TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC
Khoa Công Nghệ Thông Tin

TÀI LIỆU GIẢNG DẠY

KỸ THUẬT LẬP TRÌNH 2

| 2019 – Lưu hành nội bộ |

LỜI TÁC GIẢ

Quyển giáo trình này được biên soạn dựa theo đề cương môn học "Kỹ thuật lập trình 2" của Khoa Công nghệ thông tin Trường Cao đẳng Công nghệ Thủ Đức.

Giáo trình biên soạn sẽ không tránh khỏi những sai sót về nội dung lẫn hình thức, nhóm biên soạn rất mong nhận được sự góp ý chân thành từ quý thầy cô và các em sinh viên để giáo trình hoàn thiện hơn.

MỤC LỤC

1.

MẢNG HAI CHIỀU

Chương này nhằm giới thiệu cho sinh viên các khái niệm về mảng hai chiều, các thao tác truy xuất trên mảng hai chiều, dùng mảng hai chiều làm tham số cho hàm. Qua đó sinh viên có khả năng sử dụng thành thạo kiểu dữ liệu mảng hai chiều để giải quyết các bài toán theo yêu cầu.

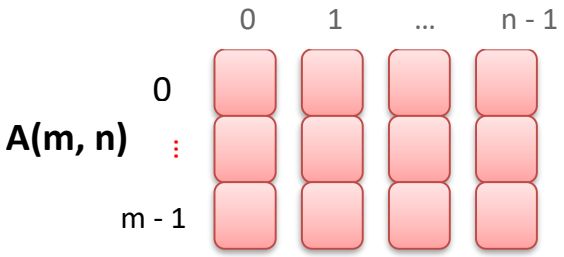
1.1| KHÁI NIỆM MẢNG NHIỀU CHIỀU

Mảng là một tập các dữ liệu có cùng kiểu. Các phần tử được đánh chỉ số (index) và được sắp xếp liền kề nhau thành một chuỗi. Kiểu của phần tử được gọi là kiểu cơ sở của mảng. Kiểu cơ sở có thể là các kiểu dữ liệu sẵn có của C# hoặc kiểu do người dùng tự định nghĩa. Chỉ số của mảng trong C# bắt đầu là 0.

Mảng trong C# có thể có nhiều hơn một chiều, gọi chung là mảng nhiều/đa chiều. Mảng trong C# có thể có nhiều hơn một chiều, gọi chung là mảng nhiều hay đa chiều. C# hỗ trợ mảng có tối đa 32 chiều.

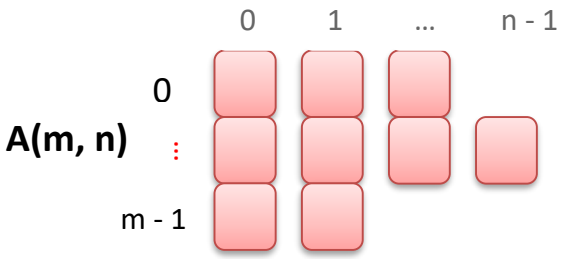
Mảng hai chiều được hình dung như một bảng (chữ nhật), trong đó mỗi ô là một phần tử. Mảng ba chiều được hình dung nó như một khối hộp lớn, gồm nhiều hộp nhỏ (giống như khối rubic). Mỗi hộp nhỏ là một phần tử.

Mảng hai chiều thường dùng để biểu diễn dữ liệu kiểu bảng, kiểu dữ liệu này rất thích hợp cho các bài bài liên quan đến đồ thị, biểu diễn ảnh, …



Mảng hai chiều (Ma trận) thực chất là mảng một chiều trong đó mỗi phần tử của mảng là một mảng một chiều và được truy xuất bởi hai chỉ số dòng và cột.

Mảng răng cưa (Jagged/ Zigzag Array) là một loại mảng đặc biệt trong C# và cũng thường được gọi là mảng của mảng. Có thể hình dung mảng răng cưa là một mảng một chiều, trong đó mỗi phần tử của nó lại là một mảng, thay vì là một dữ liệu cụ thể. Mỗi mảng phần tử có thể có kích thước khác nhau nhưng bắt buộc phải có chung kiểu cơ sở.



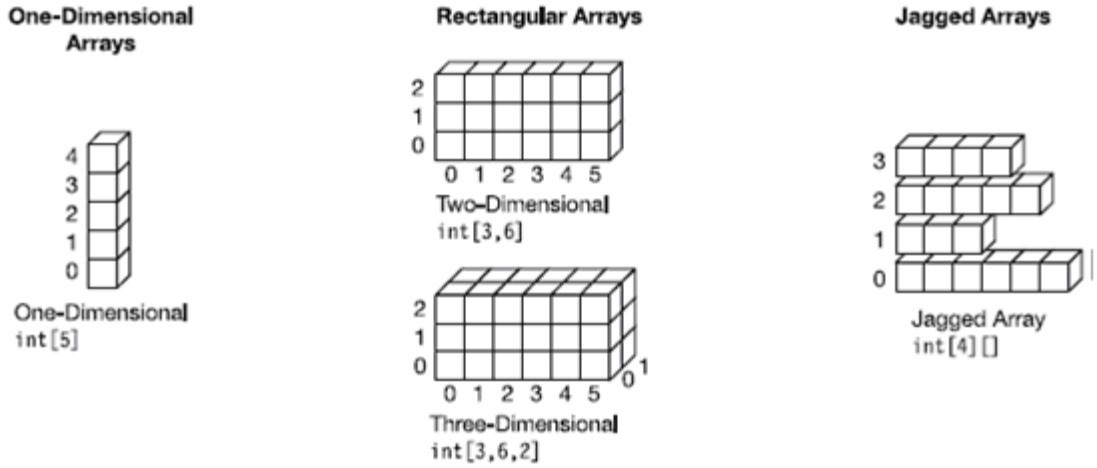


Figure 1: Sơ đồ hình khối của Mảng

1.2| KHAI BÁO, KHỞI TẠO

Cú pháp khai báo mảng hai chiều như sau:

Cách 1: khai báo mảng hai chiều rỗng

DataType[,] **ArrayName**;

Với DataType: là một kiểu dữ liệu bất kỳ

ArrayName: tên mảng được khai báo.

Ví dụ: int[,] ArrM1;

Cách 2: cấp phát bộ nhớ cho mảnghai chiều có r dòng và c cột, các phần tử mặc định có giá trị là 0.

DataType[,] **ArrayName** = **new DataType**[r, c];

Ví dụ: int[,] ArrM2 = new int[3,4];

Cách 3: cấp phát bộ nhớ cho mảng hai chiều có r dòng và c cột, khởi tạo giá trị cho các phần tử ngay khi khai báo

DataType[,] **ArrayName** = **new DataType**[r, c] {{value1, value2, …}, {valueN , ...}, ….};

Ví dụ: int[,] ArrM3 = new int[3,4] {{5, 2, 8 , 9},{1, 3, 0, 7 },{6, 8, 1, 0} };

5	**2**	**8**	**9**
1	**3**	**0**	**7**
6	**8**	**1**	**0**

Cách 4: Khi mảng có số phần tử trên mỗi dòng không bằng nhau (Jagged Array – mảng răng cưa) thì có thể khai báo như sau:

DataType[][] **ArrayName** = **new DataType**[numOfRow] [];

ArrayName[0] = **new DataType**[NumOfColumn1]{value1, …};

ArrayName[1]=**new DataType**[NumOfColumn2]{value1, value2, …};

…

Ví dụ:

```
// Khai báo mảng ZigZag có 2 dòng
int[][] arr = new int[2][]
// Khởi tạo giá trị cho các phần tử
arr[0] = new int[5] { 1, 3, 5, 7, 9 };
arr[1] = new int[4] { 2, 4, 6, 8 };
```

Ngoài ra, ta có thể vừa khai báo vừa khởi tạo mảng zigzag như sau:

```
int[][] arr1 = { new int[] { 1, 3, 5, 7, 9 },
                 new int[] { 2, 4, 6, 8 }
               };
```

1.3| TRUY XUẤT MẢNG HAI CHIỀU

1.4| ĐỌC METADATA CỦA MẢNG

Metadata là những thông tin về bản thân mảng, như số lượng phần tử, kiểu cơ sở, v.v.. Do mảng đều là các object thuộc kiểu System.Data, chúng ta có thể sử dụng các thuộc tính (và phương thức) của lớp này để đọc metadata của mảng.

Thuộc tính / Phương thức	Công dụng
Lengh / LongLength	Lấy thông tin số phần tử mảng
Rank	Lấy thông tin số chiều của mảng
GetType()	Trả về về kiểu của mảng.
GetLength(0)/GetLongLength(0) GetLength(1)/GetLongLength(1)	Trả về số phần tử trên **dòng** của mảng chữ nhật Trả về số phần tử trên **cột** của mảng chữ nhật
ToString()	Trả về về chuỗi thông tin của mảng.

Ví dụ:

```
class Program
{
    static void Main(string[] args)
    {
        int[,] ArrM = { { 5, 2, 8, 9 }, { 1, 3, 0, 7 }, { 6, 8, 1, 0 } };

        Console.WriteLine("So phan tu mang :" + ArrM.Length);
        Console.WriteLine("So chieu cua mang :" + ArrM.Rank);
        Console.WriteLine("Kieu mang  :" + ArrM.GetType() );
        Console.WriteLine("So cot :"+ ArrM.GetLength(0));
        Console.WriteLine("So dong :"+ ArrM.GetLength(1));
        Console.WriteLine("ToString :" + ArrM.ToString());

        Console.ReadKey();

    }
}
```

Kết quả:

```
So phan tu mang :12
So chieu cua mang :2
Kieu mang  :System.Int32[,]
So cot :3
So dong :4
ToString :System.Int32[,]
```

1.5| TRUY XUẤT TRỰC TIẾP ĐẾN PHẦN TỬ MẢNG

Mảng hai chiều thực chất là mảng một chiều trong đó mỗi phần tử của mảng là một mảng một chiều, và được truy xuất bởi hai chỉ số dòng và cột.

	Cột 0	Cột 1	Cột 2	Cột 3	…	Cột c-1
Dòng 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]		A[0][c-1]
Dòng 1	A[1][0]	A[1][1]	A[1][2]	A[1][3]		A[1][c-1]
Dòng 2	A[2][0]	A[2][1]	A[2][2]	A[2][3]	. . .	A[2][c-1]
…						
Dòng d-1	A[r-1][0]	A[r-1][1]	A[r-1][2]	A[d-1][3]		A[r-1][c-1]

Như vậy, Mảng A có r dòng và c cột với mỗi phần tử trong mảng được định danh bởi một tên phần tử trong mẫu A[i][j], với A là tên mảng , I là chỉ số dòng có các gí trị từ 0 đến d-1, j là chỉ số cột có các giá trị từ o đến c-1. Mỗi phần tử A[i][j] được xác định duy nhất trong mảng A.

Để truy xuất các phần tử tại dòng thứ I và cột thứ j của mảng A mảng ta dùng A[i, j].

Ví dụ: Giả sử ta có mảng

5	**2**	**8**	**9**
1	**3**	**0**	**7**
6	**8**	**1**	**0**

Câu lệnh truy xuất Console.WriteLine(ArrM3[1,3]) sẽ cho kết quả in số 7 ra màn hình

1.6| TRUY XUẤT TUẦN TỰ CÁC PHẦN TỬ MẢNG

Truy xuất tuần tự từng phần tử trong mảng theo các hướng duyệt mảng khác nhau như:

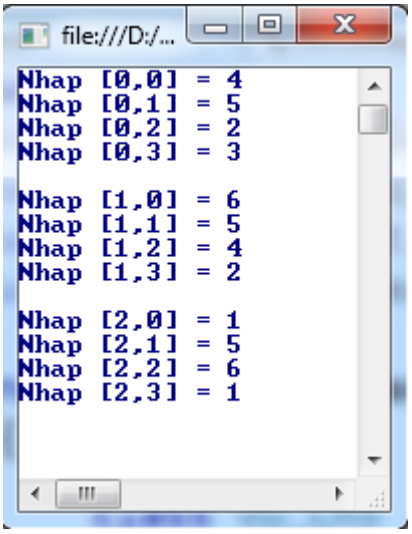
Truy xuất từng dòng theo từ trái sang phải và hướng từ trên xuống dưới.

Ví dụ1 : Cho mảng hai chiều gồm maxR dòng và maxC cột. Đoạn chương trình cho phép nhập từng phần tử của mảng như sau:

```
    static void Main(string[] args)
```

```
    {

        int maxR = 3;
        int maxC = 4;
        //Khai báo mảng hai chiều
        int[,] arr1 = new int[maxR, maxC];
        //Nhap gia trị cho mảng hai chiều
        Console.WriteLine("Nhapmang hai chieu");
        for (int i = 0; i < maxR; i++)
        {
            for (int j = 0; j < maxC; j++)
            {
                Console.Write("Nhap [{0},{1}] = ", i, j);
                int.TryParse(Console.ReadLine(), out arr1[i,

            }
            Console.WriteLine();
        }

        //In mang hai chieu chu nhat
        int dem = 0;
        foreach (int element in arr1)
        {

            Console.Write("{0}\t", element.ToString());
            if (++dem % maxR == 0)
                Console.WriteLine();

        }

        Console.ReadKey();

    }
}
```

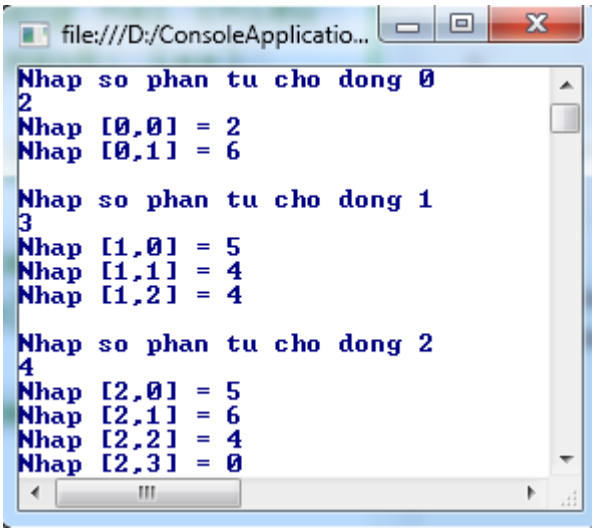
Kết quả:



Ví dụ 2 : Cho mảng hai chiều răng cưa gồm maxR dòng và số lượng cột trên mỗi dòng do người dùng nhập vào. Đoạn chương trình cho phép nhập từng phần tử của mảng răng cưa như sau:

```
class Ch1_VD2
{
   static void Main(string[] args)
   {
      int maxR = 3;
      int maxC = 0;
      //Mảng Zigzag
      int[][] arr2 = new int [maxR][];
     for (int i = 0; i < maxR; i++)
     {
        Console.WriteLine("Nhap so phan tu cho dong {0}", i);
        int.TryParse(Console.ReadLine(), out maxC);
        arr2[i] = new int[maxC];
        for (int j = 0; j < maxC; j++)
        {
          Console.Write("Nhap [{0},{1}] = ", i, j);
          int.TryParse(Console.ReadLine(), out arr2[i][j]);
        }
              Console.WriteLine();
      }
      Console.ReadKey();
  }
}
```

Kết quả:



Ví dụ 3: Dùng vòng lặp for để in mảng hai chiều

```
for (int i = 0; i < maxR; i++)
{
    for (int j = 0; j < maxC; j++)
    {
        Console.Write(arr1[i, j] + "\t");
    }
    Console.WriteLine();

}
```

Ví dụ 4: Dùng vòng lặp for để in mảng hai răng cưa

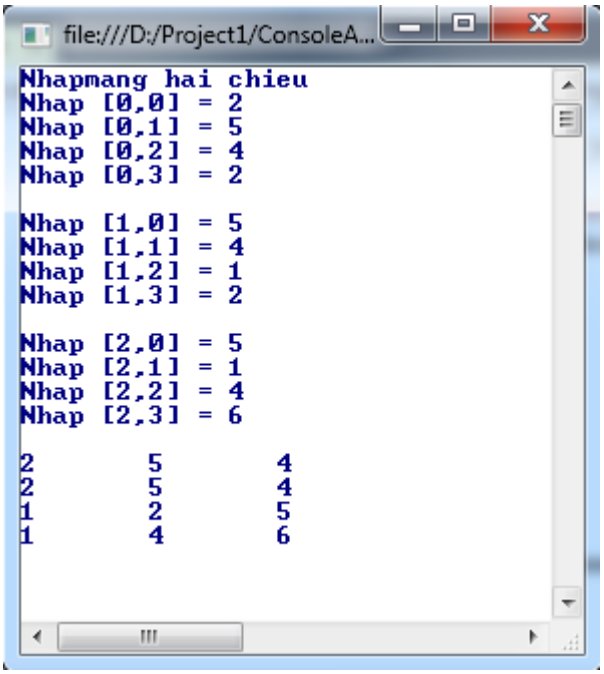
```
 for (int i = 0; i < maxR; i++)
 {
     for (int j = 0; j < arr2[i].Length; j++)
     {
         Console.Write(arr2[i][j] + "\t");
     }
     Console.WriteLine();

  }
```

Ví dụ 5: Sử dụng vòng lặp foreach để in mảng hai chiều chữ nhật

```
class Ch1_VD5
    {
        static void Main(string[] args)
        {

            int maxR = 3;
```

```
            int maxC = 4;
            //Khai báo mảng hai chiều
            int[,] arr1 = new int[maxR, maxC];
            //Nhap gia trị cho mảng hai chiều
            Console.WriteLine("Nhapmang hai chieu");
            for (int i = 0; i < maxR; i++)
            {
                for (int j = 0; j < maxC; j++)
                {
                    Console.Write("Nhap [{0},{1}] = ", i, j);
                    int.TryParse(Console.ReadLine(), out
                                arr1[i, j]);
                }
                Console.WriteLine();
            }

            //In mang hai chieu chu nhat
            int dem = 0;
            foreach (int element in arr1)
            {

                Console.Write("{0}\t", element.ToString());
                if (++dem % maxR == 0)
                    Console.WriteLine();

            }
            Console.ReadKey();


        }
```

Kết quả:

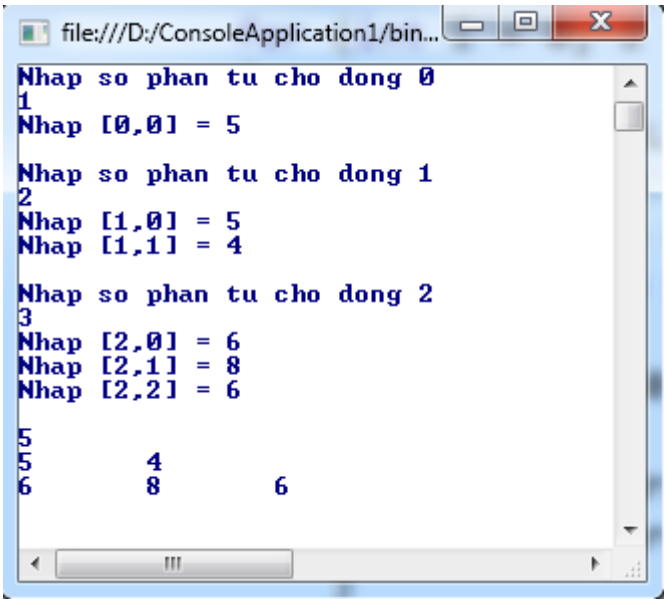


Ví dụ 6: Sử dụng vòng lặp foreach để in mảng hai chiều răng cưa

```
class VD_Ch5
    {
        static void Main(string[] args)
        {

            int maxR = 3;
            int maxC = 0;
            //Mảng Zigzag
            int[][] arr2 = new int[maxR][];
            for (int i = 0; i < maxR; i++)
            {
                Console.WriteLine("Nhap so pt dong {0}", i);
                int.TryParse(Console.ReadLine(), out maxC);
                //Cap phat bo nho cho dong
                arr2[i] = new int[maxC];
                //Nhap mang
                for (int j = 0; j < maxC; j++)
                {
                    Console.Write("Nhap [{0},{1}] = ", i, j);
                    int.TryParse(Console.ReadLine(), out
                                arr2[i][j]);
                }
                Console.WriteLine();
            }

            //In mang hai chieu răng cưa
            foreach (int[] row in arr2)
            {
```

```
                foreach (int ele in row)
                {
                    Console.Write("{0}\t", ele.ToString());
                }
                Console.WriteLine();
            }
            Console.ReadKey();

        }
```

Kết quả:



1.7| SỬ DỤNG MẢNG HAI CHIỀU LÀM ĐỐI SỐ CHO HÀM

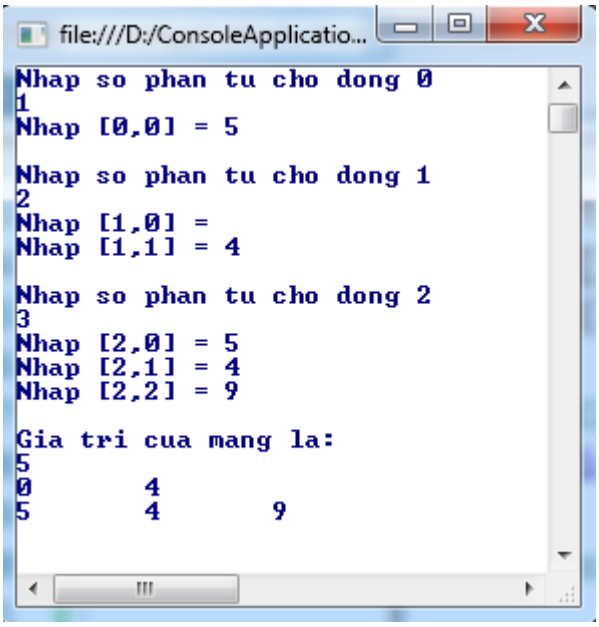
Mảng hai chiều có thể được dùng làm đối số của hàm. Khi truyền mảng vào hàm chính là truyền tham biến. Nghĩa là các thay đổi giá trị các phần tử mảng sẽ được lưu lại trên bộ nhớ khi hàm thực hiện xong.

Ví dụ 5: Dùng mảng răng cưa làm tham số cho hàm thực hiện nhập mảng và hàm xuất mảng.

```
class Ch1_VD5
    {
        static void Main(string[] args)
        {
            int maxR = 3;

            //Mảng Zigzag
            int[][] arr2 = new int[maxR][];
            NhapMang2DJ(arr2, maxR);
            //In mảng răng cưa
            Console.WriteLine("Gia tri cua mang la:");
```

```
            XuatMang2DJ(arr2, maxR);

            Console.ReadKey();

        }

        //hàm nhập mảng
        static void NhapMang2DJ(int[][] arr2, int maxR)
        {
            int maxC = 0;
            for (int i = 0; i < maxR; i++)
            {
               Console.WriteLine("Nhap so phan tu cho dong {0}", i);
                int.TryParse(Console.ReadLine(), out maxC);
                arr2[i] = new int[maxC];
                for (int j = 0; j < maxC; j++)
                {
                    Console.Write("Nhap [{0},{1}] = ", i, j);
                    int.TryParse(Console.ReadLine(), out arr2[i][j]);
                }
                Console.WriteLine();
            }

        }

        //hàm xuất mảng răng cưa
        static void XuatMang2DJ(int[][] arr2, int maxR)
        {

            for (int i = 0; i < maxR; i++)
            {

                for (int j = 0; j < arr2[i].Length; j++)
                {
                    Console.Write(arr2[i][j] + "\t");
                }
                Console.WriteLine();

            }
        }
    }
```

Kết quả:



1.8| MỘT SỐ ỨNG DỤNG MẢNG HAI CHIỀU

Ứng dụng 1:

Trong một cuộc chơi vượt chướng ngại vật để nhặt bóng gồm có 3 người, mỗi người chơi được thực hiện 4 lần, mỗi lần người chơi phải nhặt bóng bỏ vào rổ trong khoảng thời gian 60 giây. Kết qủa sau mỗi lượt nhặt bóng của người chơi được ghi lại và hiển thị trên hệ thống. Sau khi kết thúc lượt chơi, bảng điện tử thể hiện số bóng của mỗi người chơi theo thứ tự tăng dần.

Như vậy kết quả sẽ được ghi vào bảng gồm có 4 dòng , 5 cột. Số dòng thể hiện số người chơi. Số cột thể hiện số lượt người chơi lặp lại. Bảng kết quả thể hiện được sắp xếp theo thứ tự tăng dần trên mỗi dòng.

	Lượt1	Lượt2	Lượt3	Lượt4
Người chơi #1	5	2	8	9
Người chơi #2	1	3	0	7
Người chơi #3	6	8	1	0
Người chơi #4	5	2	8	3

Bảng kết quả:

	Lượt1	Lượt2	Lượt3	Lượt4
Người chơi #1	2	5	8	9
Người chơi #2	0	1	3	7
Người chơi #3	0	1	6	8
Người chơi #4	2	3	5	8

Chương trình minh hoạ:

```
class Program
{
   static void Main(string[] args)
   {
      int[,] ArrM = { { 5, 2, 8, 9 }, { 1, 3, 0, 7 }, { 6, 8, 1, 0 }, {5,2,8,3}  };

      //Sắp xếp các phần tử tăng dần theo từng

      for (int i = 0; i < ArrM.GetLength (0); i++)
      {
         //Sắp xếp dòng thứ
         for (int j = i  ; j < ArrM.GetLength(1); j++  )
         {
            for (int k = 0; k  < ArrM.GetLength(1) - 1; k++)
            {
               if (ArrM[ j , k ] > ArrM[ j , k + 1])
               {
                  int temp = ArrM[ j , k ];
                  ArrM[j , k ] = ArrM[j , k + 1];
                  ArrM[ j , k + 1] = temp;
               }
            }

         }
      }

      //In kết quả sau khi sắp xếp
      int dem = 0;
      foreach (int element in ArrM)
      {

         Console.Write("{0}\t", element.ToString());
```

```
            if (++dem % ArrM .GetLength (1) == 0)
            {
               Console.WriteLine();
            }

         }

         Console.ReadKey();
      }
}
```

Kết quả:

```
2    5    8    9
0    1    3    7
0    1    6    8
2    3    5    8
```

Ứng dụng 2: **Trò chơi Tic-Tac-Toe (Cờ caro)**

Tic-tac-toe là một trò chơi phổ biến dùng viết trên bàn cờ giấy có chín ô, 3x3. Hai người chơi, người dùng ký hiệu O, người kia dùng ký hiệu X, lần lượt điền ký hiệu của mình vào các ô. Người thắng là người thể tạo được đầu tiên một dãy liên tiếp ba ký hiệu, ngang dọc hay chéo đều được.

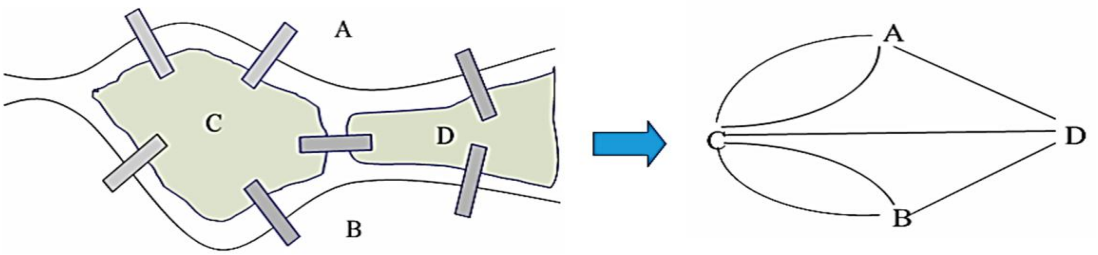
	X	O
X	O	
O		X

Ứng dụng 3: **Ứng dụng quản lý**

Trong một rạp chiếu phim có 8 hàng ghế, mỗi hàng ghế có 10 ghế ngồi. Hệ thống bán vé muốn thể hiện số ghế ngồi còn trống trong rạp. Những ghế đã có người đặt sẽ được đánh số 0, ghế còn trống sẽ được đánh số 1. Ta có bảng hai chiều thể hiện tình trạng ghế ngồi như sau:

1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1

Ứng dụng 4: Ứng dụng trong lý thuyết đồ thị.



Thành phố Konigsberg thuộc Phổ (bây giờ là Kaliningrad thuộc Cộng Hòa Liên Bang Nga), được chia thành bốn vùng bằng các nhánh sông Pregel. Vào thế kỷ 18 người ta xây bảy chiếc cầu nối các vùng này với nhau. Vào chủ nhật, người dân thường đi bộ dọc theo các phố. Họ tự hỏi không biết có thể xuất phát tại một điểm nào đó trong thành phố đi qua tất cả các cầu, mỗi chiếc cầu không đi qua nhiều hơn một lần, rồi lại trở về điểm xuất phát được không?

Nhà toán học Thụy Sỹ, Leonhard Euler, đã giải bài toán này. Lời giải công bố năm 1936, đây có thể là ứng dụng đầu tiên của lý thuyết đồ thị. Euler đã nghiên cứu bài toán này và mô hình hóa nó bằng một đa đồ thị, bốn vùng được biểu diễn bằng bốn đỉnh, và bảy cây cầu được biểu diễn bằng bảy cạnh nối với bốn đỉnh. Đồ thị trên được hiện thực trên máy tính bằng ma trận vuông 4 X 4 như sau:

	A	B	C	D
A	0	0	2	1
B	0	0	2	1
C	2	2	0	1
D	1	1	1	0

Nhờ vào việc chuyển dữ liệu đường đi giữa các địa điểm thành ma trận , con người có thể giải quyết các bai toán về tìm đường bằng máy tính như: bài toán kiểm tra tính liên thông , bài toán tìm đường đi ngắn nhất, …

1.9| BÀI TẬP

BÀI TẬP THỰC HÀNH SỐ 1

I. Thông tin chung:

- Mã số bài tập : BT-KTLT2-01
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : … / … / ……
- Nội dung : **Chương 1: Mảng hai chiều**

Chuẩn đầu ra cần đạt:

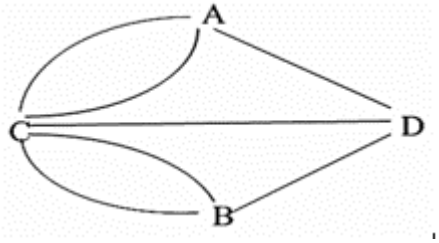
L.O.1	Sử dụng kiểu dữ liệu mảng hai chiều để xây dựng chương trình
L.O.4	Làm bài tập theo yêu cầu của Giảng viên và nộp bài đúng quy định.

Yêu cầu:

- *Sinh viên đọc Tài liệu Kĩ thuật lập trình 2 – Chương 1, tài liệu chuẩn code trước khi thực hiện bài tập.*
- *Sử dụng internet để tra cứu.*
- *Trình bày code đúng chuẩn.*
- *Viết chương trình có sử dụng mảng hai chiều*

BÀI TẬP

1. Viết ứng dụng quản lý chỗ ngồi được đặt cho một rạp chiếu phim có n hàng ghế, mỗi hàng có m ghế. Số chỗ ngồi được đánh mã là [i,j]. Viết ứng dụng cho phép người dùng chọn mã chỗ ngồi. Chương trình kiểm tra và trả về kết quả "Đặt chỗ thành công"" nếu chỗ ngồi còn trống. Nếu chỗ ngồi đã có người đặt thì in thông báo "Vui lòng chọn chỗ ngồi khác"
2. Viết đoạn chương trình thống kê số chỗ ngồi còn trống trong câu 1 như sau:
    a) Tổng số chỗ ngồi còn trống trong rạp.
    b) Số lượng ghế trống từng hàng.
    c) Số lượng ghế trống từng dãy.
    d) Số cặp ghế trống theo hàng
    e) Tìm hàng có nhiều ghế trống nhất
    f) Tìm hàng đã hết chỗ trống
    g) Kiểm tra tất cả các ghế ở ngoài biên được đặt hết hay chưa.
3. Cho đồ thị đường đi như sau:



Biết đồ thị trên được mã hoá thành ma trận vuông 4x4 sau:

0	0	2	1
0	0	2	1
2	2	0	1
1	1	1	0

Viết chương trình cho phép người dùng thực hiện các thao tác sau:

a) Nhập mảng sau và in ma trận trên ra màn hình. Biết ma trận được thành lập theo nguyên tắc là phần tử A[i,j] = A[j,i]. Các phần tử trong ma trận là số nguyên lớn hơn 0.

b) Viết hàm kiểm tra ma trận trên có phải là ma trận của một Đa đồ thị hay không. Biết đa đồ thị có ma trận với các phần tử nằm trên đường chéo chính bằng 0 và trong ma trận có ít nhất một phần tử lớn hơn 1.

c) Viết chương trình tính tổng các phần tử theo dòng (hoặc cột).

d) Viết chương trình xét Đồ thị có tồn tại một chu trình con trong nó hay không. Biết nếu tất cả các giá trị tổng trên dòng (hoặc cột) là chẵn thì Đồ thị tương ứng với ma trận sẽ có một chu trình đi qua tất cả nó.

   Biết Chu trình là đường đi có điểm xuất phát v kết thúc tại cùng một đỉnh trong đồ thị.

4. Viết chương trình mô phỏng trò chơi TIC -TAC -TOE. Biết hai người chơi sẽ nhập vào vị trí [i, j]. Mỗi lượt đặt chương trình xuất ma trận mới với các vị trí đã được đặt trước đó.
5. Viết chương trình khởi tạo giá trị các phần tử là ngẫu nhiên cho ma trận các số nguyên kích thước m× n .
6. Viết hàm in tam giác Pascal với chiều cao h.

- Ví dụ : h = 5

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

7. Viết chương trình tạo ma trận vuông mới từ ma trận hình chữ nhật. Biết kích thước ma trận vuông mới sẽ là kích thước cạnh có nhỏ nhất, Sau đó viết các hàm sau:
   a. Viết hàm tính tích các phần tử trên mỗi cột của ma trận
   b. Viết hàm tính tích các phần tử nằm trên đường chéo chính, chéo phụ của ma trận vuông.
   c. Viết hàm tính tổng các phần tử là số nguyên tố có trong ma trận.
   d. Viết hàm tính giá trị trung bình của các phần tử là số chẵn trong ma trận.
   e. Viết hàm tìm phần tử lẻ lớn nhất trong ma trận
   f. Viết hàm đếm số lần xuất hiện của phần tử x trong ma trận, với x được nhập từ bàn phím.
   g. Viết hàm sắp xếp ma trận theo thứ tự tăng dần từ trên xuống dưới và từ trái qua phải theo phương pháp dùng mảng phụ.
   h. Viết hàm sắp xếp các dòng trên ma trận theo thứ tự tăng dần.

2.

STRING

Chương này nhằm giới thiệu cho sinh viên các khái niệm chuỗi. Sử dụng chuỗi để xây dựng các chương trình vừa và nhỏ theo yêu cầu.

2.1| KHÁI NIỆM

Trong C#, String là một kiểu dữ liệu được khai báo để lưu chuỗi ký tự. Một string là một chuỗi các ký tự unicode hay là mảng các ký tự.

Phạm vi của ký tự unicode trong khoảng tuừ U+0000 đến U+FFFF. String class được định nghĩa trong thư viện chuẩn .NET. Hay nói cách khác đối tượng String là tập hợp dãy System.Char. Kích thước lớn nhất của một string objects khoảng 2GB hay khoảng một tỷ ký tự.

Các đặc điểm của lớp String:

- Lớp System.String là lớp không thể sửa đổi một khi đối tượng đã được tạo ra.
- Thuộc tính Lengh cho biết tổng số các ký tự có trong chuỗi.
- Ký tự null cũng được tính vào chiều dài chuỗi.
- Cho phép chuỗi rỗng khi khai báo

Trong C#, **String** và **string** được sử dụng song song. Thực tế chúng không có khác biệt gì, string có thể coi là một bí danh (alias) cho System.String (Tên đầy đủ bao gồm cả namespace của class String).

2.2| KHAI BÁO, KHỞI TẠO

```
// Khai báo chuỗi, không khởi tạo
string message1;

// Khai báo và khởi tạo chuỗi null
string message2 = null;

// Khai báo và khởi tạo chuỗi rỗng
// sử dụng  Empty constant thay vì ký hiệu "".
string message3 = System.String.Empty;

// Khai báo và khởi tạo chuỗi
string oldPath = "c:\\Program Files\\Microsoft Visual Studio
8.0";

// Khai báo chuỗi hằng
const string message4 = "You can't get rid of me!";
```

```
// Sử dụng System.String để khai báo
System.String greeting = "Hello World!";


// Sử dụng hàm tao khi khai báo string từ một mảng, char[]
char[] letters = { 'A', 'B', 'C' };
string alphabet = new string(letters); // chuỗi ABC

// Sử dụng hàm tạo khi khai báo string với số ký tự lặp lại
string alphabet = new string('A', 5); // chuỗi AAAAA
```

2.3| ĐỊNH DẠNG CHUỖI

❖ **Định dạng canh lề:**

Format	output
String.Format("–{0,10}–", "test");	— test–
String.Format("–{0,-10}–", "test");	–test —

❖ **Một số ký tự định dạng chuỗi:**

Escape sequence	Character name	Unicode encoding
\'	Single quote	0x0027
\"	Double quote	0x0022
\\	Backslash	0x005C
\0	Null	0x0000
\a	Alert	0x0007
\b	Backspace	0x0008
\f	Form feed	0x000C
\n	New line	0x000A

Escape sequence	Character name	Unicode encoding
\r	Carriage return	0x000D
\t	Horizontal tab	0x0009
\v	Vertical tab	0x000B
\u	Unicode escape sequence (UTF-16)	\uHHHH (range: 0000 - FFFF; example: \u00E7 = "ç")
\U	Unicode escape sequence (UTF-32)	\U00HHHHHH (range: 000000 - 10FFFF; example: \U0001F47D = "□")
\x	Unicode escape sequence similar to "\u" except with variable length	\xH[H][H][H] (range: 0 - FFFF; example: \x00E7 or \x0E7 or \xE7 = "ç")

❖ **Định dạng chuỗi số:**

Định dạng chuỗi phụ thuộc vào ngôn ngữ (văn hóa). Ví dụ, định dạng một chuỗi tiền tệ trên laptop của tôi sẽ trả về kết quả là £9.99, định dạng chuỗi tiền tệ trên một máy thiết lập vùng US sẽ trả về $9.99.

specifier	type	format	output (double 1.2345)	output (int -12345)
c	currency	{0:c}	£1.23	-£12,345.00
d	decimal (whole number)	{0:d}	System.FormatException	-12345
e	exponent / scientific	{0:e}	1.234500e+000	-1.234500e+004
f	fixed point	{0:f}	1.23	-12345.00
g	general	{0:g}	1.2345	-12345
n	number	{0:n}	1.23	-12,345.00
r	round trippable	{0:r}	1.23	System.FormatException

x	hexadecimal	{0:x4}	System.FormatException	ffffcfc7

❖ **Định dạng tuỳ chỉnh:**

specifier	type	format	output (double 1234.56)
0	zero placeholder	{0:00.000}	1234.560
#	digit placeholder	{0:#.##}	1234.56
.	decimal point placeholder	{0:0.0}	1234.6
,	thousand separator	{0:0,0}	1,235
%	percentage	{0:0%}	123456%

Bổ sung thêm các nhóm tách biệt; điều này hữu ích cho các định dạng khác nhau, dựa trên giá trị của tham số truyền vào. Ví dụ:

String.Format("{0:£#,##0.00;(£#,##0.00);Nothing}", value);

Kết quả sẽ trả về là "£1,240.00" nếu truyền vào 1243.56. Nó sẽ xuất ra cùng định dạng trong cặp ngoặc nếu giá trị là âm "(£1,240.00)", và sẽ xuất ra "Nothing" nếu giá trị là zero.

2.4| THUỘC TÍNH, PHƯƠNG THỨC LỚP STRING

Thuộc tính:

Length	Cho chiều dài của chuỗi
Char[Int32]	Trả về ký tự tại vị trí xác định trong chuỗi

Ví dụ:

```
class Program
{
   static void Main(string[] args)
   {
```

```
        String MyStr = "Cao Dang";
        Console.WriteLine("Chieu dai chuoi : "+ MyStr.Length);
        for (int i = 0; i <= MyStr.Length - 1; i++)
        {
                Console.Write("{0} ", MyStr[i] );
        }
                Console.ReadKey();
    }
}
```

Kết quả:

```
Chieu dai chuoi : 8

C a o  D a n g
```

Một số phương thức xử lý chuỗi:

Phương thức	**Ý nghĩa**
int C**ompare** (string stra, string strb) Vd: String.Compare(str1 , str2)	So sánh hai chuỗi. Trả về: -1 : str1 < str2; 0 : str1 = str2; 1. : str1 > str2
string C**oncat** (string str0, string str1, …) Vd: String .Concate (str1 , str2);	Nối chuỗi str2 vào cuối chuỗi str1, trả về chuỗi mới.
bool C**ontains**(string str1) Vd: str1.Contains(str2);	Trả true nếu str1 có chứa chuỗi str2. Ngược lại trả về false
bool **EndsWith**(string value) Vd: str1.EndsWith(str2)	Trả về true nếu str1 có chứa str2 cuối chuỗi. Ngược lại trả về false
bool **Equals**(string value) bool **Equals**(string a, string b) Vd: str1.Equals(str2)	Trả về true nếu str1 và str2 cùng giá trị, ngược lại trả về false

String.Equals(str1, str2)	
int **Indexof**(string value, int startindex) Vd: str1 .IndexOf(str2)	Trả về vị trí bắt đầu chuỗi str2 trong chuỗi str1. Nếu không có trả về -1
Public int I**ndexOfAny**(char[] anyof) Char[] ch = { 'h', 'T' }; str1.IndexOfAny(ch));	Trả về chỉ số của bất kỳ ký tự nào trong một mảng ký tự unicode (Vd: h, T) có trong str1
String **Insert**(int startindex, string value) Vd: str1.Insert(3, str2)	Chèn chuỗi str2 vào str1 tại vị trí startIndex
Bool **IsNullOrEmpty** (string str) Vd: String .IsNullOrEmpty (str1)	Trả về true nếu chuooix str null hoặc empty
int **LastIndexof**(string str) STR1.LASTINDEXOF("U"	Trả về chỉ mục (dựa trên cơ sở 0) cho sự xuất hiện cuối cùng của một chuỗi đã cho bên trong đối tượng string hiện tại
Public string remove(int startindex, int count) Vd: str1.Remove(8, 4)	Xoá trong chuỗi str1 từ vị trí thứ startindex xoá count ký tự

String **replace**(string oldvalue, string newvalue) Vd: str.Replace(str1, str2)	Thay thế tất cả chuỗi str1 xuất hiện trong chuỗi bằng chuỗi str2. Trả về chuỗi mới.
string ToLower() Vd: str1.ToLower()	Chuyển chuỗi str1 thành chữ thường. Trả về chuỗi mới
string ToUpper() Vd: str1.ToUpper()	Chuyển chuỗi str1 thành chữ hoa. Trả về chuỗi mới
string Trim() Vd: str1.Trim()	Gỡ bỏ tất cả ký tự khoảng trắng đầu và cuối chuỗi

Ví dụ:

```
using System;
using static System.Console;
namespace Co
{
   class Program
```

```
{
    static void Main(string[] args)
    {
        WriteLine("Ham xu ly chuoi trong C#");
        WriteLine("-------------------------------------");

        String str1 = "Cao Dang Thu Duc ";
        String str2 = "Thu Duc";
        String str3= "    CD2019 ";
        WriteLine("Compare : "+ String.Compare(str1 , str2 )) ;
        WriteLine("Concate : " + String.Concat(str1, " ", str2));
        WriteLine("Contains : " + str1.Contains("Dang"));
        WriteLine(value: "EndWith  : " + str1.EndsWith("Duc"));
        WriteLine("Equals  : " + str1.Equals(str2));
        WriteLine("Equals  : " + String .Equals( str1 , str2));
        WriteLine("IndexOf : {0} ", str1.IndexOf(str2));
        char[] ch = { 'H', 'T' };
        WriteLine("IndexOfAny  : {0} " , str1.IndexOfAny(ch ));
        WriteLine("Insert : {0} " ,str2.Insert(3, "-"));
        WriteLine("IsNullOrEmpty : {0} ", String.IsNullOrEmpty(str3));
        WriteLine("LastIndexOf:"
          + str1.LastIndexOf("u", StringComparison.Ordinal));
        WriteLine("Remove  : " + str1.Remove(8, 4));
        WriteLine("Replace : " + str1.Replace("Cao Dang ", "Quan "));
```

```
            WriteLine("Tolower  : " + str1.ToLower());

            WriteLine("ToUpper  : " + str1.ToUpper());

            WriteLine("Trim :" + str3.Trim());

            Console .ReadKey();

        }

    }

}
```

Kết quả:

```
Ham xu ly chuoi trong C#
-------------------------------------
Compare : -1
Concate : Cao Dang Thu Duc  Thu Duc
Contains : True
EndWith  : False
Equals  : False
Equals  : False
IndexOf : 9
IndexOfAny  : 9
Insert : Thu- Duc
IsNullOrEmpty : False
LastIndexOf  : 14
Remove  : Cao Dang Duc
Replace : Quan Thu Duc
Tolower  : cao dang thu duc
ToUpper  : CAO DANG THU DUC
Trim :CD2019
```

2.5| BÀI TẬP

BÀI TẬP THỰC HÀNH SỐ 2

I. Thông tin chung:

- Mã số bài tập : HW1-KTLT2
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : … / … / ……
- Nội dung : Chương 2. String

-

II. Chuẩn đầu ra cần đạt:

L.O.1	Sử dụng kiểu dữ liệu chuỗi để xây dựng chương trình
L.O.4	Làm bài tập và nộp bài đúng quy định.

Yêu cầu:

- *Sinh viên đọc Tài liệu Kỹ thuật lập trình 2 – Chương 2, tài liệu chuẩn code trước khi thực hiện bài tập.*
- *Sử dụng internet để tra cứu.*
- *Trình bày code đúng chuẩn.*
- *Viết chương trình có sử dụng kiểu dữ liệu String.*

BÀI TẬP

1. Viết hàm đếm số ký tự có trong chuỗi .
2. Viết hàm đếm số ký tự là chữ hoa có trong chuỗi.
3. Viết hàm đếm số ký tự là chữ số có trong chuỗi.
4. Viết hàm cho phép kiểm tra Chuỗi có tồn tại ký tự char hay không, biết char do người dùng nhập vào từ bàn phím.
5. Viết hàm đảo ngược chuỗi.
6. Viết hàm đếm số từ trong một chuỗi. Biết từ do người dùng nhập vào từ bàn phím.
7. Viết hàm so sánh hai chuỗi không phân biệt chữ hoa, chữ thường. Hàm trả về true nếu hai chuỗi giống nhau, ngược lại trả về false.
8. Viết hàm tạo email từ chuỗi họ tên của người dùng. Biết email được tạo bằng cách xoá các khoảng trắng trong chuỗi và thêm "@tdc.edu.vn".
9. Viết hàm kiểm tra chuỗi email do người dùng nhập vào có hợp lệ hay không. Biết chuỗi email hợp lệ là chuỗi không chứa các ký tự đặc biệt như #, %, $,&,^ , không có khoảng trắng nào trong chuỗi và bắt buộc phải có ký tự @ trong chuỗi.
10. Viết hàm kiểm tra chuỗi có hợp lệ hay không. Biết chuỗi hợp lệ là chuỗi không có khoảng trắng đầu và cuối chuỗi, bắt đầu bằng ký tự chữ Hoa và không chứa hai khoảng trắng liên tiếp trong chuỗi.

3. DATETIME

Chương này nhằm giới thiệu cho sinh viên các thành phần của lớp DateTime và cách sử dụng lớp DateTime.

3.1| KHÁI NIỆM

DateTime là một lớp nằm trong namespace System, giúp người dùng làm việc với thời gian. Lớp DateTime cung cấp nhiều phương thức và thuộc tính để tính toán ngày và thời gian.

3.2| KHAI BÁO VÀ KHỞI TẠO

Để khai báo và khởi tạo một đối tượng của lớp DateTime, sử dụng phương thức khởi tạo của lớp để khởi tạo giá trị (day, month, year,...) cho đối tượng tại thời điểm khai báo.

Cú pháp:

DateTime objectName = new DateTime(parameters)

Ví dụ:

```
1. DateTime dateTime = new DateTime();
2. Console.WriteLine(dateTime); // 1/1/0001 12:00:00 AM
3. // 2015 is year, 12 is month, 25 is day
4. DateTime date1 = new DateTime(2015, 12, 25);
5. Console.WriteLine(date1.ToString()); // 12/25/2015 12:00:00 AM

6.
7. // 2015 - year, 12 - month, 25 – day, 10 – hour, 30 – minute, 50
   - second
8. DateTime date2 = new DateTime(2012, 12, 25, 10, 30, 50);
9. Console.WriteLine(date1.ToString());// 12/25/2015 10:30:00 AM
```

Lớp DateTime có một thuộc tính tĩnh là Now dùng để trả về đối tượng ngày giờ hiện tại.

```
1. static void Main(string[] args)
2. {
3.      DateTime dateTime = DateTime.Now;
4.      Console.WriteLine("Today is: {0}", dateTime);
5.      Console.ReadKey();
6. }
```

3.3| MINVALUE VÀ MAXVALUE

Đối tượng lớp DateTime chứa hai trường tĩnh là MaxValue và MinValue.

```
public static readonly DateTime MinValue;
```

```
public static readonly DateTime MaxValue;
```

MinValue: là giá trị nhỏ nhất của DateTime.

```
1. // Define an uninitialized date.
2. DateTime date1 = new DateTime();
3. Console.Write(date1);
4. if (date1.Equals(DateTime.MinValue))
5.    Console.WriteLine("(Equals Date.MinValue)");
6. // output: 1/1/0001 12:00:00 AM  (Equals Date.MinValue)
```

MaxValue: là giá trị lớn nhất của DateTime.

```
1. Console.Write(DateTime.MaxValue); // 12/31/9999 11:59:59 PM
```

3.4| MỘT SỐ HÀM THƯ VIỆN DATETIME

Thuộc tính của lớp DateTime: Date, Day, Month, Year, Hour, Minute, Second, DayOfWeek, DayOfYear...

Thuộc tính	Kiểu dữ liệu	Mô tả
Date	**DateTime**	Lấy ra thành phần date (Chỉ chứa thông tin ngày tháng năm) của đối tượng này.
Day	int	Lấy ra ngày trong tháng được mô tả bởi đối tượng này.
DayOfWeek	**DayOfWeek**	Lấy ra ngày trong tuần được đại diện bởi đối tượng này.
DayOfYear	int	Lấy ra ngày của năm được đại diện bởi đối tượng này.
Hour	int	Lấy ra thành phần giờ được đại diện bởi đối tượng này.
Kind	**DateTimeKind**	Lấy giá trị cho biết liệu thời gian được đại diện bởi đối tượng này dựa trên thời gian địa phương, Coordinated Universal Time (UTC), hoặc không.
Millisecond	int	Lấy ra thành phần mili giây được đại diện bởi đối tượng này.
Minute	int	Lấy ra thành phần phút được đại diện bởi đối tượng này.
Month	int	Lấy ra thành phần tháng được đại diện bởi đối tượng này.
Now	**DateTime**	Lấy ra đối tượng DateTime được sét thông tin ngày tháng thời gian hiện tại theo máy tính địa phương.
Second	int	Lấy ra thành phần giây được đại diện bởi đối tượng này.
Ticks	long	Lấy ra số lượng "tick" được đại diện bởi đối tượng này. (1 phút = 600 triệu tick)

TimeOfDay	**TimeSpan**	Trả về thời gian của ngày được đại diện bởi đối tượng này.
Today	**DateTime**	Trả về ngày hiện tại.
UtcNow	**DateTime**	Trả về đối tượng DateTime được sét thời gian hiện tại của máy tính, được thể hiện dưới dạng Coordinated Universal Time (UTC).
Year	int	Lấy ra thành phần năm được đại diện bởi đối tượng này

Ví dụ:

```
1. DateTime myDate = new DateTime(2015, 12, 25, 10, 30, 45);
2. int year = myDate.Year; // 2015
3. int month = myDate.Month; //12
4. int day = myDate.Day; // 25
5. int hour = myDate.Hour; // 10
6. int minute = myDate.Minute; // 30
7. int second = myDate.Second; // 45
8. int weekDay = (int)myDate.DayOfWeek; // 5 due to Friday
```

Các phương thức của lớp DateTime

AddDay(): Thêm một số vào thuộc tính Day của đối tượng, trả về đối tượng DateTime mới và không làm thay đổi đối tượng hiện hành.

```
public DateTime AddDays (double value);
```

```
using System;

class Class1
{
      static void Main()
      {
            DateTime today = DateTime.Now;
            DateTime answer = today.AddDays(36);
         Console.WriteLine("Today: {0:dddd}", today);
         Console.WriteLine("36 days from today: {0:dddd}", answer);
      }
}
// The example displays output like the following:
//       Today: Wednesday

//       36 days from today: Thursday
```

AddMonth(): Thêm một số vào thuộc tính Month của đối tượng, trả về đối tượng DateTime mới và không làm thay đổi đối tượng hiện hành.

```
public DateTime AddMonths (int months);
```

```
using System;

public class Example
{
   public static void Main()
   {
      var dat = new DateTime(2015, 12, 31);
      for (int ctr = 0; ctr <= 4; ctr++)
         Console.WriteLine(dat.AddMonths(ctr).ToString("d"));

   }
}
// The example displays the following output:
//       12/31/2015
//       1/31/2016
//       2/29/2016
//       3/31/2016
//       4/30/2016
```

AddYears():Thêm một số vào thuộc tính Year của đối tượng, trả về đối tượng DateTime mới và không làm thay đổi đối tượng hiện hành.

```
public DateTime AddYears (int value);
```

```
using System;

public class Example
{
   public static void Main()
   {
      DateTime baseDate = new DateTime(2000, 2, 29);
      Console.WriteLine("    Base Date:        {0:d}\n", baseDate);

      // Show dates of previous fifteen years.
      for (int ctr = -1; ctr >= -5; ctr--)
         Console.WriteLine("{0,2} year(s) ago:        {1:d}",
                           Math.Abs(ctr), baseDate.AddYears(ctr));
      Console.ReadKey();
   }
}
// The example displays the following output:
//           Base Date:        2/29/2000
//
//        1 year(s) ago:        2/28/1999
//        2 year(s) ago:        2/28/1998
//        3 year(s) ago:        2/28/1997
//        4 year(s) ago:        2/29/1996
```

```
//        5 year(s) ago:        2/28/1995
```

AddHours():Thêm một số vào thuộc tính Hour của đối tượng, trả về đối tượng DateTime mới và không làm thay đổi đối tượng hiện hành.

```
public DateTime AddHours (double value);
```

```
using System;

public class Example
{
   public static void Main()
   {
      double[] hours = {.08333, .16667, .25, .33333, .5, .66667, 1,
2};
      DateTime dateValue = new DateTime(2009, 3, 1, 12, 0, 0);

      foreach (double hour in hours)
         Console.WriteLine("{0} + {1} hour(s) = {2}", dateValue, hour,
                           dateValue.AddHours(hour));

   }
}
// The example displays the following output on a system whose current
// culture is en-US:
//    3/1/2009 12:00:00 PM + 0.08333 hour(s) = 3/1/2009 12:04:59 PM
//    3/1/2009 12:00:00 PM + 0.16667 hour(s) = 3/1/2009 12:10:00 PM
//    3/1/2009 12:00:00 PM + 0.25 hour(s) = 3/1/2009 12:15:00 PM
//    3/1/2009 12:00:00 PM + 0.33333 hour(s) = 3/1/2009 12:19:59 PM
//    3/1/2009 12:00:00 PM + 0.5 hour(s) = 3/1/2009 12:30:00 PM
//    3/1/2009 12:00:00 PM + 0.66667 hour(s) = 3/1/2009 12:40:00 PM
//    3/1/2009 12:00:00 PM + 1 hour(s) = 3/1/2009 1:00:00 PM

//    3/1/2009 12:00:00 PM + 2 hour(s) = 3/1/2009 2:00:00 PM
```

AddMinutes():Thêm một số vào thuộc tính Minute của đối tượng, trả về đối tượng DateTime mới và không làm thay đổi đối tượng hiện hành

```
public DateTime AddMinutes (double value);
```

```
using System;

public class Example
{
   public static void Main()
   {
      DateTime dateValue = new DateTime(2013, 9,15, 12, 0, 0);
```

```
      double[] minutes = { .01667, .08333, .16667, .25};

      foreach (double minute in minutes)
         Console.WriteLine("{0} + {1} minute(s) = {2}", dateValue,
minute, dateValue.AddMinutes(minute));
   }
}
// The example displays the following output on a system whose current
culture is en-US:
//    9/15/2013 12:00:00 PM + 0.01667 minute(s) = 9/15/2013 12:00:01
PM
//    9/15/2013 12:00:00 PM + 0.08333 minute(s) = 9/15/2013 12:00:05
PM
//    9/15/2013 12:00:00 PM + 0.16667 minute(s) = 9/15/2013 12:00:10
PM
//    9/15/2013 12:00:00 PM + 0.25 minute(s) = 9/15/2013 12:00:15 PM
```

AddSecond():Thêm một số vào thuộc tính Second của đối tượng, trả về đối tượng DateTime mới và không làm thay đổi đối tượng hiện hành.

```
public DateTime AddSeconds (double value);
```

```
public static void Main()
{
      string dateFormat = "MM/dd/yyyy hh:mm:ss";
      DateTime date1 = new DateTime(2014, 9, 8, 16, 0, 0);
      Console.WriteLine("Original date: {0} ({1:N0} ticks)\n",
date1.ToString(dateFormat), date1.Ticks);

      DateTime date2 = date1.AddSeconds(30);
      Console.WriteLine("Second date:   {0} ({1:N0} ticks)",
date2.ToString(dateFormat), date2.Ticks);
}
// The example displays the following output:
//    Original date: 09/08/2014 04:00:00 (635,457,888,000,000,000
ticks)

//    Second date:   09/08/2014 04:00:30 (635,457,888,300,000,000 ticks)
```

3.5| CÁC TOÁN TỬ TRÊN DATETIME

Đối tượng lớp DateTime có thể sử dụng các toán tử cộng (+), trừ(-), so sánh (==, !=, >, <, >=, <=).

```
1. // It is 10th December 2015
2. DateTime dtObject = new DateTime(2015, 12, 10); // 12/10/2015 12:
   00:00 AM
```

```
3. // TimeSpan object with 15 days, 10 hours, 5 minutes and 1 second
   .
4. TimeSpan timeSpan = new TimeSpan(15, 10, 5, 1);
5. DateTime addResult = dtObject + timeSpan; // 12/25/2015 10:05:01
   AM
6. DateTime substarctResult = dtObject - timeSpan; // 11/24/2015 1:5
   4:59 PM
7.
8. DateTime dec25 = new DateTime(2015, 12, 25);
9. DateTime dec15 = new DateTime(2015, 12, 15);
10.
11. // areEqual gets false.
12. bool areEqual = dec25 == dec15;
13. DateTime newDate = new DateTime(2015, 12, 25);
14. // areEqual gets true.
15. areEqual = dec25 == newDate
```

3.6| CHUỖI ĐỊNH DẠNG DATETIME

Định dạng **DateTime** nghĩa là chuyển đổi đối tượng **DateTime** thành một string theo một khuôn mẫu nào đó, chẳng hạn theo định dạng ngày/tháng/năm, ... hoặc định dạng dựa vào địa phương (locale) cụ thể.

Có nhiều chuỗi định dạng cho đối tượng lớp DateTime.

Specifier	Description	Output
d	Short Date	12/8/2015
D	Long Date	Tuesday, December 08, 2015
t	Short Time	3:15 PM
T	Long Time	3:15:19 PM
f	Full date and time	Tuesday, December 08, 2015 3:15 PM
F	Full date and time (long)	Tuesday, December 08, 2015 3:15:19 PM
g	Default date and time	12/8/2015 15:15
G	Default date and time (long)	12/8/2015 15:15
M	Day / Month	8-Dec
r	RFC1123 date	Tue, 08 Dec 2015 15:15:19 GMT
s	Sortable date/time	2015-12-08T15:15:19
u	Universal time, local timezone	2015-12-08 15:15:19Z
Y	Month / Year	December, 2015
dd	Day	8
ddd	Short Day Name	Tue
dddd	Full Day Name	Tuesday
hh	2 digit hour	3
HH	2 digit hour (24 hour)	15
mm	2 digit minute	15
MM	Month	12
MMM	Short Month name	Dec

MMMM	Month name	December
ss	seconds	19
fff	milliseconds	120
FFF	milliseconds without trailing zero	12
tt	AM/PM	PM
yy	2 digit year	15
yyyy	4 digit year	2015
:	Hours, minutes, seconds separator, e.g. {0:hh:mm:ss}	9:08:59
/	Year, month , day separator, e.g. {0:dd/MM/yyyy}	8/4/2007

Ví dụ:

```
1. DateTime tempDate = new DateTime(2015, 12, 08); // creating date
   object with 8th December 2015
2. Console.WriteLine(tempDate.ToString("MMMM dd, yyyy")); //December
    08, 2105.
```

3.7| CHUYỂN ĐỔI STRING SANG DATETIME

Để chuyển đổi chuỗi sang đối tượng lớp DateTime, có thể sử dụng những phương thức sau của lớp DateTime: DateTime.Parse(), DateTime.ParseExact(), DateTime.TryParse(), DateTime.TryParseExact(). Ngoài ra có thể sử dụng Convert.ToDateTime().

```
public static DateTime Parse(string s)

public static DateTime Parse(string s, IFormatProvider provider)

public static DateTime Parse(string s, IFormatProvider provider,
DateTimeStyles styles)

public static bool TryParseExact(string s, string format,
           IFormatProvider provider, DateTimeStyles style,
           out DateTime result)

public static bool TryParseExact(string s, string[] formats,
           IFormatProvider provider, DateTimeStyles style,

           out DateTime result)
```

Ví dụ 1:

```
using System;
using System.Globalization;

public class DateTimeParser
{
```

```
    public static void Main()
    {
        // Use standard en-US date and time value
        DateTime dateValue;
        string dateString = "2/8/2019 12:15:12 PM";
        try
        {
            dateValue = DateTime.Parse(dateString);
            Console.WriteLine("'{0}' converted to {1}.", dateString,
dateValue);
            Console.WriteLine("Day Of Week: {0}",
dateValue.DayOfWeek);
        }
        catch (FormatException)
        {
            Console.WriteLine("Unable to convert '{0}'.", dateString);
        }
        Console.ReadKey();
    }

}
```

Kết quả:

```
'2/8/2019 12:15:12 PM' converted to 02/08/2019 12:15:12 PM.
Day Of Week: Friday
```

Ví dụ 2:

```
using System;
using System.Globalization;

public class Example
{
    public static void Main()
    {
        string dateString, format;
        DateTime result;
        CultureInfo provider = CultureInfo.InvariantCulture;

        // Parse date-only value with invariant culture.
        dateString = "06/15/2008";
        format = "d";
        try
        {
            result = DateTime.ParseExact(dateString, format,
provider);
            Console.WriteLine("{0} converts to {1}.", dateString,
result.ToString());
```

```
        }
        catch (FormatException)
        {
            Console.WriteLine("{0} is not in the correct format.",
dateString);
        }
        Console.ReadKey();
    }

}
```

Kết quả:

```
06/15/2008 converts to 15/06/2008 12:00:00 AM.
```

Ví dụ 3:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DateTimeTutorial
{
    class ParseDateStringExample
    {

        public static void Main(string[] args)
        {

            string dateString = "20160319 09:57";

            // Sử dụng ParseExact để phân tích một String thành
DateTime.
            DateTime dateTime = DateTime.ParseExact(dateString,
"yyyyMMdd HH:mm", null);

            Console.WriteLine("Input dateString: " + dateString);

            Console.WriteLine("Parse Results: " +
dateTime.ToString("dd-MM-yyyy HH:mm:ss"));

            // Một chuỗi datetime có khoảng trắng phía trước.
            dateString = "  20110319 11:57";

            // Sử dụng phương thức TryParseExact.
            // Phương thức này trả về true nếu chuỗi 'dateString' có
thể phân tích được.
```

```
            // Và trả về giá trị cho tham số 'out dateTime'.
            bool successful = DateTime.TryParseExact(dateString,
"yyyyMMdd HH:mm", null,

System.Globalization.DateTimeStyles.AllowLeadingWhite,
                        out dateTime);

            Console.WriteLine("\n-----------------------\n");

            Console.WriteLine("Input dateString: " + dateString);

            Console.WriteLine("Can Parse? :" + successful);

            if (successful)
            {
                Console.WriteLine("Parse Results: " +
dateTime.ToString("dd-MM-yyyy HH:mm:ss"));
            }

            Console.Read();
        }
    }

}
```

Kết quả:

```
Input dateString: 20160319 09:57
Parse Results: 19-03-2016 09:57:00

-----------------------

Input dateString:   20110319 11:57
Can Parse? :True
Parse Results: 19-03-2011 11:57:00
```

3.8| BÀI TẬP

BÀI TẬP THỰC HÀNH SỐ 3

II. Thông tin chung:

- Mã số bài tập : HW3-KTLT2
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : … / … / ……
- Nội dung : Chương 3: DateTime

Chuẩn đầu ra cần đạt:

L.O.1	Sử dụng kiểu dữ liệu DateTime để xây dựng chương trình
L.O.4	Làm bài tập theo yêu cầu của Giảng viên và nộp bài đúng quy định.

Yêu cầu:

- *Sinh viên đọc Tài liệu Kỹ thuật lập trình 2 – Chương 3, tài liệu chuẩn code trước khi thực hiện bài tập.*
- *Sử dụng internet để tra cứu.*
- *Trình bày code đúng chuẩn.*
- *Viết chương trình có sử dụng kiểu dữ liệu DateTime.*

BÀI TẬP

1. Hãy viết chương trình cho phép người dùng xuất ra ngày trong tuần, ngày, tháng, năm của ngày giờ hiện tại.

2. Viết chương trình cho phép người dùng tính ngày trước, ngày sau của một ngày.

3. Bạn An mượn sách của thư viện, ngày hạn phải trả là ngày x. Nhưng vì một số lý do An đã không thể trả sách đúng hạn. Ngày trả thực tế là ngày y. Hãy tính xem An đã trễ trả sách bao nhiêu ngày?

4. Viết chương trình cho phép lưu trữ danh sách nhân viên của một công ty. Biết rằng mỗi nhân viên cần lưu trữ thông tin sau:

   Họ tên nhân viên: string

   Ngày sinh: DateTime

   Lương cơ bản: double

   Hệ số lương: double

Các chức năng của chương trình:

a. Nhập danh sách nhân viên

b. Liệt kê họ tên của những nhân viên sinh trong quý 1.

4. STRUCT

Chương này nhằm giới thiệu cho sinh viên các khái niệm về kiểu dữ liệu struct, khai báo và khởi tạo giá cho biến, truy xuất các thành phần trên struct, sử dụng struct giải quyết bài toán cụ thể.

4.1| KHÁI NIỆM CẤU TRÚC

Cấu trúc là kiểu dữ liệu bao gồm nhiều phần tử. Các phần tử của cấu trúc là các dữ liệu thuộc các kiểu khác nhau và có tên khác nhau. Kiểu cấu trúc được định nghĩa bởi từ khóa **struct.** Mỗi phần tử của kiểu dữ liệu cấu trúc được gọi là một trường.

Dữ liệu kiểu cấu trúc được dùng để mô tả các đối tượng bao gồm các kiểu dữ liệu khác nhau, như hóa đơn mua hàng, phiếu xuất vật tư, lý lịch nhân viên, phiếu thu tiền, . . . Các dữ liệu này rất thường gặp trong các bài toán thông tin kinh tế, quản lý.

Cấu trúc là công cụ để tạo ra kiểu dữ liệu mới. Sau này kiểu cấu trúc mở rộng thành kiểu lớp.

Bên trong struct ngoài các biến có kiểu dữ liệu cơ bản còn có các phương thức, các struct khác.

4.2| KHAI BÁO CẤU TRÚC

Muốn sử dụng kiểu dữ liệu cấu trúc ta phải định nghĩa nó để xác định tên cùng với các thành phần dữ liệu có trong kiểu cấu trúc này. Một kiểu cấu trúc được khai báo theo mẫu sau:

```
struct <tên kiểu>
{
     // các thành phần;
     public <danh sách các biến>;

}; // có thể có dấu ; hoặc không
```

<tên kiểu> là tên kiểu dữ liệu do mình tự đặt và tuân thủ theo quy tắc đặt tên.

<danh sách các biến> là danh sách các biến thành phần được khai báo như khai báo biến bình thường.

Từ khoá public là từ khoá chỉ định phạm vi truy cập. Từ khoá này giúp cho người khác có thể truy xuất được để sử dụng.

Ví dụ:

```
struct SinhVien
{
    public int MaSo;
    public string HoTen;
```

```
    public double DiemToan;
    public double DiemLy;
    public double DiemVan;
}
```

Với khai báo này đã tạo ra một kiểu dữ liệu mới tên là SinhVien. Và có thể khai báo biến, sử dụng nó như sử dụng các kiểu dữ liệu khác.

Nếu như kiểu int có thể chứa số nguyên, kiểu double có thể chứa số thực thì kiểu SinhVien vừa khai báo có thể chứa 5 trường thông tin con là MaSo, HoTen, DiemToan, DiemLy, DiemVan.

Cấu trúc có thể có các phương thức khác, có phương thức khởi tạo (constructor) đã được định nghĩa, nhưng không có phương thức hủy (destructor). Tuy nhiên, không thể định nghĩa một constructor mặc định cho một cấu trúc. Constructor mặc định được định nghĩa tự động và không thể bị thay đổi.

4.3| KHAI BÁO BIẾN KIỂU CẤU TRÚC

Khai báo biến kiểu cấu trúc cũng giống như khai báo các biến kiểu cơ sở dưới dạng:

<tên cấu trúc> <danh sách biến>;

Hoặc sử dụng toán tử new:

<tên cấu trúc > biến = new < tên cấu trúc>(<danh sách giá trị khởi tạo cho biến>)

Khởi tạo biến cấu trúc:

Khi tạo một biến kiểu cấu trúc bởi sử dụng toán tử new, nó lấy đối tượng đã tạo và constructor thích hợp được gọi. Biến kiểu cấu trúc cũng có thể được khởi tạo mà không cần sử dụng toán tử new. Nếu toán tử new không được sử dụng, thì các trường chưa được gán và đối tượng không thể được sử dụng tới khi tất cả trường đó được khởi tạo.

Ví dụ:

```
using System;

struct Books
{
    public string title;
    public string author;
    public string subject;
    public int book_id;

    public Books(string t, string a, string s, int id)
    {
```

```
        title = t;
        author = a;
        subject = s;
        book_id = id;
    }

 };

public class testStructure
{
    public static void Main(string[] args)
    {
        Books Book1;   /* Khai báo biến Book1 kiểu Books*/

        // Khai báo và khởi tạo biến Book2 (sử dụng toán tử new và
constructor)
        Books Book2 = new Books("Telecom Billing",
        "Zara Ali", "Telecom Billing Tutorial", 6495700);

       // string str = Book1.title; // sẽ bị lỗi vì Book1 chưa được
khởi tạo
        /* Khởi tạo Book1 */
        Book1.title = "C Programming";
        Book1.author = "Nuha Ali";
        Book1.subject = "C Programming Tutorial";
        Book1.book_id = 6495407;

        Console.ReadKey();
    }
}
```

Kết quả:

```
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id :6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700
```

Ví dụ sau minh họa một cấu trúc có thể lồng một cấu trúc khác

```
        //khai bao cau truc NgaySinh
        struct NgaySinh
        {
            public int Day;
            public int Month;
            public int Year;
```

```
        }

        // khai bao cau truc NhanVien
        struct NhanVien
        {
            public string eName;
            public NgaySinh Date;
        }
```

4.4| TRUY CẬP CÁC THÀNH PHẦN TRONG CẤU TRÚC

Để truy nhập vào các thành phần kiểu cấu trúc ta sử dụng cú pháp:

<tên biến>.<tên thành phần >

Đối với các struct lồng nhau:

Truy nhập thành phần ngoài rồi đến thành phần của cấu trúc bên trong, sử dụng toán tử "." một cách thích hợp.

Ví dụ:

```
struct Books
{
    public string title;
    public string author;
    public string subject;
    public int book_id;

    public Books(string t, string a, string s, int id)
    {
        title = t;
        author = a;
        subject = s;
        book_id = id;
    }

 };

public static void Main(string[] args)
    {
        Books Book1;   /* Khai báo biến Book1 kiểu Books*/

        /* Khởi tạo Book1 */
        Book1.title = "C Programming";
        Book1.author = "Nuha Ali";
        Book1.subject = "C Programming Tutorial";
```

```
        Book1.book_id = 6495407;

        /* In thông tin Book1 */
        Console.WriteLine("Book 1 title : {0}", Book1.title);
        Console.WriteLine("Book 1 author : {0}",Book1.author);
        Console.WriteLine("Book 1 subject:{0}",Book1.subject);
        Console.WriteLine("Book 1 book_id:{0}",Book1.book_id);

        Console.ReadKey();
    }
}
```

Ví dụ cấu trúc lồng:

```
using System;
namespace MyNamespace
{
    class TestCsharp
    {
        //khai bao mot struct bao gom tenNhanVien va ngaySinh
        //trong do, NgaySinh la mot struct
        struct NhanVien
        {
            public string eName;
            public NgaySinh Date;
        }
        //khai bao cau truc NgaySinh
        struct NgaySinh
        {
            public int Day;
            public int Month;
            public int Year;
        }
        static void Main(string[] args)
        {
            int dd = 0, mm = 0, yy = 0;
            int total = 2;
            Console.Write("\nLong struct trong C#:\n");
            Console.Write("-------------------------\n");
            NhanVien[] emp = new NhanVien[total];
            for (int i = 0; i < total; i++)
            {
                Console.Write("Ten nhan vien: ");
                string nm = Console.ReadLine();
                emp[i].eName = nm;
                Console.Write("Nhap ngay sinh: ");
                dd = Convert.ToInt32(Console.ReadLine());
                emp[i].Date.Day = dd;
                Console.Write("Nhap thang sinh: ");
```

```
                mm = Convert.ToInt32(Console.ReadLine());
                emp[i].Date.Month = mm;
                Console.Write("Nhap nam sinh: ");
                yy = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine();
                emp[i].Date.Year = yy;
            }
            Console.ReadKey();
        }
    }

}
```

4.5| PHÉP TOÁN GÁN CẤU TRÚC

Đối với biến kiểu struct chúng ta có thể thực hiện gán giá trị của 2 biến cho nhau. Phép gán này cũng tương đương với việc gán từng thành phần của cấu trúc. Ví dụ:

```
Books Book1;   /* Declare Book1 of type Book */
Books Book2;

/* book 1 specification */
Book1.title = "C Programming";
Book1.author = "Nuha Ali";
Book1.subject = "C Programming Tutorial";
Book1.book_id = 6495407;

Book2 = Book1; // phép gán giữa 2 biến kiểu Book
```

4.6| SỬ DỤNG STRUCT LÀM ĐỐI SỐ CHO HÀM

Một cấu trúc có thể được sử dụng để làm đối của hàm dưới các dạng tham trị, tham chiếu, mảng cấu trúc.

Ví dụ:

Chương trình nhập thông tin một Sinh viên sau đó xuất toàn bộ thông tin sinh viên ra màn hình đồng thời xuất ra điểm trung bình của sinh viên đó:

```
using System;

struct SinhVien
{
    public int MaSo;
    public string HoTen;
    public double DiemToan;
    public double DiemLy;
    public double DiemVan;
```

```
}

namespace MyNamespace
{
    public class MyStruct
    {
        static void NhapThongTinSinhVien(out SinhVien SV)
        {
            Console.Write(" Ma so: ");
            SV.MaSo = int.Parse(Console.ReadLine());
            Console.Write(" Ho ten: ");
            SV.HoTen = Console.ReadLine();
            Console.Write(" Diem toan: ");
            SV.DiemToan = Double.Parse(Console.ReadLine());
            Console.Write(" Diem ly: ");
            SV.DiemLy = Double.Parse(Console.ReadLine());
            Console.Write(" Diem van: ");
            SV.DiemVan = Double.Parse(Console.ReadLine());
        }

        static void XuatThongTinSinhVien(SinhVien SV)
        {
            Console.WriteLine(" Ma so: " + SV.MaSo);
            Console.WriteLine(" Ho ten: " + SV.HoTen);
            Console.WriteLine(" Diem toan: " + SV.DiemToan);
            Console.WriteLine(" Diem ly: " + SV.DiemLy);
            Console.WriteLine(" Diem van: " + SV.DiemVan);
        }

        static double DiemTBSinhVien(SinhVien SV)
        {
            return (SV.DiemToan + SV.DiemLy + SV.DiemVan) / 3;
        }

        static void Main(string[] args)
        {
            SinhVien SV1 = new SinhVien();
            Console.WriteLine(" Nhap thong tin sinh vien: ");
            /*
             * Đây là hàm hỗ trợ nhập thông tin sinh viên.
             * Sử dụng từ khoá out để có thể cập nhật giá trị nhập
được ra biến SV1 bên ngoài
             */
            NhapThongTinSinhVien(out SV1);
            Console.WriteLine("*********");
            Console.WriteLine(" Thong tin sinh vien vua nhap la: ");
            XuatThongTinSinhVien(SV1);
            Console.WriteLine(" Diem TB cua sinh vien la: " +
DiemTBSinhVien(SV1));
```

```
            Console.ReadLine();
        }

    }

}
```

4.7| MẢNG STRUCT

Mảng kiểu cấu trúc được dùng để lưu trữ danh sách các đối tượng như danh sách học sinh, danh sách các cuốn sách, danh sách nhân viên,...

Cú pháp:

<Tên Kiểu Dữ Liệu Cấu Trúc> <Tên biến mảng> = new <Tên Kiểu Dữ Liệu Cấu Trúc>[<tổng số phần tử của mảng>];

Ví dụ:

```
struct SinhVien
{
    public int MaSo;
    public string HoTen;
    public double DiemToan;
    public double DiemLy;
    public double DiemVan;

}

static void Main(string[] args)
{
            int n;
            n = Convert.ToInt32(Console.ReadLine());

            SinhVien[] arr = new SinhVien[n];

}
```

Truyền mảng cấu trúc vào cho hàm: cách truyền mảng kiểu dữ liệu có cấu trúc vào cho hàm cũng giống như mảng kiểu dữ liệu cơ sở.

Mã nguồn sau đây minh họa ứng dụng cho phép người dùng lưu trữ danh sách n sinh viên vào trong mảng, nhập và xuất thông tin danh sách sinh viên ra màn hình.

```
using System;
```

```
struct SinhVien
{
    public int MaSo;
    public string HoTen;
    public double DiemToan;
    public double DiemLy;
    public double DiemVan;
}

namespace MyNamespace
{
    public class MyStruct
    {
        static void NhapThongTinSinhVien(out SinhVien SV)
        {
            Console.Write(" Ma so: ");
            SV.MaSo = int.Parse(Console.ReadLine());
            Console.Write(" Ho ten: ");
            SV.HoTen = Console.ReadLine();
            Console.Write(" Diem toan: ");
            SV.DiemToan = Double.Parse(Console.ReadLine());
            Console.Write(" Diem ly: ");
            SV.DiemLy = Double.Parse(Console.ReadLine());
            Console.Write(" Diem van: ");
            SV.DiemVan = Double.Parse(Console.ReadLine());
        }

        static void XuatThongTinSinhVien(SinhVien SV)
        {
            Console.WriteLine(" Ma so: " + SV.MaSo);
            Console.WriteLine(" Ho ten: " + SV.HoTen);
            Console.WriteLine(" Diem toan: " + SV.DiemToan);
            Console.WriteLine(" Diem ly: " + SV.DiemLy);
            Console.WriteLine(" Diem van: " + SV.DiemVan);
        }

        static void NhapDanhSach(SinhVien[] arr, int n)
        {
            for (int i = 0; i < n; i++)
            {
                NhapThongTinSinhVien(out arr[i]);
            }
        }

        static void XuatDanhSach(SinhVien[] arr, int n)
        {
            for (int i = 0; i < n; i++)
            {
                XuatThongTinSinhVien(arr[i]);
```

```
            }
        }

        static void Main(string[] args)
        {
            int n;
            n = Convert.ToInt32(Console.ReadLine());

            SinhVien[] arr = new SinhVien[n];

            NhapDanhSach(arr, n);
            XuatDanhSach(arr, n);

            Console.ReadLine();
        }

    }

}
```

4.8| BÀI TẬP

BÀI TẬP THỰC HÀNH SỐ 4

III. Thông tin chung:

- Mã số bài tập : HW4-KTLT2
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : … / … / ……
- Nội dung : Chương 4: Struct

Chuẩn đầu ra cần đạt:

L.O.2	Sử dụng kiểu cấu trúc và tập tin để giải quyết một số bào toán quản lý theo yêu cầu.
L.O.4	Làm bài tập theo yêu cầu của Giảng viên và nộp bài đúng quy định.

Yêu cầu:

- *Sinh viên đọc Tài liệu Kỹ thuật lập trình 2 – Chương 3, tài liệu chuẩn code trước khi thực hiện bài tập.*
- *Sử dụng internet để tra cứu.*
- *Trình bày code đúng chuẩn.*
- *Viết chương trình có sử dụng kiểu dữ liệu Struct.*

BÀI TẬP

Câu 1.

Viết chương trình để quản lý các gói cước sử dụng 3G của nhà mạng Viettel. Biết rằng mỗi gói cước cần lưu trữ các thông tin sau:

- **Tên gói**: do người dùng nhập vào từ bàn phím, gồm tối đa 10 ký tự. Ví dụ: MIMAX1, MIMAX3, MIMAX6.
- **Chu kỳ gói**: chương trình tự động tính từ tên gói. Biết chu kỳ là số ngày thuê bao được sử dụng. Tùy theo ký tự cuối trong Tên gói là 1, 3 hay 6 tương ứng với số số ngày là 30, 90 hay180.
- **Giá gói**: là số tiền người dùng phải trả cho một chu kỳ của gói 3G. Giá trị do người dùng nhập, tối đa 6 chữ số. Ví dụ: 70000, 210000, 420000.
- **Vượt gói**: là giá trị logic do người dùng nhập từ bàn phím để báo cho hệ thống biết ngắt (1) hoặc không ngắt (0) kết nối khi người dùng sử dụng hết lưu lượng tốc độ cao của gói cước 3G

Xây dựng và thực thi các chức năng sau:

a. Nhập danh sách gồm n gói cước
b. Xuất danh sách ra màn hình

Câu 2.

Chương trình quản lý danh sách và thông tin các khách hàng là thuê bao sử dụng 3G của nhà mạng Viettel bao như sau:

– ***Thông tin về Gói cước gồm:***

- **Tên gói**: do người dùng nhập vào từ bàn phím, gồm tối đa 10 ký tự. Ví dụ: MIMAX1, MIMAX3, MIMAX6.

- **Chu kỳ gói**: chương trình tự động tính từ tên gói. Biết chu kỳ là số ngày thuê bao được sử dụng. Tùy theo ký tự cuối trong Tên gói là 1, 3 hay 6 tương ứng với số số ngày là 30, 90 hay180.
- **Giá gói**: là số tiền người dùng phải trả cho một chu kỳ của gói 3G. Giá trị do người dùng nhập, tối đa 6 chữ số. Ví dụ: 70000, 210000, 420000.
- **Vượt gói**: là giá trị logic do người dùng nhập từ bàn phím để báo cho hệ thống biết ngắt (1) hoặc không ngắt (0) kết nối khi người dùng sử dụng hết lưu lượng tốc độ cao của gói cước 3G

– ***Thông tin về <u>Thuê bao</u> gồm:***
- **Họ tên**: là chuỗi không bao gồm khoảng trắng, viết hoa đầu từ.
- **Số CMND**: là chuỗi gồm 9 ký tự số do người dùng nhập vào từ bàn phím.
- Các Thông tin về gói 3G gồm **Tên gói**, **Chu kỳ gói**, **Giá gói**, **Vượt gói**.

Yêu cầu:
1. Hãy tổ chức và khai báo các struct cho chương trình.
2. Viết hàm nhập các thông tin của thuê bao gồm: Họ tên, Số CMND, Tên gói, Giá gói, Vượt gói.
3. Viết hàm hiển thị các thông tin của thuê bao vừa nhập ra màn hình.
4. Viết hàm main gọi tất cả các hàm đã có trong chương trình.

<u>Câu 3:</u>

Chương trình quản lý danh sách và thông tin các khách hàng là thuê bao sử dụng dịch vụ truyền hình FPT như sau:

– ***Thông tin về <u>Gói cước TV</u> gồm:***
- **Tên gói**: do người dùng nhập vào từ bàn phím, gồm tối đa 4 ký tự. Ví dụ: 16TV, 22TV, 27TV.
- **Tốc độ**: chương trình tự động tính từ tên gói. Biết tốc độ là hai số đầu tiên trong Tên gói. Ví dụ Tên gói là 16TV thì tốc độ có giá trị là 16 (Mbps), ….
- **Giá gói**: là số tiền người dùng phải trả cho một chu kỳ của gói 3G. Giá trị do người dùng nhập, tối đa 6 chữ số. Ví dụ: 265.000, 305.000, 365.000.
- **Phí hòa mạng**: là dữ liệu dạng số do chương trình tự động tính theo Quận. Biết quận 1, 3, 5, 7 phí hòa mạng là 700.000 đồng, các quận khác phí là 0 đồng.

– ***Thông tin về <u>Thuê bao TV</u> gồm:***
- **Họ tên**: là chuỗi không bao gồm khoảng trắng giữa chuỗi.
- **Số CMND**: là chuỗi gồm 9 ký tự số do người dùng nhập vào từ bàn phím.
- **Quận**: là dữ liệu dạng chuỗi do người dùng nhập vào tối đa 10 ký tự.
- **Gói cước sử dụng**: bao gồm thông tin: Tên gói, Tốc độ, Giá gói, Phí hòa mạng.

Yêu cầu:
1. Hãy tổ chức và khai báo các struct cho chương trình.
2. Viết hàm nhập các thông tin của thuê bao gồm: Họ tên, Số CMND, Quận, thông tin gói cước sử dụng gồm: Tên gói, Giá gói.
3. Viết hàm hiển thị các thông tin của thuê bao vừa nhập ra màn hình.
4. Viết hàm tính Phí hòa mạng biết khách hàng ở các quận 1, 3, 5, 7 phí hòa mạng là 700.000 đồng, các quận khác phí là 0 đồng.
5. Viết hàm tính giá cho trường Tốc độ biết rằng giá trị tùy theo hai ký số đầu trong Tên gói tương ứng với giá trị là 16, 22 hay 27.

6. Viết hàm main gọi tất cả các hàm đã có trong chương trình.

Lưu ý: Trình bày code theo chuẩn.

5.

FILE

Chương này nhằm giới thiệu cho sinh viên các khái niệm file. Sử dụng File để thực hiện Input và Ouput cho chương trình.

5.1| KHÁI NIỆM

Tập tin là tập hợp dữ liệu được lưu trữ trên đĩa với tên, phần mở rộng và đường dẫn thư mục cụ thể. Khi mở tập tin bằng C# cho mục đích đọc và ghi, phải sử dụng luồng.

Luồng được sử dụng khi chương trình cần "đọc" hoặc "ghi" dữ liệu vào nguồn dữ liệu ngoài như file, PC, máy chủ khác, v.v. Luồng là một chuỗi byte được sắp xếp theo thứ tự, được gửi từ một ứng dụng hoặc thiết bị đầu vào đến một ứng dụng hoặc thiết bị đầu ra khác. Các byte này được viết và đọc lần lượt từng byte và luôn đến theo thứ tự như chúng được gửi.

Tùy thuộc vào luồng, có những luồng hỗ trợ cả đọc và ghi, và cả tìm kiếm (seek) bằng cách di chuyển con trỏ trên luồng, và ghi đọc dữ liệu tại vị trí con trỏ.

Với Stream bạn có thể ghi từng byte hoặc ghi một mảng các byte vào luồng (stream). Và khi đọc bạn có thể đọc từng byte hoặc đọc nhiều byte và gán vào một mảng tạm.

Mỗi khi đọc hoặc ghi từ hoặc vào một tập tin, phải thực hiện theo trình tự:

- Mở một luồng đến tập tin tương ứng.
- Thực hiện đọc hoặc viết.
- Đóng luồng.

Có hai loại luồng - luồng văn bản và luồng nhị phân.

5.2| PHÂN LOẠI TẬP TIN

Có 2 loại tập tin: nhị phân và văn bản

Các lớp để làm việc với các luồng nhị phân là: FileStream, BinaryReader và BinaryWriter

Các lớp chính để làm việc với các luồng văn bản là: TextReader và TextWriter

5.3| FILESTREAM CLASS

Lớp **FileStream** cung cấp các phương thức khác nhau để đọc và ghi từ file nhị phân (đọc / ghi một byte và chuỗi byte), kiểm tra số byte có sẵn và phương thức để đóng luồng. Để có được đối tượng của lớp FileStream bằng cách gọi là hàm tạo với tham số - tên file

Để sử dụng lớp FileStream, ta cần khai báo thư viện **using System.IO**

Thao tác trên tập tin luôn gồm bốn bước sau:

Bước 1 : Khai báo Đường dẫn file (filePath). Kiểm tra đường dẫn file có tồn tại hay không bằng phương thức File.Exists(filePath).

Phương thức trả về true khi đường dẫn tồn tại trong hệ thống, ngược lại trả về false.

Bước 2 : Khai báo, khởi tạo đối tượng, cài đặt các mode

Bước 3 : Thao tác đọc hoặc ghi trên tập tin

Bước 4 : Đóng tập tin

Cú pháp Khai báo, khởi tạo đối tượng File:

Cú pháp khởi tạo:

```
FileStream <object_name> = new FileStream(
                                    <file_path>,
                                    <FileMode Enumerator>,
                                    <FileAccess Enumerator>,
                                    [<FileShare Enumerator>]
);
```

Trong đó:

<object_path> **:** Đường dẫn của đối tượng tập tin được tạo
<file_name> : Đường dẫn và tên tập tin.
<FileMode > : Chế độ làm việc của file sau khi mở, bao gồm các giá trị sau:

Mode	Công dụng
Append	Nếu file đang tồn tại, mở và đặt con trỏ cuối file, nếu chưa có tạo file mới
Create	Tạo file mới, nếu file đang tồn tại sẽ ghi đè nội dung cũ
CreateNew	Tạo một file mới và nếu file đã tồn tại thì ném IOException
Open	Mở file đang tồn tại
OpenOrCreate	Mở file đang tồn tại và nếu file không tồn tại thì tạo file mới
Truncate	Mở một file hiện có và cắt tất cả các dữ liệu được lưu trữ. Vì vậy, kích thước tập tin = 0

<**FileAccess** > : Qui định chế độ truy cập file Read, ReadWrite hoặc Write .

Mode	Giá trị	Công dụng
Read	1	Mở file để đọc dữ liệu
Write	2	Mở file để ghi dữ liệu
ReadWrite	3	Mở file vừa đọc vừa ghi

<FileShare> **:** Qui định chế độ chia sẻ. Mode FileShare cần khi hệ thống có nhiều tiến trình chạy cùng lúc để tránh gây deadlock (tắt ghẽn). Có thể bỏ qua mode này nếu không có các tiến trình song song.

Mode	Giá trị	Công dụng
None	0	Không chia sẻ tập tin hiện tại
Read	1	Chia sẻ chỉ đọc. Không được phép mở file để đọc nếu cờ này không được bật

Write	2	Chia sẻ cho phép mở tập tin để ghi Không được phép mở file để đọc nếu cờ này không được bật
ReadWrite	3	Chia sẻ cho phép mở tập tin để ghi hoặc đọc. Không được phép mở file để đọc hoặc ghi nếu cờ này không được bật
Delete	4	Chia sẻ cho phép xóa tập tin
Inherible	16	Cho phép file truyền tính kế thừa cho tiến trình con. Không hỗ trợ trong Win32.

Bước 2: Thực hiện thao tác trên tập tin

- ***Thao tác ghi file:***
  - ***Ghi nguyên khối bytes vào file stream***

```
Write(byte[] array, int offset, int count);
```

Trong đó:

Aray: tên chuỗi byte nguồn

Offset : vị trí byte bắt đầu sao chép trong chuỗi

Count : Số lượng byte tối đa ghi vào file

  - ***Ghi một byte vào vị trí hiện hành trong file stream***

```
- WiteByte(byte value)
```

Trong đó:

Value : là giá trị byte muốn ghi

- ***Thao tác đọc file***
  - ***Đọc nguyên khối bytes vào file stream***

```
Read(byte[] array, int offset, int count);
```

Công dụng: Nội dung trong mảng Byte từ vị trí offset đến vị trí (offset + count -1) sẽ được đọc vào file stream. Hàm trả về tổng số lượng byte đã đọc kiểu Int32.

Trong đó

- ✓ Aray: tên chuỗi byte đọc
- ✓ Offset : vị trí byte bắt đầu sao chép trong chuỗi
- ✓ Count : Số lượng byte tối đa ghi vào file

- ***Đọc một byte vào file stream***

ReadByte();

Công dụng: Phương thức trả về byte được đọc. Nếu gặp cuối file thì trả về -1

- ***Đọc từ buffer file stream***

ReadBlock(char[] buffer, int index, int count)

Công dụng : Đọc tất cả nội dung từ stream và ghi dữ liệu vào buffer bắt đầu từ vị trí index và đọc count ký tự.

Aray: tên chuỗi byte đích

Offset : vị trí byte bắt đầu sao chép trong chuỗi

Count : Số lượng byte tối đa ghi vào file

- ***Đọc một dòng từ stream***

ReadLine()

Công dụng : Đọc một dòng ký tự từ stream và trả về một chuỗi

- ***Đọc từ một vị trí đến cuối file***

ReadToEnd()

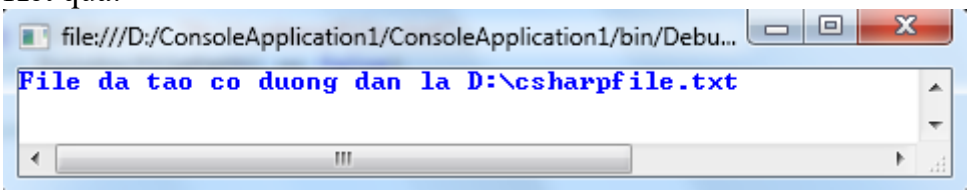
Công dụng: Đọc từ một vị trí đến hết stream

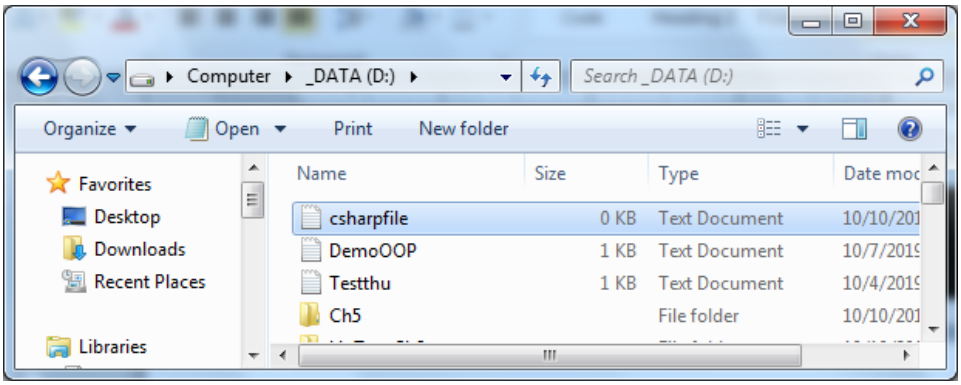
Đóng file.

Ví dụ1 : Tạo file trắng

```
using System;
using System.IO;

namespace Chuong5_File
{
    class Ch5_VD1
    {
        static void Main(string[] args)
        {
            string filePath = "D:\\csharpfile.txt";
            //Kiểm tra đường dẫn có tồn tại hay không
            if(File.Exists(filePath) == false)
            {
                Console.WriteLine("Khong ton tai thu muc!");
            }
            else
            {
                FileStream fs = new FileStream(filePath,
                                    FileMode.Create);
                fs.Close();
                Console.Write("File da tao co duong dan la
                                    {0}", filePath);
            }

            Console.ReadKey();
        }
    }
}
```

Kết quả:





Ví dụ 2: Ghi file

```
using System;
using System.IO;
using System.Text;

namespace Chuong5_File
{
    class Ch5_VD1
    {
        static void Main(string[] args)
        {
            //Khai báo đường dẫn file
            string filePath = "D:\\csharpFile.txt";
            if(File.Exists(filePath) == false)
            {
                Console.WriteLine("Duong dan khong
                    ton tai!");
            }else
            {
                //Khởi tạo đối tượng file
                FileStream fs = new FileStream(filePath,
                                    FileMode.Append);

                //Luu noi dung vao file
                byte[] bData = Encoding.UTF8.GetBytes("Cao
                                Đẳng Công Nghệ Thủ Đức");
               //Ghi noi dung vao file
                fs.Write(bData, 0, bData.Length);
                Console.WriteLine("Ghi file thanh cong!");

                //dong file
                fs.Close();
            }


            Console.ReadKey();
        }
    }
}
```

Kết quả:

```
Luu file thanh cong
```

Nội dung file `csharpFile.txt`

```
Cao Đẳng Công Nghệ Thủ Đức
```

Ví dụ 3: Đọc dữ liệu từ file

```
using System;
using System.IO;
using System.Text;

namespace Chuong5_File
{
    class Ch5_VD1
    {
        static void Main(string[] args)
        {
            string data;

            //khoi tao doi tuong fileStream de doc file
            string fileName = "D:\\csharpFile.txt";
            if (File.Exists(filePath) == false)
            {
                Console.WriteLine("Duong dan file khong ton
tai!");
            }
            FileStream fsSource = new FileStream(fileName,
FileMode.Open, FileAccess.Read);

            //doc noi dung tu file xuat ra man hinh, dong file
            using (StreamReader sr = new
StreamReader(fsSource))
            {
                data = sr.ReadToEnd();
            }
            Console.WriteLine(data);
            Console.ReadLine();
        }
    }
}
```

Kết quả:

```
Xin chao!
```

5.4| BINARY STREAMS

5.5| BINARYWRITER

Lớp BinaryWriter cho phép ghi các kiểu nguyên thủy và các giá trị nhị phân theo một mã hóa cụ thể vào một luồng.

Lớp BinaryWriter sử dụng phương thức **`Write(…)`** để ghi lại mọi kiểu dữ liệu cơ bản - số nguyên, ký tự, Booleans, mảng, chuỗi.

Nếu không cung cấp các loại mã hóa trong khi tạo đối tượng thì mã hóa mặc định UTF-8 sẽ được sử dụng.

Để ghi dữ liệu lên file thực hiện các bước sau:

- **Mở file(gọi hàm tạo)**

```
BinaryWriter binWriter = new BinaryWriter(
                File.Open(fileName, FileMode, FileAccess));
```

- **Ghi dữ liệu**

```
binWriter.Write(“content”)”;
```

- **Đóng file : Sử dụng các phương thức Close**

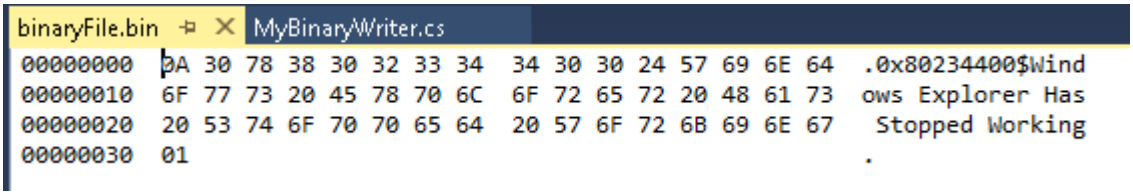
```
binWriter.Close();
```

```
using System;

using System.Text;

using System.IO;

namespace Chuong5
{
        class MyBinaryWriter
        {
            static void Main(string[] args)

      {

        //Tao doi tuong BinaryWriter de ghi noi
          dung len file

        string fileName = "binaryFile.bin";
                BinaryWriter writer = new
         BinaryWriter(File.Open(fileName,
         FileMode.CreateNew, FileAccess.Write));

        //ghi noi dung len file

        writer.Write("0x80234400");
                writer.Write("Windows Explorer Has
                                        Stopped Working");
                writer.Write(true);
                //Dong file
                write.Close();

          //ket thuc

         Console.WriteLine("Binary FileCreated!");

          Console.ReadKey();

      }

   }

}
```

Kết quả

```
Binary File Created!
```

Khi mở file trong Visual Studio ta được



5.6| BINARYREADER

BinaryReader cho phép đọc các kiểu dữ liệu cơ bản và các giá trị nhị phân được ghi lại bằng BinaryWriter. Các phương thức chính cho phép chúng ta đọc một ký tự, một mảng các ký tự, số nguyên, dấu phẩy động, v.v. Giống như hai lớp trước, có thể vào đối tượng của lớp đó bằng cách gọi hàm tạo của nó.

Để đọc dữ liệu từ file nhị phân thực hiện các bước sau:

➢ *Mở file(gọi hàm tạo)*

```
BinaryReader reader = new BinaryReader(
                                File.Open(fileName, FileMode));
```

➢ *Đọc dữ liệu : Sử dụng các phương thức Read():*

Hàm	**Công dụng**
Read()	Đọc các ký tự từ stream
Read(Byte[], int pos, int num)	Đọc mảng số Byte từ stream bắt đầu từ vụ trí pos đọc num ký tự
Read(Char[], int pos, int num)	Đọc mảng số Char từ stream bắt đầu từ vụ trí pos đọc num ký tự

➢ Đóng file: Sử dụng phương thức Close()

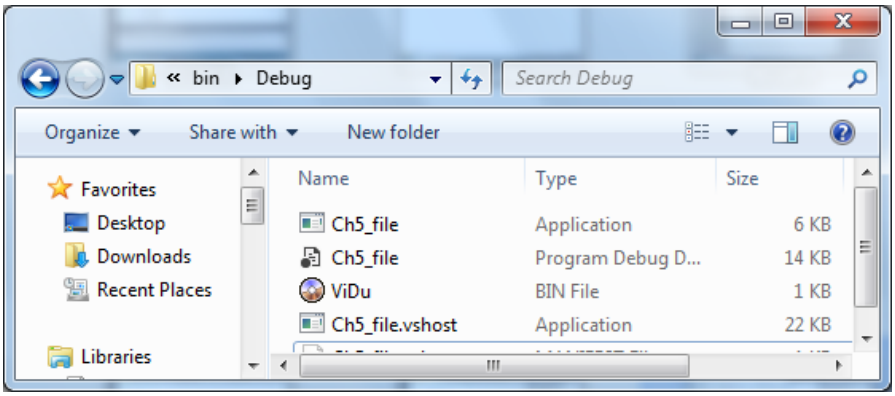
Ví dụ: Đọc, ghi file nhị phân

```
using System;
using System.Text;
using System.IO;
namespace Ch5_file
{
    class Binary_Write_Read
    {
        /* Minh hoa doc ghi file nhị phan
         * */
        static void Main(string[] args)
        {
            string fileName = "ViDu.bin";

            //tạo file
            WriteBinaryFile(fileName);
            Console.WriteLine("tao file thanh cong");
            //doc noi dung tu file
            Console.WriteLine("doc file: ");
            ReadBinaryFile(fileName);
            Console.ReadKey();
        }
        static void WriteBinaryFile(string fileName)
        {
            //tao file
            BinaryWriter writer = new
                        BinaryWriter(File.Open(fileName,
FileMode.Create,FileAccess.Write));
```

```
            //ghi noi dung len file

            writer.Write("0x80234400");

            writer.Write("Windows Explorer Has
                    Stopped Working");

            writer.Write(true);

            //dong file

            writer.Close();

        }

        static void ReadBinaryFile(string fileName)

        {

            //mo file

            BinaryReader reader = new
              BinaryReader(File.Open(fileName,FileMode.Open));

            //doc file

       Console.WriteLine(“Error Code : “
+reader.ReadString());

            Console.WriteLine("Message : " +
           reader.ReadString());

            Console.WriteLine("Restart Explorer : " +
           reader.ReadBoolean());

            //dong file

            reader.Close();

        }}}
```

Kết quả:

```
tao file thanh cong
doc file:
Error Code: 0x80234400
Message: Windows Explorer Has Stopped Working
Restart Explorer : True
```



5.7| TEXT STREAMS

Luồng văn bản rất giống với luồng nhị phân, nhưng chỉ hoạt động với dữ liệu văn bản hoặc đúng hơn là một chuỗi các ký tự (char) và chuỗi (string)

Các lớp chính để làm việc với các luồng văn bản trong .NET là TextReader và TextWriter. Các lớp này xác định chức năng cơ bản để đọc và viết nội dung trên luồng.

5.8| STREAMWRITTER

Lớp StreamWriter cho phép ghi các dữ liệu kiểu text vào một luồng.

Để ghi dữ liệu lên file text thực hiện các bước sau:

– **Mở file(gọi hàm tạo)**

```
StreamReader sReader = new StreamReader(string fileName);
```

– **Đọc dữ liệu: Sử dụng các phương thức Write():**

`Write()` viết một chuỗi vào luồng.

`WriteLine()` viết một dòng văn bản vào luồng.

– **Đóng file:** Sử dụng phương thức Close()

Ví dụ:

```
using System;
using System.Text;
using System.IO;

namespace Ch5_file
{
   class MyStreamWritter
   {
```

```
        static void Main(string[] args)
        {
            //tao file
            string file = "csharpfile.txt";
            StreamWriter sWriter = new StreamWriter(file);

            //ghi noi dung len file
            sWriter.WriteLine("day la vi du minh hoa");
            sWriter.WriteLine("su dung StreamWriter Class");

            //dong file
            sWriter.Close();

            //ket thuc
            Console.WriteLine("Luu file thanh cong!");
            Console.ReadKey();
        }
    }
}
```

Kết quả:

```
Luu file thanh cong
```

Nội dung file:

```
Day la vi du minh hoc
Su dung StreamWriter
```

5.9| STREAMREADER

Lớp StreamReader cho phép đọc các dữ liệu văn bản vào một luồng.

Lớp StreamReader cung cấp cách dễ nhất để đọc dữ liệu kiểu text, vì giống như đọc từ console.

Để đọc dữ liệu từ file text thực hiện các bước sau:

- Mở file(gọi hàm tạo)

```
StreamReader sReader = new StreamReader(string fileName);
```

- Đọc dữ liệu: Sử dụng các phương thức Read():

  ReadLine() đọc một dòng văn bản và trả về một chuỗi.

  ReadToEnd() đọc toàn bộ luồng đến cuối của nó và trả về một chuỗi.

– Đóng file: Sử dụng phương thức Close()

Ví dụ:

```
class MyStreamReader
{
    0 references
    static void Main(string[] args)
    {
        //tao file
        string file = "csharpfile.txt";
        StreamReader sReader = new StreamReader(file);

        //doc noi dung tu file
        Console.WriteLine(sReader.ReadToEnd());

        //dong file
        sReader.Close();

        //ket thuc
        Console.WriteLine("doc file thanh cong!");
        Console.ReadKey();
    }
}
```

Kết quả:

```
day la vi du minh hoa
sử StreamWriter Class
doc file thanh cong
```

5.10| TỰ ĐỘNG ĐÓNG LUỒNG SAU KHI LÀM VIỆC

Khi đã hoàn thành làm việc với đối tượng kiểu StreamReader, StreamWriter phải gọi **Close()** để đóng luồng. Tuy nhiên, rất thường xuyên lập trình viên quên gọi phương thức **Close()**, do đó tập tin bị ngăn chặn việc sử dụng tiếp theo. Ngoài ra có trường hợp ngoại lệ xảy ra, tập tin có thể bị bỏ ngỏ. Điều này gây ra rò rỉ tài nguyên và có thể dẫn đến các hiệu ứng như treo chương trình, chương trình sai và các lỗi lạ.

Cách chính xác để xử lý việc đóng tệp là từ khóa sử dụng:

using (<stream object>) { … }

Việc đọc, ghi tập tin thường phát sinh các lỗi trong quá trình thực hiện ví dụ không mở được file, đường dẫn sai,… .Để bắt được các lỗi sử dụng cú pháp

```
Try{Khối lệnh;
}Catch (IOexception){Xử lý}
```

5.11| BÀI TẬP

Các tập tin văn bản cung cấp giải pháp lý tưởng để đọc và ghi dữ liệu. Nếu chúng ta muốn nhập một số dữ liệu tự động (thay vì nhập bằng tay), chúng ta có thể đọc nó từ một tập tin văn bản.

Ví dụ viết chương trình đọc danh sách sinh viên từ file, xuất ra danh sách sinh viên, xuất danh sách các sinh viên được nhận học bổng(điểm trung bình >=7)

Gợi ý giải quyết bài toán:

+ Xây dựng struct SinhVien bao gồm: maSV(string), hoTenSV(string), diemTB(float)

+ Xây dựng các hàm
  - Main:
  - Tạo file DanhSach2.txt
  - Đọc dữ liệu từ file để tạo mảng sinh viên:
  - Xuất danh sách lớp

BÀI TẬP THỰC HÀNH SỐ 5

IV. Thông tin chung:

- Mã số bài tập : HW4-KTLT2
- Hình thức nộp bài : Nộp qua Moodle môn học
- Thời hạn nộp bài : … / … / ……
- Nội dung : Chương 5: File

Chuẩn đầu ra cần đạt:

L.O.2	Sử dụng kiểu cấu trúc và tập tin để giải quyết một số bào toán quản lý theo yêu cầu.
L.O.4	Làm bài tập theo yêu cầu của Giảng viên và nộp bài đúng quy định.

1. Viết chương trình tạo file MangNguyen.txt bao gồm các thông tin:

- Dòng 1: số phần tử mảng
- Từ dòng 2 trở đi: giá trị các phần tử mảng.

2. Viết chương trình thực hiện:

- Đọc file MangNguyen.txt, tạo mảng các số nguyên
- Xuất tổng các phần tử ở vị trí chẵn của mảng
- Xuất tổng các số nguyên tố có trong mảng

3. Viết chương trình tạo file HangHoa.txt bao gồm các thông tin:

- Dòng 1: số phần tử mảng
- Từ dòng 2 trở đi: giá trị các phần tử mảng.
- Mỗi phần tử mảng là thông tin hàng hóa bao gồm: Mã hàng, Tên hàng, Số lượng bán, đơn giá.
- Mỗi mặt hàng có thể được bán nhiều lần.

4. Viết chương trình đọc nội dung từ file HangHoa.txt

Xuất ra màn hình theo mẫu:

Ma Hang | Ten Hang | So luong ban | Don gia | Thanh Tien

- Trong đó **Thanh Tien = So Luong Ban * Don Gia**Xuất ra màn hình theo mẫu:

Ma Hang	**Ten Hang**	**So luong ban**	**Don gia**	**Thanh Tien**
001	Bàn phím	1000	20$	200.000$
….				
			Tong thanh tien: …….	

- Lưu nội dung trên thành file có tên HangHoa_MaHang.txt(mỗi mã hàng được lưu thành 1 file riêng biệt)

- Ví dụ: trong câu 3 tạo thông tin của 20 mã hàng từ 001 đến 020, câu 4 sẽ tạo ra 20 tập tin tên tương ứng HangHoa_001.txt, HangHoa_002.txt, …. HangHoa_020.txt

---Hết---