

Lab 08

Generics and Iterators

Mục tiêu

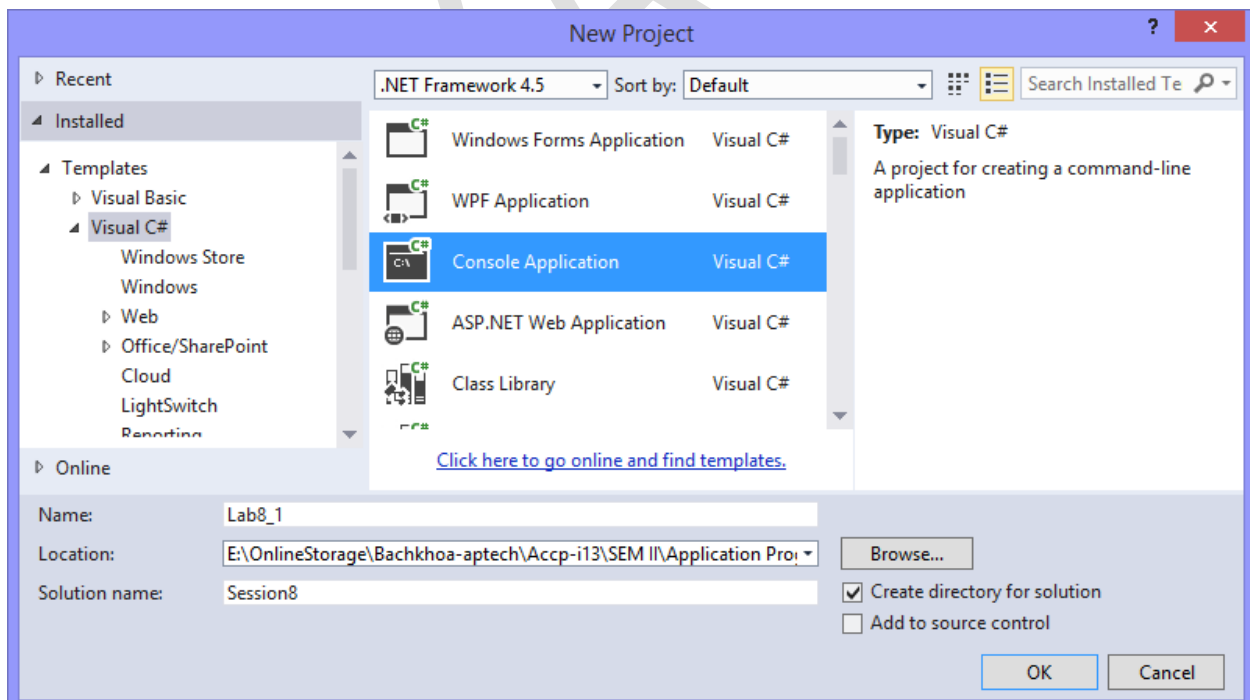
- Tạo và sử dụng Collection Generics
- Tạo và sử dụng Iterators

Phần I Bài tập step by step

Bài 8.1

Viết chương trình C# minh họa việc tạo một lớp Generic với các thao tác thêm, xóa, hiển thị phần tử trong tập hợp.

Bước 1: Mở Visual Studio 2013, vào menu File -> New -> Project -> chọn loại project “Console Application”, nhập tên project, tên solution -> OK.



Bước 2: Tạo lớp GenericList theo code gợi ý như sau:

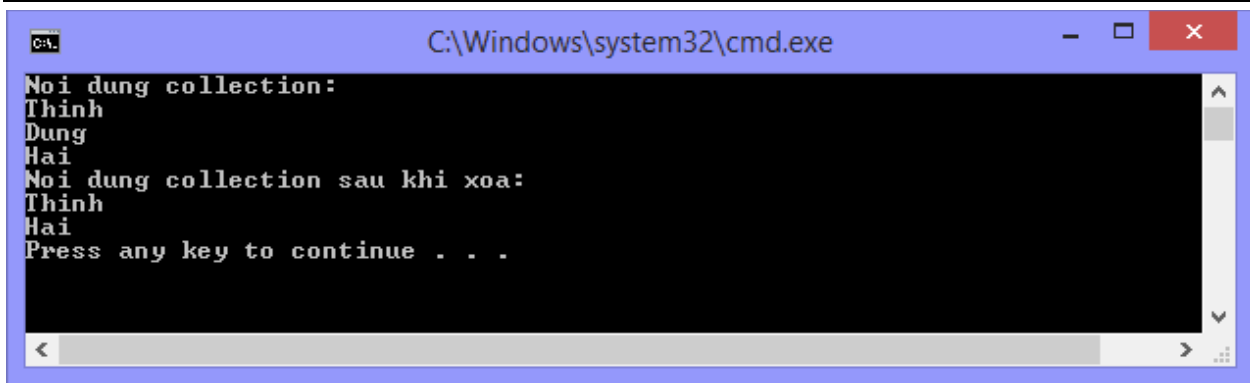
```
class GenericList<T>
{
    //Khai báo mảng
    T[] data;
    //biến vị trí
    int position;
    //Constructor khởi tạo số phần tử
    public GenericList(int n)
    {
        data = new T[n];
        position = 0;
    }
    //phương thức thêm 1 phần tử
    public void Add(T item)
    {
        if (position < data.Length)
        {
            data[position] = item;
            position++;
        }
    }
    //phương thức xóa 1 phần tử
    public void Remove(T item)
    {
        int index = Array.IndexOf<T>(data, item, 0);
        if(index < position && index >= 0)
        {
            for (int i = index; i < position-1; i++)
            {
                data[i] = data[i + 1];
            }
            position--;
        }
        else
        {
            Console.WriteLine("\'" + item + "' not found");
        }
    }
}
```

```
    }  
}  
//phương thức hiển thị các phần tử  
public void Display()  
{  
    for (int i = 0; i < position; i++)  
    {  
        Console.WriteLine(data[i]);  
    }  
}
```

Bước 3: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
static void Main(string[] args)  
{  
    //Khởi tạo collection 10 phần tử  
    GenericList<string> students = new GenericList<string>(10);  
    students.Add("Thịnh");  
    students.Add("Dung");  
    students.Add("Hai");  
    //hiển thị  
    Console.WriteLine("Nội dung collection:");  
    students.Display();  
    //xóa phần tử  
    students.Remove("Dung");  
    //hiển thị  
    Console.WriteLine("Nội dung collection sau khi xóa:");  
    students.Display();  
}
```

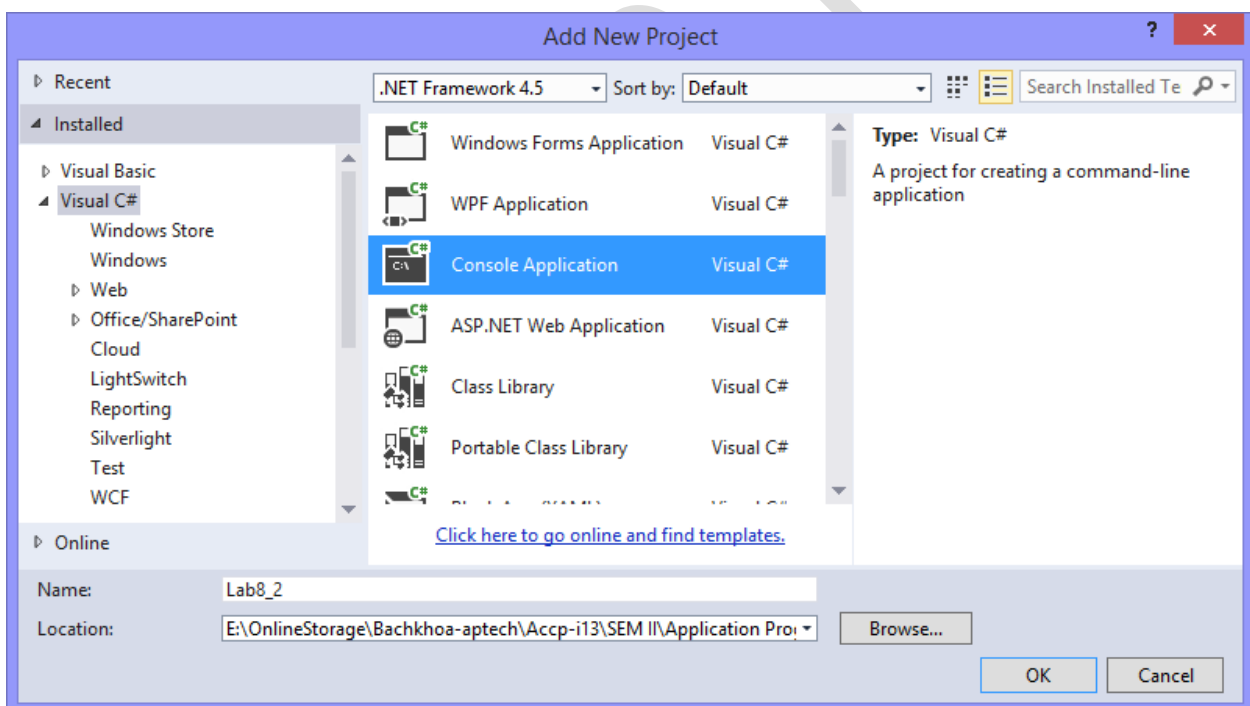
Bước 4: Nhấn Ctrl+F5 để chạy và xem kết quả



Bài 8.2

Viết chương trình C# minh họa việc tạo các phương thức để tìm max, min trong một tập hợp.

Bước 1: Kích chuột phải vào Solution “Session8” chọn Add -> New Project -> nhập tên.



Bước 2: Tạo lớp MathEx theo code gợi ý như sau:

```
//Định nghĩa lớp
public class MathEx
{
    //Phương thức Generic tìm phần tử lớn nhất trong một mảng
    public static T Max<T>(T first, params T[] values) where T : IComparable
```

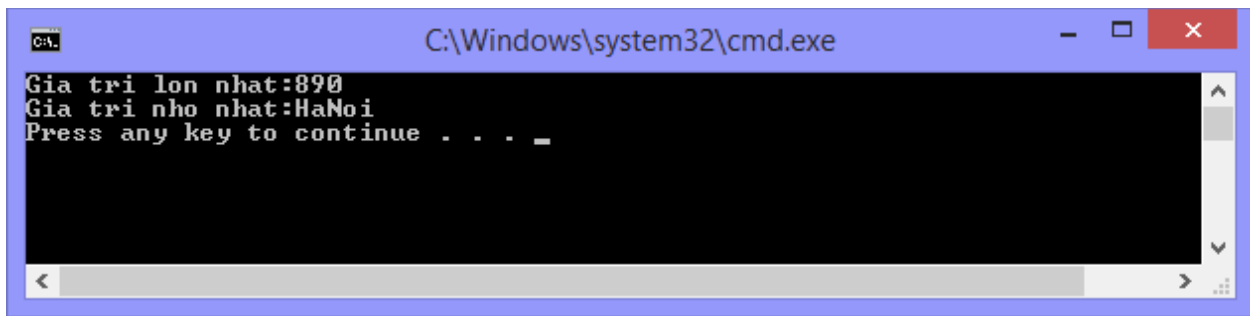
```
{
    T maximum = first;
    foreach (T item in values)
    {
        if (item.CompareTo(maximum) > 0)
        {
            maximum = item;
        }
    }
    return maximum;
}

//Phương thức Generic tìm phần tử nhỏ nhất trong một mảng
public static T Min<T>(T first, params T[] values) where T : IComparable
{
    T minimum = first;
    foreach (T item in values)
    {
        if (item.CompareTo(minimum) < 0)
        {
            minimum = item;
        }
    }
    return minimum;
}
}
```

Bước 3: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
static void Main(string[] args)
{
    //sử dụng các phương thức Generic
    Console.WriteLine("Gia tri lon nhat:" + MathEx.Max<int>(7, 490, 56, 890, 45));
    Console.WriteLine("Gia tri nho
    nhat:" + MathEx.Min<string>("NewYork", "LonDon", "HaNoi", "Paris"));
}
```

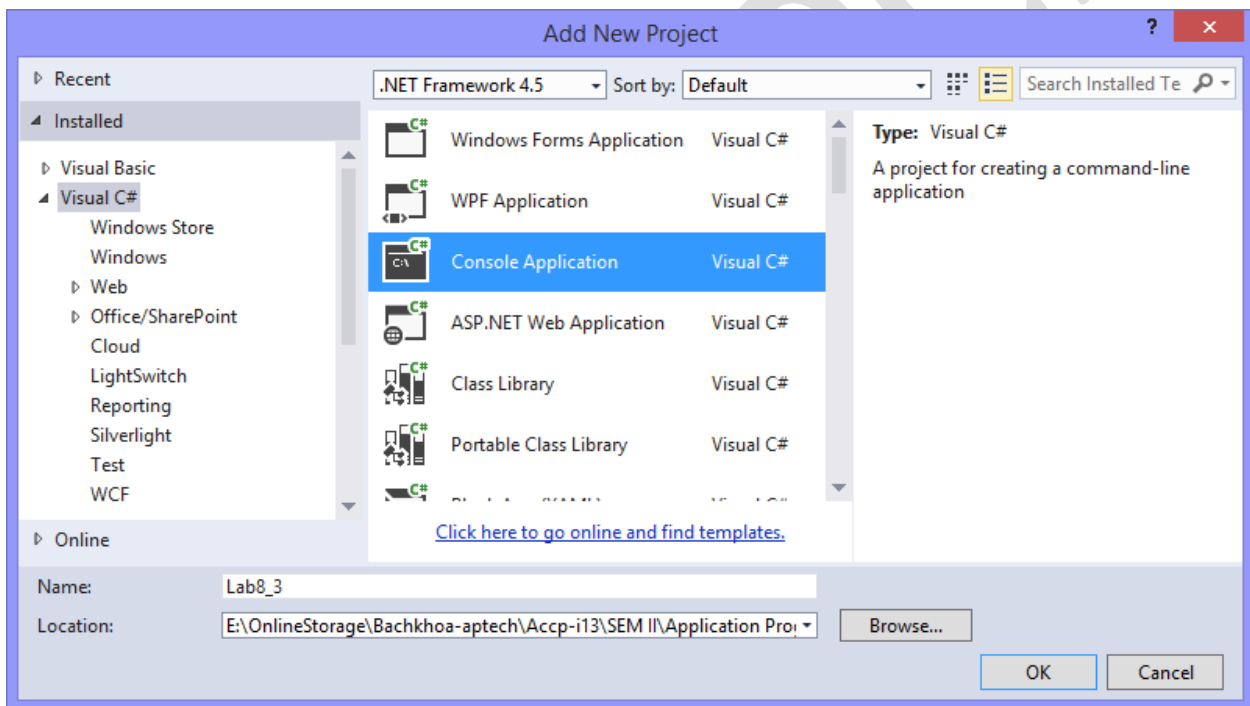
Bước 4: Nhấn Ctrl+F5 để chạy và xem kết quả



Bài 8.3

Viết chương trình C# minh họa việc tạo lớp Generic với các constraint.

Bước 1: Kích chuột phải vào Solution “Session8” chọn Add -> New Project ->nhập tên.



Bước 2: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
//định nghĩa giao diện IPerson
public interface IPerson
{
    string FullName { get; set; }
    DateTime DateofBirth { get; set; }
}
//Thực thi giao diện
```

```
public class PersonalIdentification : IPerson
{
    private string _name;
    private DateTime _dob;

    public PersonalIdentification(string name)
    {
        _name = name;
        _dob = new DateTime(0);
    }

    public virtual string FullName
    {
        get { return _name; }
        set { _name = value; }
    }

    public DateTime DateofBirth
    {
        get { return _dob; }
        set
        {
            _dob = value;
        }
    }
}

//định nghĩa lớp Generic với ràng buộc là PersonalIdentification
public class Employee<T> where T : PersonalIdentification
{
    private T info;

    public Employee()
    {
    }

    public Employee(T record)
```

```
{
    info = record;
}

public T Identification
{
    get
    {
        return info;
    }

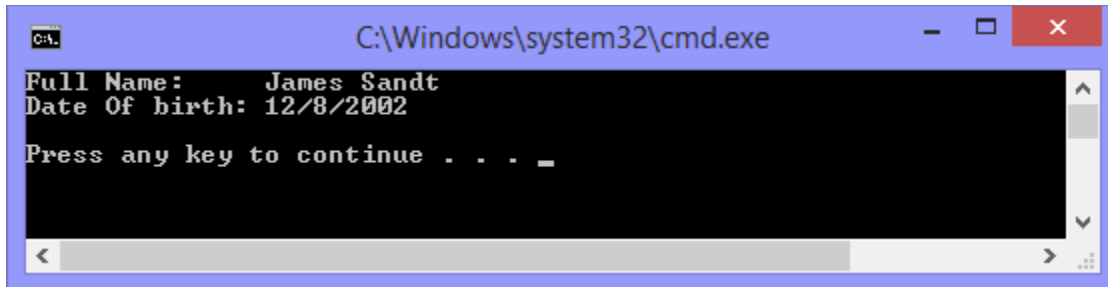
    set
    {
        info = value;
    }
}
}

class Program
{
    static void Main(string[] args)
    {
        //tạo đối tượng personalIdentification
        var std = new PersonalIdentification("James Sandt");
        std.DateOfBirth = new DateTime(2002, 12, 8);
        //tạo đối tượng employee
        Employee<PersonalIdentification> empl = new
        Employee<PersonalIdentification>();
        //gán thuộc tính Identification của Employee cho
        PersonalIdentification
        empl.Identification = std;
        //hiển thị thông tin
        Console.WriteLine("Full Name: {0}",
        empl.Identification.FullName);
        Console.WriteLine("Date Of birth: {0}",
        empl.Identification.DateOfBirth.ToShortDateString());
        Console.WriteLine();
    }
}
```



```
}  
}
```

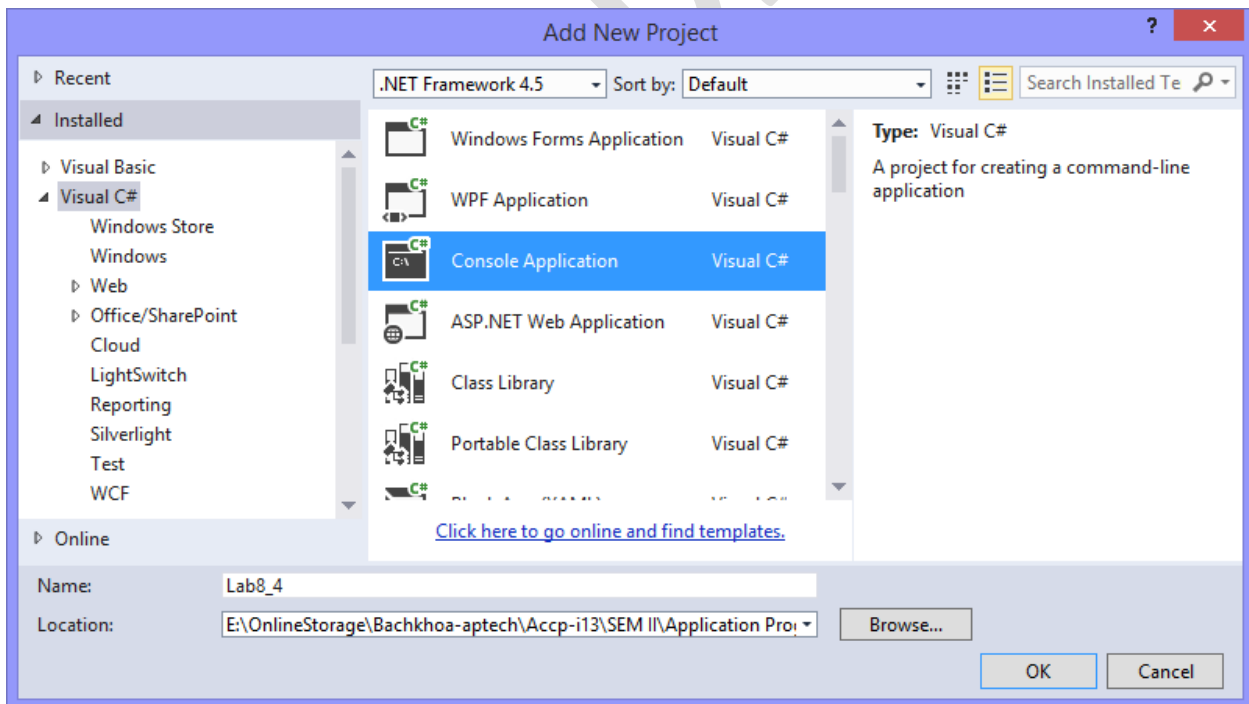
Bước 3: Nhấn Ctrl+F5 để chạy và xem kết quả



Bài 8.4

Viết chương trình C# minh họa việc tạo giao diện Generic và thực thi giao diện.

Bước 1: Kích chuột phải vào Solution “Session8” chọn Add -> New Project -> nhập tên.



Bước 2: Tạo giao diện ICalculator theo code gợi ý sau:

```
//Định nghĩa giao diện Generic  
interface ICaluclator<T>  
{
```

```
T Add(T a, T b);  
T Sub(T a, T b);  
T Div(T a, T b);  
T Mul(T a, T b);  
}
```

Bước 3: Tạo lớp CalculatorInt thực thi từ giao diện ICalculator theo gợi ý sau:

//Định nghĩa lớp thực thi giao diện từ Generic

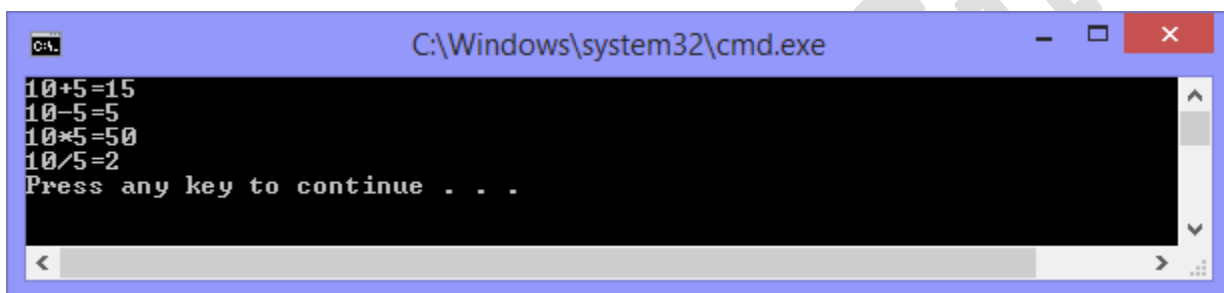
```
class CalculatorInt: ICaluclator<int>  
{  
    public int Add(int a, int b)  
    {  
        return a + b;  
    }  
  
    public int Sub(int a, int b)  
    {  
        return a - b;  
    }  
  
    public int Div(int a, int b)  
    {  
        return a / b;  
    }  
  
    public int Mul(int a, int b)  
    {  
        return a * b;  
    }  
}
```

Bước 4: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
static void Main(string[] args)  
{  
    //Tạo đối tượng calculator
```

```
CalculatorInt cal = new CalculatorInt();  
int a=10,b=5;  
//Test kết quả  
Console.WriteLine("{0}+{1}={2}", a, b, cal.Add(a, b));  
Console.WriteLine("{0}-{1}={2}", a, b, cal.Sub(a, b));  
Console.WriteLine("{0}*{1}={2}", a, b, cal.Mul(a, b));  
Console.WriteLine("{0}/{1}={2}", a, b, cal.Div(a, b));  
}
```

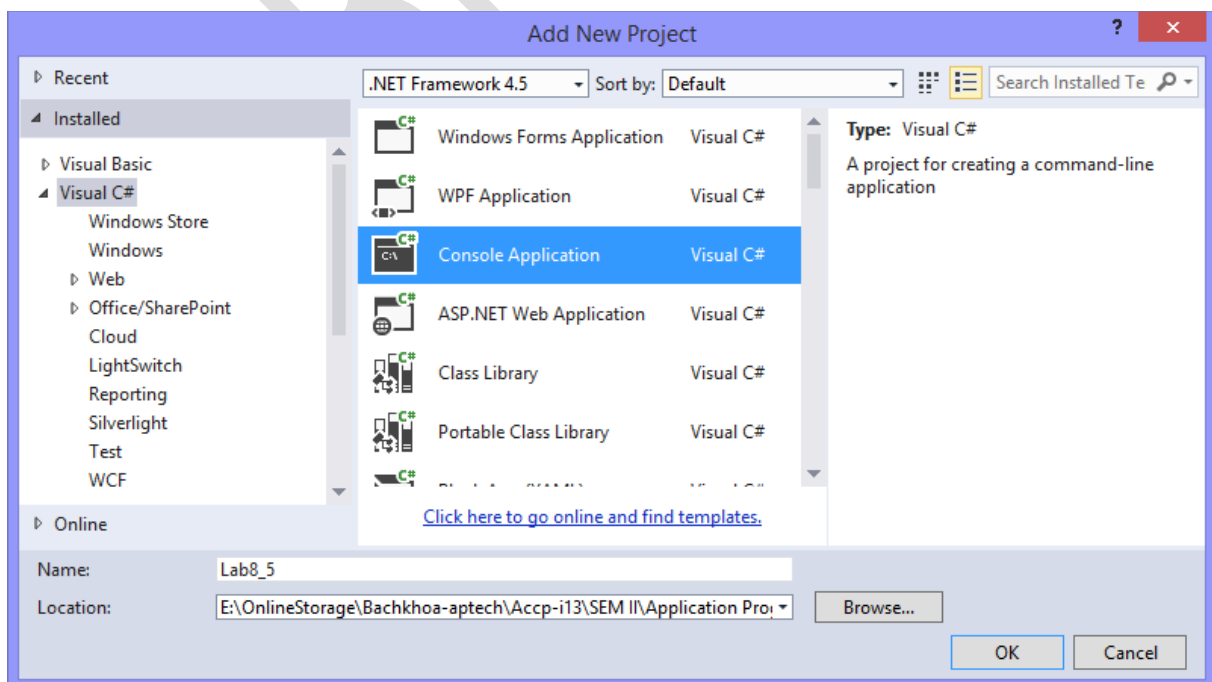
Bước 5: Nhấn Ctrl+F5 để chạy và xem kết quả



Bài 8.5

Viết chương trình C# minh họa việc tạo lớp và phương thức hỗ trợ Iterator.

Bước 1: Kích chuột phải vào Solution “Session8” chọn Add -> New Project ->nhập tên.



Bước 2: Tạo lớp Department theo code gợi ý sau:

```
class Department : IEnumerable
{
    //khai báo mảng dữ liệu
    string[] names = { "Finance", "Human Resource", "Information Technology",
        "Marketing" };
    //thực thi phương thức GetEnumerator của giao diện IEnumerable
    public IEnumerator GetEnumerator()
    {
        for (int i = 0; i < names.Length; i++)
        {
            yield return names[i];
        }
    }
}
```

Bước 3: Tạo lớp Flower theo code gợi ý sau:

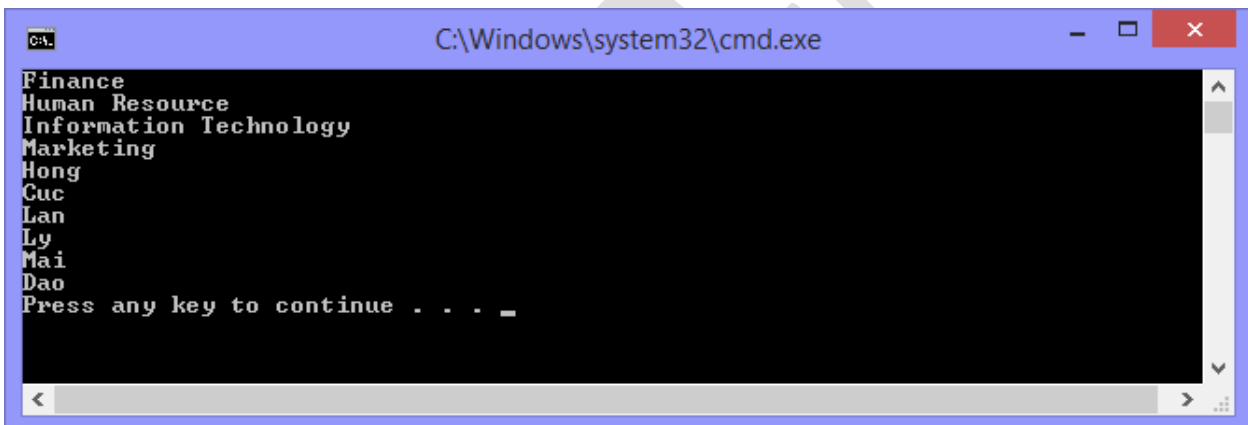
```
class Flower
{
    string[] names = { "Hong", "Cuc", "Lan", "Ly", "Mai", "Dao" };
    //tạo phương thức có kiểu trả về là IEnumerable
    public IEnumerable GetFlower()
    {
        for (int i = 0; i < names.Length; i++)
        {
            yield return names[i];
        }
    }
}
```

Bước 4: Mở tệp Program.cs và code cho hàm Main theo gợi ý sau:

```
static void Main(string[] args)
{
    //tạo đối tượng Department
    Department dep = new Department();
}
```

```
//sử dụng foreach truy xuất tập hợp
foreach (string item in dep)
{
    Console.WriteLine(item);
}
//tạo đối tượng Flower
Flower f = new Flower();
//dùng foreach duyệt qua tập hợp
foreach (string item in f.GetFlower())
{
    Console.WriteLine(item);
}
}
```

Bước 5: Nhấn Ctrl+F5 để chạy và xem kết quả



```
CA. C:\Windows\system32\cmd.exe
Finance
Human Resource
Information Technology
Marketing
Hong
Cuc
Lan
Ly
Mai
Dao
Press any key to continue . . . _
```

Phần II Bài tập tự làm

Bài 8.1: Viết chương trình C# thực hiện công việc sau:

- Tạo một lớp Generic một tả việc thêm và lấy dữ liệu từ một mảng
- Lớp cung cấp constructor cho phép chỉ ra số phần tử cần khởi tạo
- Test kết quả

Bài 8.2: Viết chương trình C# thực hiện công việc sau:

-
- Tạo phương thức Generic trong lớp Program, phương thức cho phép hoán vị 2 phần tử
 - Test kết quả

Bài 8.2: Viết chương trình C# và thực hiện công việc sau.

- Tạo lớp ProductEnumerable với biến thành viên là 1 mảng danh sách các sản phẩm.
- Hãy viết phương thức lấy dữ liệu từ mảng hỗ trợ vòng lặp foreach.

HẾT