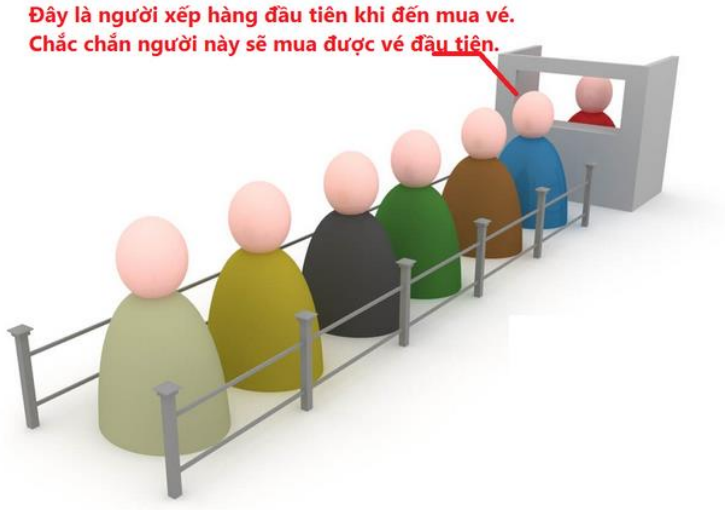
- Stack cung cấp danh sách dạng LIFO cho các item có kiểu Object.
- Các thao tác cơ bản Push và Pop

```
Stack objs = new Stack();
objs.Push("Ha Giang");
objs.Push("Ha Nam");
objs.Push("Ha Noi");
While (objs.Count > 0)
     Console.WriteLine(objs.Pop());
```



TẬP HỢP

- Queue: cung cấp danh sách dạng FIFO
- Item được thêm vào cuối danh sách và lấy ra ở đầu danh sách




```
Queue objs = new Queue();

objs.Enqueue("Ha Giang");
objs.Enqueue("Ha Nam");
objs.Enqueue("Ha Noi");

while (objs.Count > 0)
      Console.WriteLine(objs.Dequeue());
```

TẬP HỢP

- Hashtable: cung cấp cách thức nhanh chóng để lưu trữ và truy cập những item có kiểu object.
- Hỗ trợ tìm kiếm theo khóa

```
Hashtable hashtable = new Hashtable();
hashtable.Add(1, "Ha Nam");
hashtable.Add(2, "Ha Giang");
hashtable.Add(3, "Ha Noi");

foreach(int key in hashtable.Keys) //xuất các khóa
      Console.WriteLine(key);
```

- Sử dụng IDictionaryEnumerator

```
Hashtable hashtable = new Hashtable();
hashtable.Add(1, "Ha Nam");
hashtable.Add(2, "Ha Giang");
hashtable.Add(3, "Ha Noi");

IDictionaryEnumerator en =
hashtable.GetEnumerator();
while (en.MoveNext())
      Console.WriteLine("Key: {0} ==> Value:
{1}",
                        en.Key,en.Value);
```

TẬP HỢP

- SortedList: cho phép lưu trữ các item theo dạng khóa-giá trị, hỗ trợ sắp xếp các item

```
SortedList slist = new SortedList();
slist.Add(3, "Ha Giang");
slist.Add(1, "Ha Nam");
slist.Add(2, "Ha Noi");

foreach (string s in slist.Values) // duyệt qua các giá trị
        Console.WriteLine(s);
```

TẬP HỢP

- Hỗ trợ collection với kiểu dữ liệu bất kỳ (điểm mới từ .NET 2.0)
- Các tính năng nổi bật
  - Type safe
  - No boxing/unboxing
  - Richer functionality through System defined delegate types

Generic Collections

- List<T>

```
List<int> list = new List<int>();

list.Add(100); // thêm 1 phần tử
list.AddRange(new int[]{33,44,55}); // thêm nhiều phần tử

list.Insert(list.Count, 9999); // chèn vào cuối

foreach(int i in list)
     Console.WriteLine(i);
```

Phân biệt List<T> và ArrayList ?

Generic Collections

- Dictionary<T>

```
struct Car  {
        public string model;
        public Car(string m) {
            model = m;
        }
}
Dictionary<int, Car> showroom = new Dictionary<int, Car>();
showroom.Add(1, new Car("Camry"));
showroom.Add(200, new Car("Lexus"));
showroom.Add(3000,new Car("Accura"));

foreach(Car c in showroom.Values)
    Console.WriteLine(c.model);
```

Enumeration

- Dùng thay thế hằng
- Tập hợp các giá trị hằng được đặt tên
- Khai báo trực tiếp trong namespace
- Là kiểu dữ liêu

```
enum Color { Red, Green, Blue };
enum Access { personal = 1, group = 2, all = 4 };
```

Sử dụng

```
Color c = Color.Red;

Access a = Access.personal | Access.group;
If ((Access.personal & a) != 0)
   Console.WriteLine("access granted");
```

Enumeration

- Enumeration kế thừa từ object (Equals, ToString()).

Compare	if (c == Color.red) ... if (c > Color.red && c <= Color.green) ...
+, -	c = c + 2;
++, --	c++;
&	if ((c & Color.red) == 0) ...
|	c = c | Color.blue;
~	c = ~ Color.red;

Enumeration

```
public enum TimeOfDay
{
    Morning = 0,
    Afternoon = 1,
    Evening = 2
}
public static int Main()
{

    TimeOfDay day;
    day = TimeOfDay.Evening;
    WriteGreeting(day);
    return 0;
}
static void WriteGreeting(TimeOfDay timeOfDay)
{
    switch (timeOfDay)
    {
        case TimeOfDay.Morning:
            Console.WriteLine("Good morning!");
            break;
        case TimeOfDay.Afternoon:
            Console.WriteLine("Good afternoon!");
            break;
        case TimeOfDay.Evening:
            Console.WriteLine("Good evening!");
            break;
        default:
            Console.WriteLine("Hello!");
            break;
    }
}
```

Tóm tắt

- Cú pháp khá giống với C/C++
- Kiểu dữ liệu tham chiếu & giá trị
- Truyền tham số kiểu giá trị cho hàm
- Type-cast
- Boxing & Unboxing
- Điều khiển lặp foreach duyệt tập hợp
- Kiểu dữ liệu mảng
- Collections
- Enumeration

Bài tập

- Viết chương trình giải phương trình bậc 2

Bài tập

- Viết chương trình nhập vào một tháng, tính số ngày của tháng đó
  - Khi người dùng nhập vào số <1 hoặc >12, báo không tồn tại tháng.
  - Khi người dùng nhập vào tháng 1, 3, 5, 7, 8, 10, 12 thông báo "tháng X có 31 ngày".
  - Khi người dùng nhập vào tháng 4, 6, 9, 11 thông báo "tháng X có 30 ngày"
  - Khi người dùng nhập vào tháng 2, máy tính sẽ hỏi năm nào nếu năm nhuận thì có 29 ngày.
  - *Lưu ý:* Năm nhuận – năm không chia hết cho 100 nhưng lại chia hết cho 4, hoặc là năm chia hết cho 400 thì tháng 2 có 29 ngày. Những năm khác không nhuận, tháng 2 có 28 ngày.

Bài tập

- Viết chương trình Nhập họ tên, ngày sinh giới tính của một nhân viên
- Tính ngày về hưu của nhân viên đó biết:
  - Nếu giới tính là nam thì 60 tuổi
  - Nếu giới tính là nữ thì 55 tuổi
- Nếu ngày sinh chính là ngày hôm nay thì ghi ra là chúc mừng sinh nhật

Bài tập

- Viết chương trình Nhập một mảng các số nguyên
- Tính tổng các số nguyên của mảng đó
  - Số lớn nhất trong mảng đó là số nào
  - Số lượng và tổng các số nguyên dương của mảng đó

Bài tập

- Viết chương trình tạo một Dictionary<string, string> với khóa là mã môn học, giá trị value là tên của các môn học trong học kỳ 1 năm 2019-2020 của từng sinh viên.
- Viết lệnh nhập tất cả các môn học vào trong Dictionary
- Thực hiện kiểm tra xem môn "Kỹ thuật thương mại điện tử" đã tồn tại hay chưa, nếu chưa thì chèn môn "Kỹ thuật thương mại điện tử" vào với khóa là IS311.
- Đếm xem có bao nhiêu môn học
- Thực hiện xóa môn CS414 nếu có.

CÂU HỎI

1. Type Casting (Ép kiểu) là gì?
2. Biến local hay biến cục bộ là gì?
3. Theo quyền ưu tiên của các toán tử trong C#, toán tử nào có quyền ưu tiên cao nhất?
4. Mục đích của lớp System?
5. Nêu ý nghĩa cảu việc sử dụng lệnh Using trong chương trình của c#?

YOUTUBE

1. https://www.youtube.com/watch?v=pSiIHe2uZ2w
2. https://www.youtube.com/watch?v=qOruiBrXlAw&list=PL_c9BZzLwBRIXCJGLd4UzqH34uCclOFwC
3. https://www.youtube.com/watch?v=HB1aPYPPJ24