LAB 02: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG (TIẾP)

A. MỤC TIÊU

- Hiểu và áp dụng thiết kế abstract class và interface.
- Phân biệt giữa abstract và interface.
- Ôn tập về mảng 2 chiều và xâu ký tự.

B. NỘI DUNG

- Class interface và abstract.
- Mảng 2 chiều.
- Xâu ký tự.

C. YÊU CẦU PHẦN CỨNG

- Máy tính cài đặt sẵn Visual Studio bản tối thiểu 2010.
- Máy tính có RAM tối thiểu 4GB và chip Core i3 trở lên để chạy mượt ứng dụng.

D. KẾT QUẢ SAU KHI HOÀN THÀNH

Hiểu rõ khái niệm abstract class và interface.

E. HƯỚNG DẪN CHI TIẾT

2.1. Interface

Interface (giao diện hoặc lớp giao tiếp) là 1 tập các thành phần chỉ có khai báo mà không có phần định nghĩa

Các thành phần này có thể là:

- Phương thức.
- Property
- Event
- Indexers

Một interface được hiểu như là 1 khuôn mẫu mà mọi lớp thực thi nó đều phải tuân theo. Interface sẽ định nghĩa phần "**làm gì**" (khai báo) và những lớp thực thi interface này sẽ định nghĩa phần "**làm như thế nào**" (định nghĩa nội dung) tương ứng.

Đặc điểm của interface

Chỉ chứa khai báo không chứa phần định nghĩa (giống phương thức thuần ảo). Mặc dù giống phương thức thuần ảo nhưng bạn không cần phải khai báo từ khoá abstract.

Việc ghi đè 1 thành phần trong interface cũng không cần từ khoá override.

Không thể khai báo phạm vi truy cập cho các thành phần bên trong interface. Các thành phần này sẽ mặc định là public.

Interface không chứa các thuộc tính (các biến) dù là hằng số hay biến tĩnh vẫn không được.

Interface không có constructor cũng không có destructor.

Các lớp có thể thực thi nhiều interface cùng lúc (ở 1 góc độ nào đó có thể nó là phương án thay thế đa kế thừa).

Một interface có thể kế thừa nhiều interface khác nhưng không thể kế thừa bất kỳ lớp nào.

Cú pháp:

```
interface <tên interface>
    {
        // Khai báo các thành phần bên trong interface
    }
```

Trong đó:

Interface là từ khoá dùng để khai báo 1 interface.

<tên interface> là tên do người dùng đặt và tuân theo các quy tắc đặt tên trong C#

Lưu ý là để tránh nhầm lẫn với lớp kế thừa thì khi đặt tên interface người ta thường thêm tiền tố "**I**" để nhận dạng.

Việc thực thi 1 interface hoàn toàn giống kế thừa từ 1 lớp

Bài 1: Xây dựng 2 lớp lần lượt có tên là **GiaoDichHangHoa**, **TestProgram** và 1 interface là: **GiaoDich**.

Hướng dẫn:

Xây dựng interface **GiaoDich:**

```
using System;
namespace Csharp
{
    public interface GiaoDich
    {
        // cac thanh vien cua interface
        //cac phuong thuc
        void hienThiThongTinGiaoDich();
        double laySoLuong();
    }
}
```

Xây dựng Lớp **GiaoDichHangHoa** kế thừa interface **GiaoDich**

```
using System;
namespace Csharp
{
    class GiaoDichHangHoa : GiaoDich
      {
```

```
        private string ma_hang_hoa;
        private string ngay;
        private double so_luong;
        public GiaoDichHangHoa()
        {
           ma_hang_hoa = " ";
           ngay = " ";
           so_luong = 0.0;
         }
    public GiaoDichHangHoa(string c, string d, double a)
        {
          ma_hang_hoa = c;
          ngay = d;
          so_luong = a;
        }
     public double laySoLuong()
        {
          return so_luong;
        }
     public void hienThiThongTinGiaoDich()
      {
         Console.WriteLine("Ma hang hoa: {0}", ma_hang_hoa);
         Console.WriteLine("Ngay giao dich: {0}", ngay);
         Console.WriteLine("So luong: {0}", laySoLuong());
      }
    }
}
```

Xây dựng lớp **TestProgram** chứa phương thức **main()** để thao tác trên đối tượng **GiaoDichHangHoa**

```
using System;
namespace Csharp
{
    public class TestProgram
    {
        static void Main(string[] args)
          {
            Console.WriteLine("Interface trong C#");
            Console.WriteLine("Vi du minh hoa interface");
            Console.WriteLine("--------------------------");
            //tao cac doi tuong GiaoDichHangHoa
              GiaoDichHangHoa t1 = new GiaoDichHangHoa("001",
             "8/10/2012", 78900.00);
             GiaoDichHangHoa t2 = new GiaoDichHangHoa("002",
             "9/10/2012", 451900.00);
            t1.hienThiThongTinGiaoDich();
            t2.hienThiThongTinGiaoDich();
            Console.ReadKey();
            }
        }
}
```

Bài 2: Xây dựng 1 interface **Nguoi**, class **SinhVien** và class **TestProgram**.

Hướng dẫn:

Xây dựng interface **Nguoi:**

```
interface Nguoi
{
     // phương thức hiển thị thông tin;
      void HienThi();
}
```

Xây dựng lớp **SinhVien:**

```
class SinhVien : Nguoi
  {
    private string ma;
    private string hoten;
    private string quequan;
    private int khoa;
    Encoding utf8 = Encoding.UTF8;
    public SinhVien()
    {
    }
public SinhVien(string ma, string hoten, string quequan, int
khoa)
    {
     // con trỏ this ở đấy là đại diện cho đối tượng cụ thể là
      SinhVien
      //Vì vậy this có thể truy xuất được các thuộc tính của đối
      tượng.
     ma = this.utf8.GetString(utf8.GetBytes(ma));
     hoten = this.utf8.GetString(utf8.GetBytes(hoten));
      quequan = this.utf8.GetString(utf8.GetBytes(quequan));
     this.ma = ma;
     this.hoten = hoten;
     this.quequan = quequan;
     this.khoa = khoa;
    }
    // getter setter là phương thức giúp chúng ta truy xuất vào
  các thuộc tính có đổ bảo mật cáo như private và protected
public string Ma
    {
```

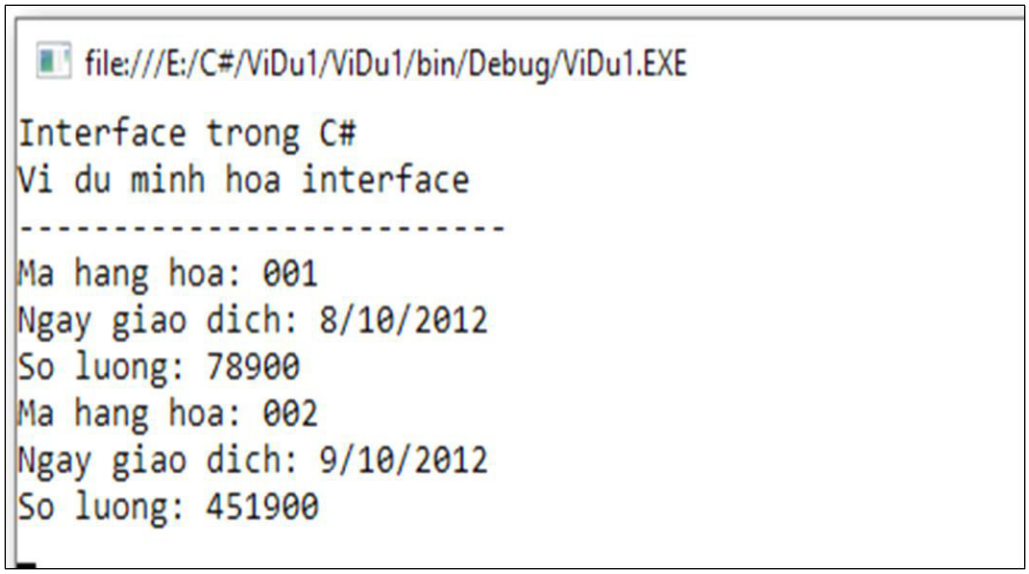
```
        get { return this.ma; }
        set { this.ma = value; }
    }
    public string HoTen
    {
        get { return this.hoten; }
        set { this.hoten = value; }
    }
    public string QueQuan
    {
        get { return this.quequan; }
        set { this.quequan = value; }
    }
    public int Khoa
    {
        get { return this.khoa; }
        set { this.khoa = value; }
    }
    public void TrangThai() {
        Console.Write("Trạng thái: ");
        if (this.khoa < 11)
          {
              Console.WriteLine("Đã Tốt Nghiệp");
          } else
          {
              Console.WriteLine("Đang Theo Học");
          }
    }
```

```
       public void HienThi()
        {
         Console.WriteLine("Mã Sinh Viên: "+this.ma);
         Console.WriteLine("Họ và Tên: "+this.hoten);
         Console.WriteLine("Quê Quán: "+ this.quequan);
       Console.WriteLine("Khóa:"+Convert.ToString(this.khoa));
         TrangThai();
        }
}
```

Xây dựng lớp **TestProgram:**

```
     class TestProgram
     {
        static void Main(string[] args)
        {
         Console.OutputEncoding = System.Text.Encoding.UTF8;
         SinhVien sv3 = new SinhVien();
         sv3.Ma = "115";
         sv3.HoTen = "Lê Thị Thanh Trà";
         sv3.QueQuan = "Hà Nam";
         sv3.Khoa = 8;
         sv3.HienThi();
         Console.WriteLine("--------------------------");
         Console.WriteLine("Sử dụng phương thức của interface cho ra
          kết quả");
         Nguoi ng = sv3;
         ng.HienThi();
         Console.ReadKey();
         }
    }
```

Kết quả chạy chương trình:



2.2. Abstract Class

- Lớp trừu tượng (Abstract Class) là lớp dùng để định nghĩa những thuộc tính và hành vi chung của những lớp khác..

Hay nói cách khác: Lớp trừu tượng là lớp dùng để khai báo thuộc tính và phương thức cho các lớp khác sử dụng.

(Lớp trừu tượng không cho phép khởi tạo tham số, chỉ khai báo mà thôi).

- Lớp trừu tượng được dùng như một lớp cha (base class) của các lớp có cùng bản chất.

Bản chất ở đây được hiểu là kiểu, loại, nhiệm vụ của class.

Mỗi lớp dẫn xuất (derived class - lớp con) có thể thừa kế từ một lớp trừu tượng.

Từ khóa abstract được dùng để định nghĩa một Lớp trừu tượng.

Những lớp được định nghĩa bằng cách dùng từ khóa abstract thì không cho phép khởi tạo đối tượng (instance) của lớp ấy.

Bài 1: Xây Dựng 1 abstract class **NhanVien**, class **GiamDoc**, class **KeToan**, class **TestProgram**.

Hướng dẫn:

Xây dựng abstract class **NhanVien:**

```
abstract class NhanVien
    {
        public string HoTen;
```

```
        public int Tuoi;
        public string GioiTinh;
        public abstract string ChucVu();
        public abstract void CongViec();
    }
```

Xây dựng lớp **GiamDoc**:

```
class GiamDoc : NhanVien
    {
        public override string ChucVu()
        {
            return "Giám Đốc";
        }
        public override void CongViec()
        {
         Console.WriteLine("Ban hành quyết định, chỉ ra đường lối
         chiến lược kinh doanh");
        }
    }
```

Xây dựng lớp **KeToan**:

```
class KeToan : NhanVien
    {
        public override string ChucVu()
        {
            return "Kế Toán";
        }
        public override void CongViec()
        {
            Console.WriteLine("Kiểm kê ngân sách, chi thu");
        }
    }
```

Xây dựng lớp **TestProgram**:

```
class TestProgram
     {
```

```
static void Main(string[] args)
    {
        // để chế độ output là UTF-8
        Console.OutputEncoding = System.Text.Encoding.UTF8;
        // chỉnh màu nền
        Console.BackgroundColor = ConsoleColor.White;
        //Xóa màu nên cũ
        Console.Clear();
        // chỉnh màu chữ
        Console.ForegroundColor = ConsoleColor.Black;
        GiamDoc objGiamDoc = new GiamDoc();
        objGiamDoc.HoTen = "Nguyễn Thị A";
        objGiamDoc.Tuoi = 30;
        objGiamDoc.GioiTinh = "Nữ";
        System.Console.WriteLine("Lop Dan xuat 'GiamDoc':
         {0},{1},{2},{3}",objGiamDoc.HoTen,objGiamDoc.Tuoi,
         objGiamDoc.GioiTinh, objGiamDoc.ChucVu());
        KeToan objKeToan = new KeToan();
        objKeToan.HoTen = "Nguyễn Thị B";
        objKeToan.Tuoi = 26;
        objKeToan.GioiTinh = "Nữ";
        System.Console.WriteLine("Lop Dan xuat 'Ke Toan': {0},
         {1}, {2}, {3}",objKeToan.HoTen, objKeToan.Tuoi,
         objKeToan.GioiTinh,objKeToan.ChucVu());
        Console.ReadKey();
        KeToan objKeToan1 = new KeToan();
        objKeToan1.HoTen = "Nguyễn Thị A";
        objKeToan1.Tuoi = 25;
        objKeToan1.GioiTinh = "Nữ";
```

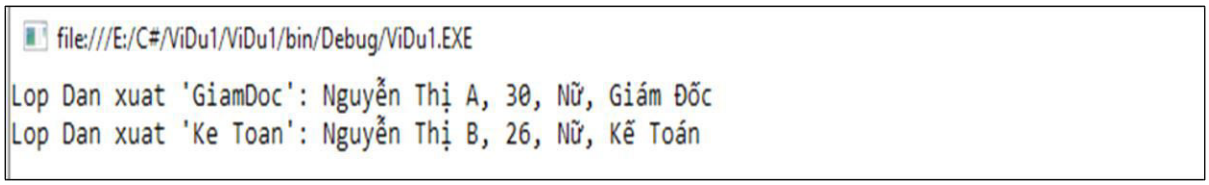
```
            System.Console.WriteLine("Lop Dan xuat 'Ke Toan': {0},
             {1}, {2}, {3}",objKeToan.HoTen, objKeToan.Tuoi,
             objKeToan.GioiTinh, objKeToan.ChucVu());

            Console.ReadKey();

        }

}
```

Kết quả chạy chương trình:



Bài 2: Tiếp tục với đề bài trên với abstract class kế thừa interface Nguoi.

Hướng dẫn:

Các class sẽ thay đổi như sau:

Xây dựng lớp **NhanVien**:

```
abstract class NhanVien: Nguoi

    {

        public string HoTen;

        public int Tuoi;

        public string GioiTinh;

        public string className;

        public abstract string ChucVu();

        public abstract void CongViec();

        public void HienThi()

        {

          System.Console.WriteLine("Lop Dan xuat '{4}': {0}, {1},
           {2},    {3}",this.HoTen,    this.Tuoi,    this.GioiTinh,
           this.ChucVu(), className);

        }

    }
```

Xây dựng lớp **GiamDoc**:

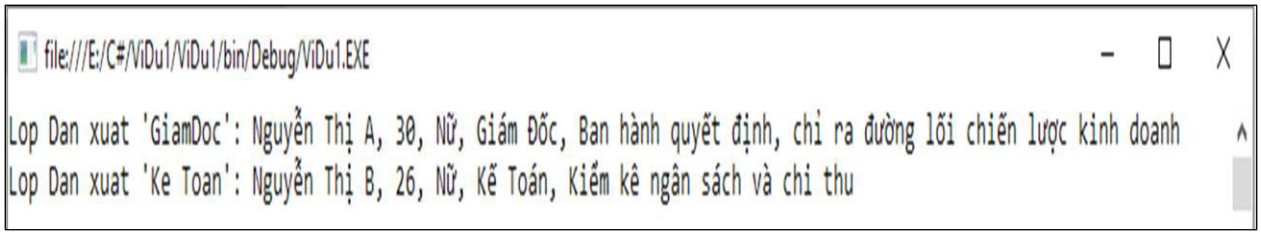
```
class GiamDoc : NhanVien
    {
        public GiamDoc()
        {
            this.className = this.GetType().Name;
        }
        public override string ChucVu()
        {
            return "Giám Đốc";
        }
        public override void CongViec()
        {
           Console.WriteLine("Ban hành quyết định, chỉ ra đường lối
           chiến lược kinh doanh");
        }
    }
```

Xây dựng lớp **KeToan**:

```
class KeToan : NhanVien
    {
        public KeToan()
        {
            this.className = this.GetType().Name;
        }
        public override string ChucVu()
        {
            return "Kế Toán";
        }
```

```
        public override void CongViec()
        {
            Console.WriteLine("Kiểm kê ngân sách, chi thu");
        }
    }
```

Kết quả chạy chương trình:



Bài 3: Viết chương trình thực hiện các công việc sau:

- Xây dựng lớp trìu tượng Shape có một phương thức trừu tượng draw()

- Xây dựng 2 lớp dẫn xuất của lớp Shape là Hình chữ nhật và Hình tròn

Trong đó lớp Hình chữ có một phương thức draw() cho phép in ra thông báo drawing rectangle..., lớp Hình tròn cũng có một phương thức draw() cho phép in ra thông báo drawing circle...

Viết một hàm main() để tạo ra 2 đối tượng của lớp Hình chữ nhật và Hình tròn sau đó thực thi 2 phương thức của 2 lớp trên.

Hướng dẫn:

Xây dựng lớp **Shape:**

```
public abstract class Shape
{
    public abstract void draw();
}
```

Xây dựng lớp **Rectangle:**

```
public class Rectangle : Shape
{
    public override void draw()
```

```
    {
        Console.WriteLine("drawing rectangle...");
    }
}
```

Xây dựng lớp **Circle:**

```
public class Circle : Shape
{
    public override void draw()
    {
        Console.WriteLine("drawing circle...");
    }
}
```

Xây dựng lớp **TestProgram:**

```
public class TestProgram
{
    public static void Main()
    {
        Shape s;
        s = new Rectangle();
        s.draw();
        s = new Circle();
        s.draw();
    }
}
```

Kết quả chạy chương trình:

```
drawing ractangle...
drawing circle...
```

2.3. Bài tập về mảng hai chiều

1. Khai báo

Kiểu_dữ_liệu [,] Tên_mảng;

hoặc

Kiểu_dữ_liệu [,] Tên_mảng = new Kiểu_dữ_liệu [kích_thước_hàng, kích_thước_cột];

2. Truy cập đến các phần tử của mảng hai chiều

Tên_mảng[chỉ_số_hàng, chỉ_số_cột];

Bài 1: Viết chương trình C# để nhập một mảng hai chiều có kích cỡ 3x3, sau đó in các phần tử mảng hai chiều này trên màn hình.

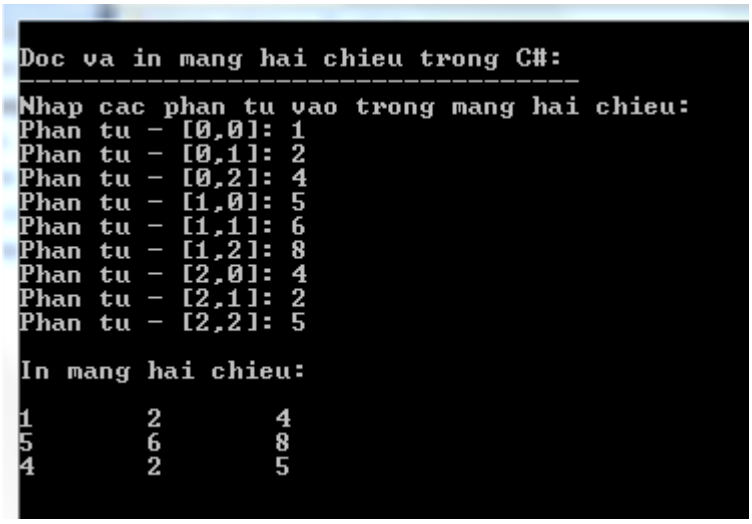
Hướng dẫn:

```
using System;
namespace Csharp
{
   class TestCsharp
   {
     public static void Main()
      {

       int i, j;
       int[,] arr1 = new int[3, 3];
     Console.Write("\nDoc va in mang hai chieu trong C#:\n");
       Console.Write("--------------------------------\n");
       /* nhap cac phan tu vao trong mang*/
     Console.Write("Nhap cac phan tu vao mang hai chieu:\n");
            for (i = 0; i < 3; i++)
            {
             for (j = 0; j < 3; j++)
             {
              Console.Write("Phan tu - [{0},{1}]: ",i,j);
            arr1[i, j] = Convert.ToInt32(Console.ReadLine());
             }
            }
            Console.Write("\nIn mang hai chieu: \n");
            for (i = 0; i < 3; i++)
            {
                Console.Write("\n");
                for (j = 0; j < 3; j++)
```

```
                    Console.Write("{0}\t", arr1[i, j]);
                }
                Console.Write("\n\n");
                 Console.ReadKey();
            }
        }
}
```

Kết quả chạy chương trình:



Bài 2: Viết chương trình C# để cộng hai ma trận và sau đó in ma trận kết quả ra màn hình.

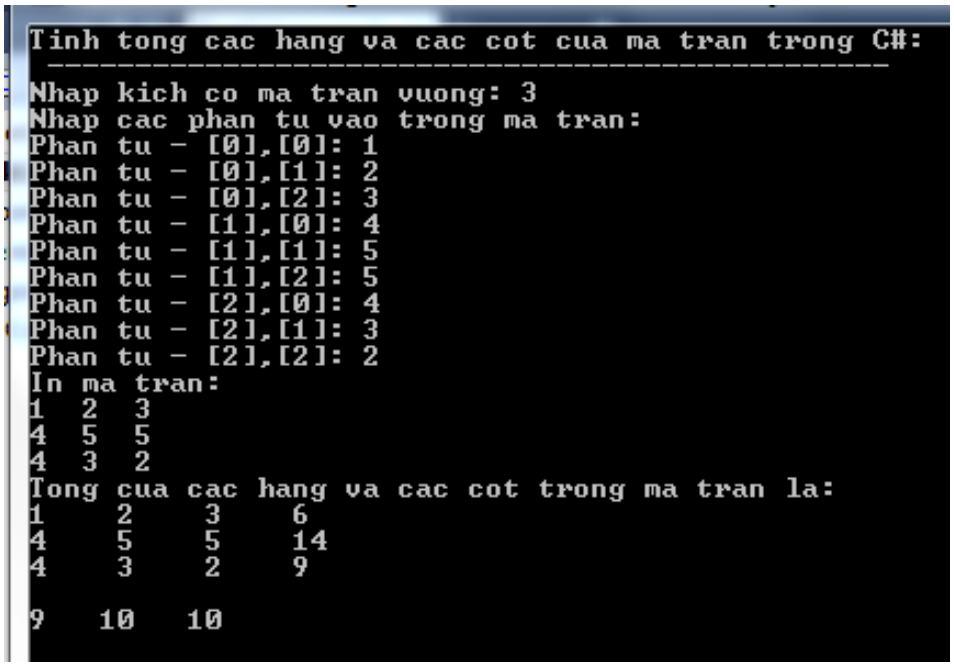
Hướng dẫn:

```
using System;
namespace Csharp
{
    class TestCsharp
    {
        public static void Main()
        {
            int i, j, n;
            int[,] arr1 = new int[50, 50];
            int[,] arr2 = new int[50, 50];
            int[,] ma_tran_tong = new int[50, 50];
            Console.Write("\nCong hai ma tran trong C#:\n");
            Console.Write("----------------------------\n");
            Console.Write("Nhap kich co cua hai ma tran vuong (nho
             hon 5): ");
```

```
n = Convert.ToInt32(Console.ReadLine());
/* Nhap cac phan tu vao trong mang da chieu*/
Console.Write("Nhap cac phan tu vao trong ma tran  dau
 tien:\n");
for (i = 0; i < n; i++)
{
 for (j = 0; j < n; j++)
 {
  Console.Write("Phan tu - [{0},{1}]: ", i, j);
  arr1[i, j]=Convert.ToInt32(Console.ReadLine());
 }
}
Console.Write("Nhap cac phan tu vao trong ma tran  thu
 hai:\n");
for (i = 0; i < n; i++)
{
 for (j = 0; j < n; j++)
 {
  Console.Write("Phan tu - [{0},{1}]: ", i, j);
  arr2[i, j]=Convert.ToInt32(Console.ReadLine());
 }
}
Console.Write("\nIn ma tran thu nhat:\n");
for (i = 0; i < n; i++)
{
    Console.Write("\n");
    for (j = 0; j < n; j++)
        Console.Write("{0}\t", arr1[i, j]);
}
Console.Write("\nIn ma tran thu hai:\n");
for (i = 0; i < n; i++)
{
    Console.Write("\n");
    for (j = 0; j < n; j++)
```

```
                    Console.Write("{0}\t", arr2[i, j]);
            }
            /* cong hai ma tran */
            for (i = 0; i < n; i++)
                for (j = 0; j < n; j++)
                ma_tran_tong[i, j] = arr1[i, j] + arr2[i, j];
            Console.Write("\nMa tran tong cua hai ma tran
             tren la: \n");
            for (i = 0; i < n; i++)
            {
                Console.Write("\n");
                for (j = 0; j < n; j++)
                  Console.Write("{0}\t", ma_tran_tong[i, j]);
            }
            Console.Write("\n\n");
            Console.ReadKey();
        }
    }
}
```

Kết quả chạy chương trình:



Bài 3: Viết chương trình tính tổng các phần tử trên đường chéo chính của ma trận

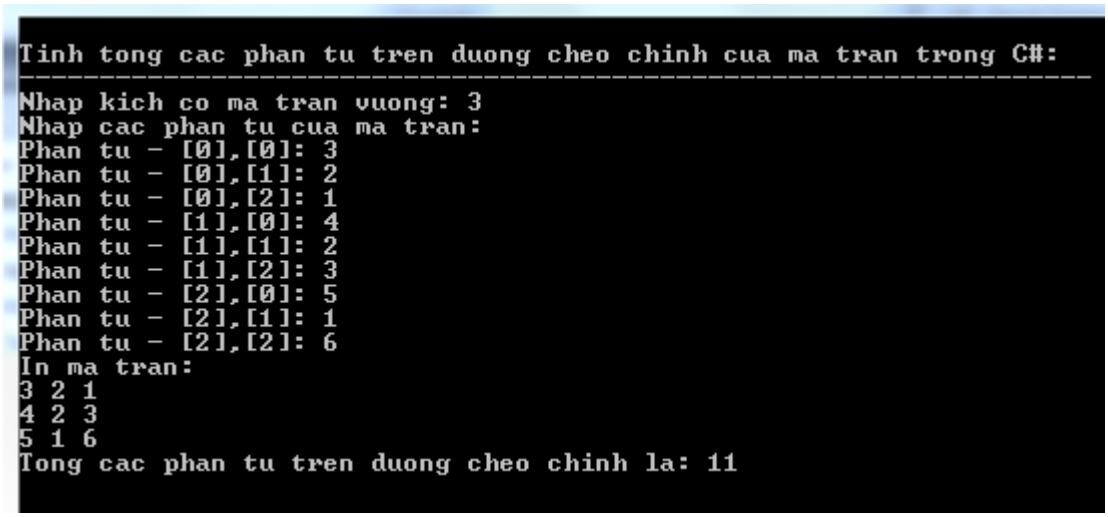
Hướng dẫn:

```
using System;
namespace Csharp
{
    class TestCsharp
    {
        public static void Main()
        {
            int i, j, sum = 0, n;
            int[,] arr1 = new int[50, 50];
            Console.Write("\nTinh tong cac phan tu tren duong
             cheo chinh cua ma tran trong C#:\n");
            Console.Write("-----------------------\n");
            Console.Write("Nhap kich co ma tran vuong: ");
            n = Convert.ToInt32(Console.ReadLine());
            Console.Write("Nhap cac phan tu cua ma tran:\n");
            for (i = 0; i < n; i++)
            {
            for (j = 0; j < n; j++)
            {
            Console.Write("Phan tu - [{0}],[{1}]: ", i, j);
            arr1[i, j] = Convert.ToInt32(Console.ReadLine());
              //tinh tong cac phan tu tren duong cheo chinh
            if (i == j) sum = sum + arr1[i, j];
            }
            }
            Console.Write("In ma tran:\n");
            for (i = 0; i < n; i++)
            {
                for (j = 0; j < n; j++)
                    Console.Write("{0} ", arr1[i, j]);
                Console.Write("\n");
```

```
            }

            Console.Write("Tong cac phan tu tren duong cheo
              chinh la: {0}\n", sum);
             Console.ReadKey();
        }
    }
}
```

Kết quả chạy chương trình:



2.4. Bài tập xâu ký tự

1. Khai báo

string Tên_xâu [= "gán một chuỗi"];

2. Truy cập vào các thành phần của xâu

Tên_xâu[chỉ_số];

Một số ký tự đặc biệt như: "\n", "\\","\t",...lần lượt đại diện cho ký tự xuống dòng, dấu "\" và dấu tab.

3. Phương thức Tostring()

Dùng để chuyển đổi một đối tượng bất kỳ sang kiểu chuỗi.

4. Các thao tác trên chuỗi

Lớp chuỗi cung cấp nhiều phương thức cho việc so sánh, tìm kiếm… được liệt kê trong bảng sau:

Bảng 2.1. Các phương thức của chuỗi

Thành viên	Mô tả
Empty	Biến thành viên tĩnh đại diện cho một chuỗi rỗng
Compare()	Phương thức tĩnh so sánh hai chuỗi
CompareOrdinal()	Phương thức tĩnh, so sánh 2 chuỗi không quan tâm đến ngôn ngữ
Concat()	Phương thức tĩnh, tạo một chuỗi mới từ nhiều chuỗi
Copy()	Phương thức tĩnh, tạo một bản sao
Equals()	Phương thức tĩnh, so sánh hai chuỗi có giống nhau
Format()	Phương thức tĩnh, định dạng chuỗi bằng các định dạng đặc tả
Intern()	Phương thức tĩnh, nhận về một tham chiếu đến chuỗi
IsInterned()	Phương thức tĩnh, nhận về một tham chiếu đến chuỗi đã tồn tại
Join()	Phương thức tĩnh, ghép nối nhiều chuỗi, mềm dẻo hơn Concat()
Chars	Indexer của chuỗi
Length	Chiều dài chuỗi (số ký tự)
Clone()	Trả về một chuỗi
CompareTo()	So sánh với chuỗi khác
CopyTo()	Sao chép một lượng ký tự trong chuỗi sang mảng ký tự
EndsWith()	Xác định chuỗi có kết thúc bằng chuỗi tham số không
Equals()	Xác định hai chuỗi có cùng giá trị

Insert()	Chèn một chuỗi khác vào chuỗi
LastIndexOf()	vị trí xuất hiện cuối cùng của một chuỗi con trong chuỗi
PadLeft()	Canh phải các ký tự trong chuỗi, chèn thêm các khoảng trắng bên trái khi cần
PadRight()	Canh trái các ký tự trong chuỗi, chèn thêm các khoảng trắng bên phải khi cần
Remove()	Xóa một số ký tự
Split()	Cắt một chuỗi thành nhiều chuỗi con
StartsWith()	Xác định chuỗi có bắt đầu bằng một chuỗi con tham số
SubString()	Lấy một chuỗi con
ToCharArray()	Sao chép các ký tự của chuỗi thành mảng các ký tự
ToLower()	Tạo bản sao chuỗi chữ thường
ToUpper()	Tạo bản sao chuỗi chữ hoa
Trim()	Cắt bỏ các khoảng trắng hai đầu chuỗi
TrimEnd()	Cắt bỏ khoảng trắng cuối chuỗi
TrimStart()	Cắt bỏ khoảng trắng đầu chuỗi

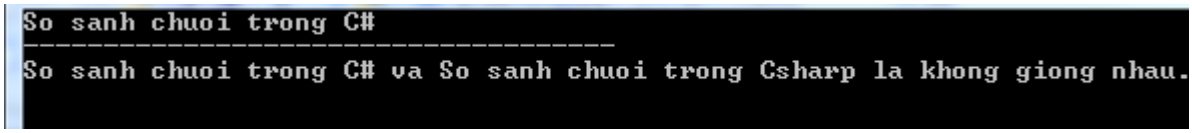
Bài 1: Viết chương trình nhập vào 2 chuỗi và kiểm tra xem 2 chuỗi vừa nhập vào có giống nhau không

Hướng dẫn:

```
using System;
namespace Csharp
{
    class TestCsharp
    {
        static void Main(string[] args)
```

```
        {
        Console.WriteLine("So sanh chuoi trong C#");
        Console.WriteLine("------------------------------");
        string str1 = "So sanh chuoi trong C#";
        string str2 = "So sanh chuoi trong Csharp";
        if (String.Compare(str1, str2) == 0)
        {
         Console.WriteLine(str1+"va"+str2 +"la giong nhau");
        }
        else
        {
            Console.WriteLine(str1+" va" str2+"la khong giong
             nhau.");
        }
        Console.ReadKey();
        }
    }
}
```

Kết quả chạy chương trình:



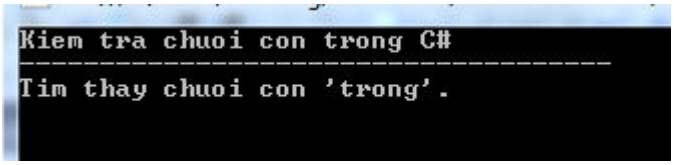
Bài 2: Viết chương trình kiểm tra xem một chuỗi có mặt trong một chuỗi khác hay không

Hướng dẫn:

```
using System;
namespace Csharp
{
    class TestCsharp
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Kiem tra chuoi con trong C#");
```

```
            Console.WriteLine("--------------------------");
            string str = "Chuoi con trong C#";
            if (str.Contains("trong"))
            {
            Console.WriteLine("Tim thay chuoi con 'trong'.");
            }
            Console.ReadKey();
        }
    }
}
```

Kết quả chạy chương trình:

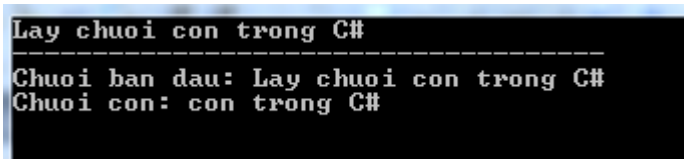


Bài 3: Viết chương trình trích ra một chuỗi con trong một chuỗi cho trước

Hướng dẫn:

```
using System;
namespace Csharp
{
    class TestCsharp
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Lay chuoi con trong C#");
            Console.WriteLine("----------------------");
            string str = "Lay chuoi con trong C#";
            Console.WriteLine("Chuoi ban dau: " +str);
            string substr = str.Substring(10);
            Console.WriteLine("Chuoi con: " +substr);
            Console.ReadKey();
        }
    }
}
```

Kết quả chạy chương trình:



2.5. Bài tập về nhà

Bài 1: Tạo lớp SinhVienUneti gồm hai thuộc tính họ tên và ngành cùng phương thức trừu tượng là getDiem(). Thêm phương thức getXepLoaiHL() để xếp loại học lực. Lớp cũng bao gồm một phương thức xuất để xuất họ tên, ngành, điểm và học lực ra màn hình.

Bài 2: Tạo *interface* Person có 2 phương thức sau:

- *public void* input();
- *public void* display();

Tạo một lớp Student thực thi giao diện trên và bổ sung thêm thuộc tính:

- *String* name;
- int *Age;*
- String *nativePlace;*
- String *id.*

Phương thức: tạo, set, get, display();

Viết chương trình chính thực hiện các công việc sau:

1. *Nhập vào một danh sách sinh viên*
2. *Hiển thị danh sách sinh viên vừa nhập ra màn hình*
3. *Tìm kiếm và đưa ra thông tin của sinh viên có tên là "Nam"*

Bài 3: Viết chương trìnhthực hiện các công việc sau:

1. Nhập một ma trận số thực gồm n hàng, m cột (với n và m là hai số nguyên dương).
2. Hiển thị ma trận.
3. Tìm số âm nhỏ nhất của ma trận.
4. Sắp xếp từng cột của ma trận theo thứ tự tăng dần.
5. Nhập số nguyên dương k, xóa cột thứ k của ma trận nếu có.
6. Tính trung bình cộng các phần tử có giá trị chẵn trong ma trận

Bài 4: Viết chương trìnhthực hiện các công việc sau:

1. Nhập một ma trận số thực vuông cấp n (với n là số nguyên dương).
2. Hiển thị ma trận.
3. Tính tổng các phần tử nằm trên đường chéo phụ của ma trận.
4. Tìm số âm lớn nhất trên đường chéo chính của ma trận

5. Đếm các phần tử của ma trận có giá trị chia hết cho 3 và 5.

Bài 5: Viết chương trình thực hiện các công việc sau:

1. Nhập vào một xâu ký tự sau đó hiểnt thị xâu ký tự vừa nhập ra màn hình
2. Đếm xem trong xâu có bao nhiêu chữ thường, bao nhiêu chữ hoa
3. Đếm số từ trong xâu vừa nhập
4. Đếm số phụ âm và nguyên âm trong xâu vừa nhập
5. Nhập vào một xâu con và đếm xem số lần xuất hiện của xâu con trong xâu đã nhập ở trên.