



TÀI LIỆU GIẢNG DẠY

KỸ THUẬT LẬP TRÌNH 2

**ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH
TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC
KHOA CÔNG NGHỆ THÔNG TIN**

GIÁO TRÌNH

HỌC PHẦN: KỸ THUẬT LẬP TRÌNH 2

NGÀNH: CÔNG NGHỆ THÔNG TIN

TRÌNH ĐỘ: CAO ĐẲNG

*Ban hành kèm theo Quyết định số 72/QĐ-CNTĐ-CN ngày 17 tháng 06 năm 2020
của Hiệu trưởng Trường Cao Đẳng Công Nghệ Thủ Đức.*

TP. Hồ Chí Minh, năm 2020

TUYÊN BỐ BẢN QUYỀN

Tài liệu này thuộc loại sách giáo trình nên các nguồn thông tin có thể được phép dùng nguyên bản hoặc trích dùng cho các mục đích về đào tạo và tham khảo.

Mọi mục đích khác mang tính lệch lạc hoặc sử dụng với mục đích kinh doanh thiếu lành mạnh sẽ bị nghiêm cấm.

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

Chương 1. MẢNG HAI CHIỀU.....	1
1.1 Khái niệm mảng nhiều chiều	2
1.2 Khai báo, khởi tạo mảng hai chiều	3
1.3 Khai báo, khởi tạo mảng chữ nhật	3
1.4 Khai báo, khởi tạo mảng răng cưa (Jagged Array)	4
1.5 Truy xuất mảng hai chiều	4
1.5.1 Đọc metadata của mảng	5
1.5.2 Truy xuất trực tiếp đến phần tử mảng	6
1.5.3 Truy xuất tuần tự các phần tử mảng	7
1.6 Sử dụng mảng hai chiều làm đối số cho hàm	11
1.7 Một số ứng dụng mảng hai chiều	13
1.8 Bài tập	17
Chương 2. STRING	Error! Bookmark not defined.
2.1 Khái niệm	22
2.2 Khai báo, khởi tạo	23
2.3 Định dạng chuỗi	23
2.4 Thuộc tính, Phương thức lớp String	26
2.5 Bài tập	29
Chương 3. DATETIME	Error! Bookmark not defined.
3.1 Khái niệm	33
3.2 Khai báo và khởi tạo	33
3.3 MinValue và MaxValue	34
3.4 Một số hàm thư viện DateTime	36
3.5 Các toán tử trên DateTime	43
3.6 Chuỗi định dạng DateTime	44
3.7 Chuyển đổi String sang DateTime	46
3.8 Bài tập	47
Chương 4. STRUCT.....	49
4.1 Khái niệm cấu trúc	50
4.2 Khai báo cấu trúc	50
4.3 Khai báo biến kiểu cấu trúc	51
4.4 Truy cập các thành phần trong cấu trúc	53
4.5 Phép toán gán cấu trúc	54
4.6 Sử dụng struct làm đối số cho hàm	55
4.7 Mảng struct	57
4.8 Bài tập	59
Chương 5. FILE.....	65
5.1 Khái niệm	66
5.2 Phân loại tập tin	67
5.3 FileStream class	68
5.3.1 Sử dụng StreamReader để Đọc file	70
5.3.2 Sử dụng StreamWriter để ghi file	75
5.4 Binary Streams	77

5.4.1	BinaryWriter	77
5.4.2	BinaryReader	79
5.5	Bài tập	81
5.6	Bài tập tổng hợp	84

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.

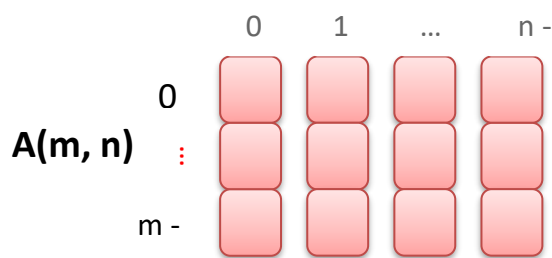
1.1| KHÁI NIỆM MẢNG NHIỀU CHIỀU

Mảng là một tập các dữ liệu. Các phần tử được đánh chỉ số (index) và được sắp xếp liên tiếp 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 có sẵn trong C# (built-in data types) hoặc kiểu do người dùng tự định nghĩa (user-defined data types). 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 hay đa chiều. C# hỗ trợ mảng có tối đa 32 chiều.

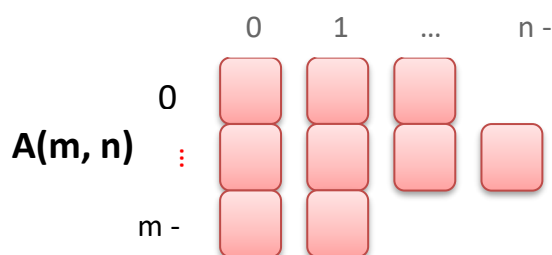
Mảng hai chiều được hình dung như một bảng chữ nhật, trong đó mỗi ô là một phần tử. Mảng ba chiều được hình dung nó như một khối hộp lớn, gồm nhiều hộp nhỏ (giống như khối rubic). Mỗi hộp nhỏ là một phần tử.

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



Mảng hai chiều (ma trận) là mảng 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.



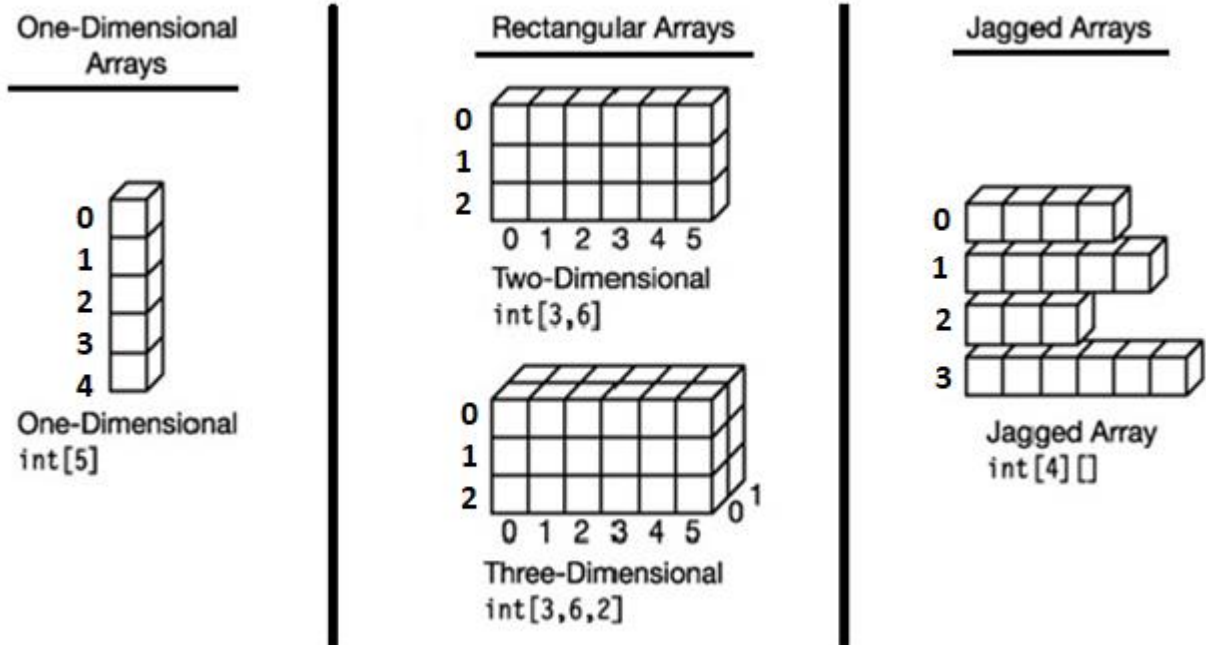


Figure 1: Sơ đồ hình khối của các loại mảng Mảng

1.2| KHAI BÁO, KHỞI TẠO MẢNG HAI CHIỀU

1.3| KHAI BÁO, KHỞI TẠO MẢNG CHỮ NHẬT

Cách 1: Khai báo mảng chữ nhật 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 mảng hai 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

1.4| KHAI BÁO, KHỞI TẠO MẢNG RĂNG CỬA (JAGGED ARRAY)

Khi mảng có số phần tử trên mỗi dòng không bằng nhau tạo thành một mảng răng cưa (Jagged Array). Cú pháp lệnh 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 }  
                };
```

1.5| TRUY XUẤT MẢNG HAI CHIỀU

1.6| ĐỌC METADATA CỦA MẢNG

Metadata là những thông tin về bản thân mảng, như số lượng phần tử, kiểu cơ sở, ... 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
Length / 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)	Trả về số phần tử trên dòng của mảng chữ nhật
GetLength(1)/GetLongLength(1)	Trả về số phần tử trên cột của mảng chữ nhật
ToString()	Trả về về chuỗi thông tin của mảng.

Ví dụ:

File: Program.cs

```
/*Ví dụ: Chương trình đọc meta data của mảng
*/
using System;
namespace VD_MangHaiChieu
{
    class Program
    {
        static void Main(string[] args)
        {
            //Khai báo, khởi tạo mảng
            int[,] arrM = { { 5, 2, 8, 9 },
                           { 1, 3, 0, 7 },
                           { 6, 8, 1, 0 } };

            //Đọc thông tin metadata của mảng
            Console.WriteLine("Số phần tử mảng : " +
                              arrM.Length); //số phần tử
            Console.WriteLine("Số chiều của mảng : " +
                              arrM.Rank); //Số chiều
            Console.WriteLine("Kiểu mảng : " +
                              arrM.GetType()); //Kiểu dữ liệu
            Console.WriteLine("Số dòng : " +
                              arrM.GetLength(0)); //Số dòng
            Console.WriteLine("Số cột : " +
                              arrM.GetLength(1)); //Số cột
            Console.WriteLine("ToString : " +
                              arrM.ToString()); //Chuỗi thông tin
        }
    }
}
```

Kết quả:

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

1.7| 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	...j	Cột c-1
Dòng 0	arr[0][0]	arr[0][1]	arr [0][2]	arr [0][3]		arr [0][c-1]
Dòng 1	arr [1][0]	arr [1][1]	arr [1][2]	arr [1][3]		arr [1][c-1]
Dòng 2	arr [2][0]	arr [2][1]	arr [2][2]	arr [2][3]	...	arr [2][c-1]
...i						
Dòng r-1	arr [-1][0]	arr[r-1][1]	arr[r-1][2]	arr[r-1][3]		arr [r-1][c-1]

Như vậy, Mảng arr có r dòng và c cột với mỗi phần tử trong mảng được định danh bởi một tên phần tử là arr[i][j], với arr là tên mảng, i là chỉ số dòng có các giá trị từ 0 đến r-1, j là chỉ số cột có các giá trị từ 0 đến c-1. Mỗi phần tử arr [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 ta dùng arr[i, j].

Ví dụ: Giả sử ta có mảng arr như sau

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

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

1.8| 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:

File: Program.cs

```
/*Ví dụ: Chương trình nhập, xuất mảng chữ nhật
*/
using System;
namespace VD_MangHaiChieu
{
    class Program
    {
        static void Main(string[] args)
        {
            //Khai báo số dòng, cột
            int maxR = 3;
            int maxC = 4;
            //Khai báo mảng hai chiều chữ nhật
            int[,] arrM = new int[maxR, maxC];
            //Nhập giá 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
                        arrM[i, j]);
                }
                Console.WriteLine();
            }
        }
    }
}
```

```

        //In mang hai chieu chu nhac
        int dem = 0;
        foreach (int e in arrM)
        {

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

        }
    }
}

```

Kết quả:

Nhapmang hai chieu

Nhap [0,0] = 2

Nhap [0,1] = 4

Nhap [0,2] = 5

Nhap [0,3] = 6

Nhap [1,0] = 7

Nhap [1,1] = 8

Nhap [1,2] = 9

Nhap [1,3] = 6

Nhap [2,0] = 5

Nhap [2,1] = 4

Nhap [2,2] = 1

Nhap [2,3] = 7

2 4 5 6

7 8 9 6

5 4 1 7

Press any key to continue . . .

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:

File: Program.cs

```
/*Ví dụ: Chương trình nhập, xuất mảng răng cưa
*/
using System;
namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            int maxR = 3; //số dòng
            int maxC; //số cột
            //Mảng Zigzag
            int[][] arrM = new int[maxR][];
            for (int i = 0; i < maxR; i++)
            {
                Console.Write("Nhập SPT cho dòng {0}", i);
                int.TryParse(Console.ReadLine(), out maxC);
                arrM[i] = new int[maxC];
                for (int j = 0; j < maxC; j++)
                {
                    Console.Write("Nhập [{0},{1}] = ", i, j);
                    int.TryParse(Console.ReadLine(), out
                        arrM[i][j]);
                }
                Console.WriteLine();
            }
            Console.ReadKey();
        }
    }
}
```

Kết quả:

```
Nhập số phần tử cho dòng 0: 2
Nhập [0,0] = 5
Nhập [0,1] = 4

Nhập số phần tử cho dòng 1: 3
Nhập [1,0] = 8
Nhập [1,1] = 7
Nhập [1,2] = 6

Nhập số phần tử cho dòng 2: 3
```

```
Nhap [2,0] = 5
Nhap [2,1] = 9
Nhap [2,2] = 1
Press any key to continue . . .
```

Ví dụ 3: Dùng vòng lặp for và hàm GetLength() để in mảng hai chiều

```
for (int i = 0; i < arrM.GetLength(0); i++)
{
    for (int j = 0; j < arrM.GetLength(1); j++)
    {
        Console.Write(arr1[i, j] + "\t");
    }
    Console.WriteLine();
}
```

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

```
//In mảng răng cưa
for (int i = 0; i < arrM.GetLength(0); i++)
{
    for (int j = 0; j < arrM[i].Length; j++)
    {
        Console.Write(arrM[i][j] + "\t");
    }
    Console.WriteLine();
}
```

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

```
//In mảng chữ nhật

foreach (int element in arr1)
{
    Console.Write("{0}\t", element.ToString());
    if (++dem % maxR == 0)
        Console.WriteLine();
}
```

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

```
//In mảng hai chiều răng cưa
foreach (int[] row in arr2)
{
    foreach (int ele in row)
    {
        Console.Write("{0}\t", ele.ToString());
    }
    Console.WriteLine();
}
```

1.9| 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 sau khi hàm thực hiện xong. Trường hợp mảng truyền vào chưa có giá trị thì phải sử dụng từ khóa **out** như các biến thông thường.

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

```
File: Program.cs
/*Ví dụ: Chương trình nhập, xuất mảng răng cưa
*/
using System;
namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            //Khai báo mảng Zigzag
            int[][] arrM;
            //Gọi hàm nhập, xuất mảng
            NhapMang(out arrM);
            XuatMang(arrM);
        }
        //Hàm nhập mảng răng cưa
        static void NhapMang(out int[][] arrM)
        {
            int maxR = 0;
            int maxC = 0;
            //Nhập số dòng
            Console.Write("Nhap so dong: ");
            maxR = int.Parse(Console.ReadLine());
            //Khai báo mảng
            arrM = new int[maxR][];
            //Nhập từng dòng
```



```

        for (int i = 0; i < maxR; i++)
        {
            Console.WriteLine("Nhap SPT cho dong {0}: ", i);
            int.TryParse(Console.ReadLine(), out maxC);
            arrM[i] = new int[maxC];
            for (int j = 0; j < maxC; j++)
            {
                Console.WriteLine("Nhap [{0},{1}] = ", i, j);
                arrM[i][j] = int.Parse(Console.ReadLine());
            }
            Console.WriteLine();
        }
    }

    //Xuất mảng rỗng của
    static void XuatMang(int[][] arrM)
    {
        //In mảng rỗng của
        for (int i = 0; i < arrM.GetLength(0); i++)
        {
            for (int j = 0; j < arrM[i].Length; j++)
            {
                Console.Write(arrM[i][j] + "\t");
            }
            Console.WriteLine();
        }
    }
}

```

Kết quả:

```

Nhap so dong: 3
Nhap so phan tu cho dong 0: 2
Nhap [0,0] = 7
Nhap [0,1] = 8

```

```

Nhap so phan tu cho dong 1: 1
Nhap [1,0] = 7

```

```

Nhap so phan tu cho dong 2: 3
Nhap [2,0] = 4
Nhap [2,1] = 5
Nhap [2,2] = 9

```

```

7      8
7

```

1.10| 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 quả 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 họa:

```
/*Ví dụ: Chương trình nhập, xuất mảng rằng của
*/
using System;
namespace ConsoleApplication11
{
    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 e in arrM)
```

```

    {
        Console.WriteLine("{0}\t", e.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.

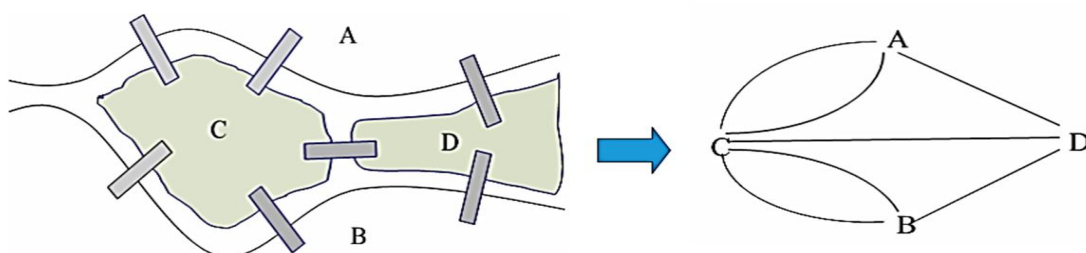
	X	O
X	O	
O		X

🚦 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 Königsberg 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 Sĩ, 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 bài toán về tìm đường bằng máy tính như: bài toán kiểm tra tính liên thông , bài toán tìm đường đi ngắn nhất, ...

1.11| 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

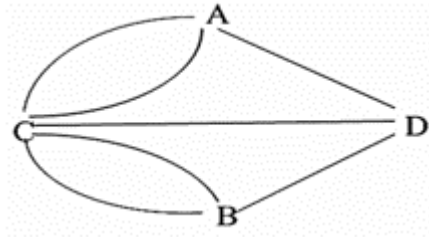
L.O.4 Làm bài tập theo yêu cầu của Giảng viên và nộp bài đúng quy định.

Yêu cầu:

- Sinh viên đọc Tài liệu Kỹ 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

- 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”.
- Viết đoạn chương trình thống kê số chỗ ngồi còn trống trong câu 1 như sau:
 - Tổng số chỗ ngồi còn trống trong rạp.
 - Số lượng ghế trống từng hàng.
 - Số lượng ghế trống từng dãy.
 - Số cặp ghế trống theo hàng
 - Tìm hàng có nhiều ghế trống nhất
 - Tìm hàng đã hết chỗ trống
 - Kiểm tra tất cả các ghế ở ngoài biên được đặt hết hay chưa.
- 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:

- 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.
- 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.
- Viết chương trình tính tổng các phần tử theo dòng (hoặc cột).
- 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 và kết thúc tại cùng một đỉnh trong đồ thị.

- 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.
- Viết chương trình khởi tạo giá trị các phần tử là ngẫu nhiên cho ma trận các số nguyên kích thước $m \times n$.

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:

- Viết hàm tính tích các phần tử trên mỗi cột của ma trận
- Viết hàm tính tích các phần tử nằm trên đường chéo chính, chéo phụ của ma trận vuông.
- Viết hàm tính tổng các phần tử là số nguyên tố có trong ma trận.
- Viết hàm tính giá trị trung bình của các phần tử là số chẵn trong ma trận.
- Viết hàm tìm phần tử lẻ lớn nhất trong ma trận
- Viết hàm đếm số lần xuất hiện của phần tử x trong ma trận, với x được nhập từ bàn phím.
- 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.
- Viết hàm sắp xếp các dòng trên ma trận theo thứ tự tăng dần.

2.

MỘT SỐ KIỂU DỮ LIỆU XÂY DỰNG SẴN TRONG C#

Chương này nhằm giới thiệu cho sinh viên sử dụng kiểu dữ liệu xây dựng sẵn (built-in data types) trong C# như String, DateTime để xây dựng các chương trình vừa và nhỏ theo yêu cầu.

C# chia thành hai tập hợp kiểu dữ liệu chính:

- Built-in data types: là các kiểu dữ liệu mà ngôn ngữ cung cấp cho người lập trình.
- User-define data types: là kiểu dữ liệu do người dùng tự tạo ra.

C# phân tập hợp kiểu dữ liệu này thành hai loại:

- Kiểu dữ liệu giá trị (value types): một biến (variable) khi được khai báo với kiểu tham trị thì vùng nhớ của nó sẽ chứa giá trị của dữ liệu. Các kiểu dữ liệu tham trị có sẵn như: bool, byte, char, decimal, double, enum, float, int, long, sbyte, short, struct, Thư viện DateTime được định nghĩa theo kiểu struct nên là một kiểu giá trị được xây dựng sẵn.
- Kiểu dữ liệu tham chiếu (reference types): một biến khi được khai báo kiểu dữ liệu tham chiếu thì vùng nhớ của nó được dùng để lưu địa chỉ tham chiếu tới vùng nhớ chứa giá trị thật sự của biến. Một số kiểu built-in reference types như: string, object, delegate, dynamic.

Trong chương này chúng ta sẽ tìm hiểu kiểu dữ liệu được xây dựng sẵn string và DateTime

2.1| STRING

2.1.1| 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 từ 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 Length 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).

2.1.2| 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\\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 tạo 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

// Khai báo chuỗi đường dẫn thư mục
string sPath = @"c:\Program Files\Visual Studio 8.0";

// Khai báo chuỗi nội suy với $ (interpolate string)
string str = $"Hello, {name}! Today is {DateTime.Now}";
```

2.1.3| 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
\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 tùy 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.

2.1.4| 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:

```
/*Ví dụ: Chương trình in chiều dài chuỗi
*/
using System;
namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            String MyStr = "Cao Dang";
            Console.WriteLine("Chiều dài chuỗi : " +
                             MyStr.Length);
            for (int i = 0; i <= MyStr.Length - 1; i++)
            {
                Console.Write("{0} ", MyStr[i]);
            }
            Console.ReadKey();
        }
    }
}
```

Kết quả:

```
Chiều dài chuỗi : 8

C a o   D a n g
```

Một số phương thức xử lý chuỗi:

Phương thức	Ý nghĩa
int Compare (string str1, string str2) Vd: String.Compare(str1 , str2)	So sánh hai chuỗi. Trả về: -1 : str1 < str2; 0 : str1 = str2; 1. : str1 > str2
string Concat (string str0, string str1, ...) Vd: String .Concate (str1 , str2);	Nối chuỗi str2 vào cuối chuỗi str1, trả về chuỗi mới.
bool Contains (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) String.Equals(str1, str2)	Trả về true nếu str1 và str2 cùng giá trị, ngược lại trả về false
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 IndexOfAny (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 chuỗi str null hoặc empty
int LastIndexOf (string str) Vd: str1.lastIndexOf("u")	Trả về chỉ mục cho sự xuất hiện cuối cùng của một chuỗi str đã cho bên trong đối tượng hiện tại

	ại
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:

```

File: Program.cs
/*Ví dụ: Chương trình minh họa sử dụng hàm xử lý chuỗi
*/
using System;
using static System.Console;

namespace ConsoleApplication11
{
    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

```

2.1.5| 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.

11. Viết chương trình cho phép người dùng nhập vào một chuỗi họ tên. Chương trình tạo username và password tự động cho người dùng bằng cách ghép phần tên và phần họ, password mặc định được tạo ra bằng cách ghép các ký tự đầu tiên của chuỗi họ tên và viết thường ghép với một số có 6 chữ số ngẫu nhiên.
Ví dụ: Họ tên: Nguyen Van Anh, username: AnhNguyen, password: nva261782
12. Viết chương trình cho phép người dùng nhập vào một danh sách. Chương trình thực hiện sắp xếp danh sách vừa nhập theo thứ tự alphabet rồi in ra màn hình. Biết họ tên hợp lệ là chuỗi chỉ bao gồm các ký tự chữ, 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.

2.2| DATETIME

2.2.1| 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.

2.2.2| 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:

```
File: Program.cs
/*Ví dụ: Chương trình minh họa sử dụng DateTime
*/
using System;

namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime dateTime = new DateTime();
            Console.WriteLine(dateTime); // 1/1/0001 12:00:00 AM
            // 2015 is year, 12 is month, 25 is day
            DateTime date1 = new DateTime(2015, 12, 25);
            Console.WriteLine(date1.ToString());
            // 2015-year, 12-month, 25-day, 10-hour, 30-minute, 50-second
            DateTime date2 = new DateTime(2012, 12, 25, 10, 30, 50);
            Console.WriteLine(date1.ToString());
        }
    }
}
```

Kết quả:

```
1/1/0001 12:00:00 AM
```

```
12/25/2015 12:00:00 AM
12/25/2015 12:00:00 AM
```

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

File: Program.cs

```
/*Ví dụ: Chương trình minh họa sử dụng DateTime.Now */
using System;

namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime dateTime = DateTime.Now;
            Console.WriteLine("Today is: {0}", dateTime);
        }
    }
}
```

Kết quả:

```
Today is: 6/20/2020 6:04:29 AM

Press any key to continue . . .
```

2.2.3| 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`.

`MaxValue`: là giá trị lớn nhất của `DateTime`.

Ví dụ 1: Chương trình so sánh giá trị MinValue và giá trị khởi tạo mặc định

File: Program.cs

```
/*Ví dụ: Chương trình minh họa sử dụng DateTime.MinValue
*/
using System;

namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            // Khai báo, khởi tạo
            DateTime date1 = new DateTime();
            Console.Write(date1);
            //Kiểm tra giá trị
            if (date1.Equals(DateTime.MinValue))
                Console.WriteLine("(Equals Date.MinValue)");
        }
    }
}
```

Kết quả:

```
1/1/0001 12:00:00 AM(Equals Date.MinValue)
```

Ví dụ 2: Chương trình sử dụng giá trị MaxValue

File: Program.cs

```
/*Ví dụ: Chương trình minh họa sử dụng DateTime.MaxValue
*/
using System;

namespace ConsoleApplication11
{
    class Program
    {
        static void Main(string[] args)
        {
            // Khai báo, khởi tạo
            DateTime myDate = new DateTime();
            myDate = DateTime.MaxValue;
        }
    }
}
```



```
        Console.WriteLine(myDate) ;
    }
}
```

Kết quả:

12/31/9999 11:59:59 PM

2.2.4| 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:

File: Program.cs

```

/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime myDate = new DateTime(2015, 12, 25, 10,30, 45);
            int year = myDate.Year; // 2015
            int month = myDate.Month; //12
            int day = myDate.Day; // 25
            int hour = myDate.Hour; // 10
            int minute = myDate.Minute; // 30
            int second = myDate.Second; // 45
            int weekDay = (int)myDate.DayOfWeek; // 5 due to Friday
        }
    }
}

```

Kết quả:

```

/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

```

```

namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime myDate = new DateTime(2015,12,25,10,30,45);
            int year = myDate.Year; // 2015
            int month = myDate.Month; //12
            int day = myDate.Day; // 25
            int hour = myDate.Hour; // 10
            int minute = myDate.Minute; // 30
            int second = myDate.Second; // 45
            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.

Cú pháp:

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

Ví dụ:

File: Program.cs

```

/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime myDate = new DateTime(2015, 12, 25, 10,30, 45);
            DateTime endDate = myDate.AddDays(36);
        }
    }
}

```

```

        Console.WriteLine("Start Date: {0}", myDate);
        Console.WriteLine("36 days after: {0}", endDate);
    }
}

```

Kết quả:

```

Start Date: 12/25/2015 10:30:45 AM
36 days after: 1/30/2016 10:30:45 AM

```

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.

Cú pháp

```

public DateTime AddMonths (int months);

```

Ví dụ:

File: Program.cs

```

/*Ví dụ: Chương trình minh họa các hàm thư viện DateTime
*/
using System;
namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime myDate = new DateTime(2015, 12, 25, 10, 30, 45);
            Console.WriteLine("Start Date: {0}", myDate);
            Console.WriteLine("12 Month from Start date: {0}",
                myDate.AddMonths(12));
        }
    }
}

```

Kết quả:

```

Start Date: 12/25/2015 10:30:45 AM

```

12 Month from Start date: 12/25/2016 10:30:45 AM

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);
```

Ví dụ:

File: Program.cs

```
/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime myDate = new DateTime(2015,12,25,10,30,45);
            DateTime endDate = myDate.AddYears(4);
            Console.WriteLine("Start Date: {0}", myDate);
            Console.WriteLine("4 years after: {0}", endDate);
        }
    }
}
```

Kết quả:

```
Start Date: 12/25/2015 10:30:45 AM
4 years after: 12/25/2019 10:30:45 AM
```

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);
```

Ví dụ:

File: Program.cs

```
/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime myDate = new DateTime(2015,12,25,10,30,45);
            Console.WriteLine("Start Date: {0}", myDate);
            Console.WriteLine("48 hours after {0}",
                             myDate.AddHours(48));
        }
    }
}
```

Kết quả:

```
Start Date: 12/25/2015 10:30:45 AM
48 hours after: 12/27/2015 10:30:45 AM
```

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);
```

Ví dụ:

File: Program.cs

```
/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;
```

```

namespace VD_DateTime
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime dateValue = new DateTime(2013, 9, 15, 12, 0, 0);
            double[] minutes = { .01667, .08333, .16667, .25 };
            foreach (double min in minutes)
            {
                Console.WriteLine("{0} + {1} minute(s) = {2}",
                    dateValue, min, dateValue.AddMinutes(min));
            }
        }
    }
}

```

Kết quả:

```

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);

```

Ví dụ:

File: Program.cs

```

/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

namespace ConsoleApplication11
{
    class Program

```

```

{
    static void Main(string[] args)
    {
        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);
    }
}

```

Kết quả:

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)			

Phương thức Ticks(): Lấy ra số lượng "tick" được đại diện bởi đối tượng kiểu DateTime.

(1 phút = 600 triệu tick)

2.2.5| CÁC TOÁN TỬ TRÊN DATETIME

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

File: Program.cs

```

/*Ví dụ: Chương trình minh họa sử dụng
 * các hàm thư viện lớp DateTime
 */
using System;

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

```



```
// It is 10th December 2015
DateTime dtObject = new DateTime(2015, 12, 10);
// TimeSpan object with 15 days, 10 hours, 5 min and 1
sec.
TimeSpan timeSpan = new TimeSpan(15, 10, 5, 1);

DateTime addResult = dtObject + timeSpan;
// 12/25/2015 10:05:01 AM
DateTime subtractResult = dtObject - timeSpan;

Console.WriteLine("Time value: " + dtObject);
Console.WriteLine("Time span(15, 10, 5, 1) = " +
    timeSpan);
Console.WriteLine("Add result: " + addResult);
Console.WriteLine("Subtract results: " +
    subtractResult);
bool areEqual = (addResult == (dtObject + timeSpan));
Console.WriteLine("are equal value is: " + areEqual);
}
}
}
```

Kết quả:

```
Time value: 12/10/2015 12:00:00 AM
Time span(15, 10, 5, 1) = 15.10:05:01
Add result: 12/25/2015 10:05:01 AM
Subtract results: 11/24/2015 1:54:59 PM
areEqual value is: True
```

2.2.6| 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 (local) 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:

```
DateTime tempDate = new DateTime(2015, 12, 08);

// creating date object with 8th December 2015

Console.WriteLine(tempDate.ToString("MMMM dd, yyyy"));

//December 08, 2105.
```

2.2.7| 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: Đoạn chương trình minh họa các hàm chuyển đổi kiểu dữ liệu từ String sang DateTime

File: Program.cs

```
/*Ví dụ chuyển string sang DateTime
*/
using System;

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

```

string strDate = "20/06/2020";
//Chuyển string sang DateTime
DateTime myDateConver = Convert.ToDateTime(strDate);
DateTime myDateParse = DateTime.Parse(strDate);
DateTime myDateParseExact = DateTime.ParseExact(strDate,
    "dd/mm/yyyy", null);

DateTime myDateTryParse;
DateTime.TryParse(strDate, out myDateTryParse);
//In kết quả
Console.WriteLine("Dung Convert" + myDateConver);
Console.WriteLine("Dung Parse" + myDateParse);
Console.WriteLine("Dung TryParseExact" +
    myDateParseExact);
Console.WriteLine("Dung TryParseExact"+myDateTryParse);
    }
}
}

```

Kết quả:

```

Dung Convert20/06/2020 12:00:00 SA
Dung Parse20/06/2020 12:00:00 SA
Dung TryParseExact20/01/2020 12:06:00 SA
Dung TryParseExact20/06/2020 12:00:00 SA

```

2.2.8| 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 cá nhân như 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 bao gồm các thông tin cá nhân
- b. Liệt kê họ tên của những nhân viên sinh trong quý 1.

3.

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.

3.1| KHÁI NIỆM CẤU TRÚC

Trong C#, kiểu dữ liệu cấu trúc là kiểu giá trị (value type). Nó giúp cho người dùng có thể tạo một biến bao gồm nhiều phần tử dữ liệu có liên quan, thuộc các kiểu dữ liệu 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.

3.2| 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.

3.3| 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:

```
/* Ví dụ khai báo và sử dụng struct  
*/  
using System;  
  
struct Books  
{  
    //Khai báo các trường giá trị  
    public string title;  
    public string author;  
    public string subject;  
}
```



```

public int book_id;

//hàm khởi tạo struct
public Books(string t, string a, string s, int id)
{
    title = t;
    author = a;
    subject = s;
    book_id = id;
}
//Hàm trả về chuỗi thông tin của struct
public string Print()
{
    string s = $"{book_id} - {title} - {author} - {subject}";
    return s;
}

};

public class Program
{
    public static void Main(string[] args)
    {
        // Khai báo và khởi tạo biến book1 sử dụng toán tử new và constructor
        Books book1 = new Books("Telecom Billing",
            "Zara Ali", "Telecom Billing Tutorial", 6495700);

        // Khai báo và khởi tạo biến book2 không sử dụng new
        Books book2;

        // string str = book2.title; //lỗi vì book2 chưa khởi tạo
        // Khởi tạo Book1
        book2.title = "C Programming";
        book2.author = "Nuha Ali";
        book2.subject = "C Programming Tutorial";
        book2.book_id = 6495407;

        //In thông tin
        Console.WriteLine(book1.Print());
        Console.WriteLine(book2.Print());
        Console.ReadKey();
    }
}

```

Kết quả:

```
6495700 - Telecom Billing - Zara Ali - Telecom Billing Tutorial
6495407 - C Programming - Nuha Ali - C Programming Tutorial

Press any key to continue . . .
```

Lưu ý: ngoài thành phần data, trong struct người dùng có thể định nghĩa thêm hàm khởi tạo và các phương thức xử lý, tính toán như là một thành phần của struct. Ngoài ra, thành viên struct có thể sử dụng một struct khác làm kiểu dữ liệu. Trường hợp này ta gọi là các struct lồng nhau.

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;
}
```

3.4| 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 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.WriteLine("\nstruct long nhau trong C#:\n");
        Console.WriteLine("-----\n");
        NhanVien[] emp = new NhanVien[total];
        for (int i = 0; i < total; i++)
        {
            Console.WriteLine("Ten nhan vien: ");
            string nm = Console.ReadLine();
            emp[i].eName = nm;
            Console.WriteLine("Nhap ngay sinh: ");
            dd = Convert.ToInt32(Console.ReadLine());
            emp[i].Date.Day = dd;
            Console.WriteLine("Nhap thang sinh: ");
            mm = Convert.ToInt32(Console.ReadLine());
            emp[i].Date.Month = mm;
            Console.WriteLine("Nhap nam sinh: ");
            yy = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            emp[i].Date.Year = yy;
        }
        Console.ReadKey();
    }
}

```

Kết quả:

```

struct long nhau trong C#:
-----

Ten nhan vien: Cao Dang
Nhap ngay sinh: 22

```

```
Nhap thang sinh: 12
Nhap nam sinh: 2000
Ten nhan vien: Thu Duc
Nhap ngay sinh: 1
Nhap thang sinh: 5
Nhap nam sinh: 1999
Press any key to continue . . .
```

3.5| 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
```

3.6| 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:

```
/* Ví dụ sử dụng struct
 */
using System;

//Khai báo struct
struct SinhVien
{
    //data
    public int maSo;
    public string hoTen;
```

```

    public double diemToan;
    public double diemLy;
    public double diemVan;
}

namespace MyNamespace
{
    public class MyStruct
    {
        static void Main(string[] args)
        {
            SinhVien sV1 = new SinhVien();
            Console.WriteLine(" Nhập thông 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(" Thông tin sinh vien vua nhap la:
");

            XuatThongTinSinhVien(sV1);
            Console.WriteLine(" Diem TB của sinh vien la: " +
DiemTBSinhVien(sV1));

            Console.ReadLine();
        }
        //Hàm nhập thông tin
        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());
        }
        //Hàm xuất thông tin
        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);
        }
        //Hàm tính điểm trung bình
        static double DiemTBSinhVien(SinhVien SV)

```

```

        {
            return (SV.diemToan + SV.diemLy + SV.diemVan) / 3;
        }
    }
}

```

Kết quả:

```

Nhap thong tin sinh vien:
Ma so: 1
Ho ten: Cao Dang
Diem toan: 8
Diem ly: 6
Diem van: 9
*****
Thong tin sinh vien vua nhap la:
Ma so: 1
Ho ten: Cao Dang
Diem toan: 8
Diem ly: 6
Diem van: 9
Diem TB cua sinh vien la: 7.666666666666667

```

3.7| 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.

```

/* Ví dụ sử dụng struct
*/
using System;

//Khai báo struct
struct SinhVien
{
    //data
    public int maSo;
    public string hoTen;
    public double diemToan;
    public double diemLy;
    public double diemVan;
}

namespace MyNamespace
{
    public class MyStruct
    {
        static void Main(string[] args)
        {
            //Nhập số phần tử
            int soPT = 0;
            Console.WriteLine("Nhap so phan tu: ");
            int.TryParse(Console.ReadLine(), out soPT);
            //Tạo mảng
            SinhVien[] arrSV1;
            arrSV1 = NhapMang(soPT);
            //In mảng
            XuatMang(arrSV1);
            Console.ReadLine();
        }
        //Hàm nhập thông tin
        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());
    }
    //Hàm xuất thông tin
    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);
    }
    //Hàm tính điểm trung bình
    static double DiemTBSinhVien(SinhVien SV)
    {
        return (SV.diemToan + SV.diemLy + SV.diemVan) / 3;
    }

    //Nhập mảng Sinh vien
    static SinhVien[] NhapMang(int soPT)
    {
        SinhVien[] arrSV = new SinhVien[soPT];
        for(int i = 0; i < soPT; i++)
        {
            Console.WriteLine("Nhập thông tin SV thu {0} :",
                               i + 1);
            NhapThongTinSinhVien(out arrSV[i]);
        }
        return arrSV;
    }

    //Xuất mảng Sinh vien
    static void XuatMang(SinhVien[] arrSV)
    {
        for (int i = 0; i < arrSV.Length; i++)
        {
            XuatThongTinSinhVien(arrSV[i]);
        }
    }
}

```

Kết quả:


```
Nhap so phan tu:
2
Nhap thong tin SV thu 1 :
Ma so: 1
Ho ten: Cao
Diem toan: 3
Diem ly: 6
Diem van: 8
Nhap thong tin SV thu 2 :
Ma so: 2
Ho ten: Dang
Diem toan: 9
Diem ly: 7
Diem van: 6
In thong tin SV:
Ma so: 1
Ho ten: Cao
Diem toan: 3
Diem ly: 6
Diem van: 8
Ma so: 2
Ho ten: Dang
Diem toan: 9
Diem ly: 7
Diem van: 6
```

3.8| 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 bài 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 hay 180.
- **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 hay 180.
 - **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 Thuê bao 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.

Câu 3:

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 Gói cước TV 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 Thuê bao TV 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.

4.

FILE

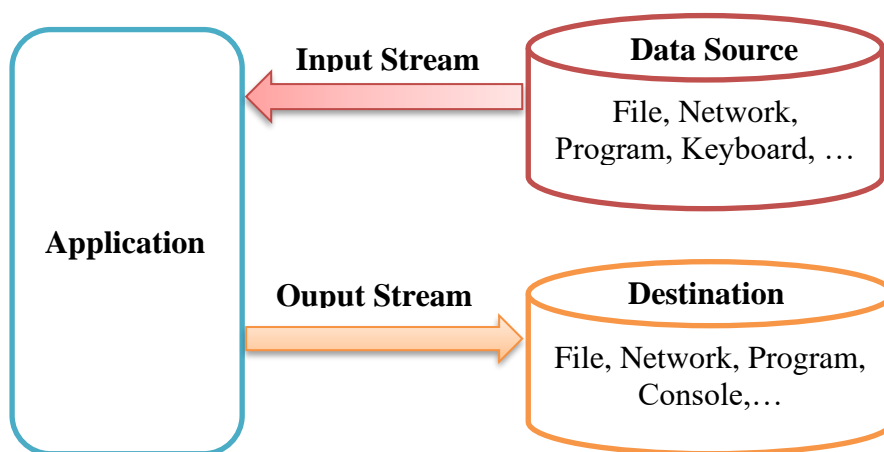
Chương này nhằm giới thiệu cho sinh viên các khái niệm file. Sử dụng File để thực hiện Input và Output cho chương trình.

4.1| 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.

Một luồng (stream) là một đối tượng được sử dụng để truyền dữ liệu. Khi dữ liệu truyền từ các nguồn bên ngoài vào ứng dụng ta gọi đó là đọc stream, và khi dữ liệu truyền từ chương trình ra nguồn bên ngoài ta gọi nó là ghi stream.

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.



Sơ đồ luồng đọc và luồng ghi

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 có thể ghi từng byte hoặc ghi một mảng các byte vào luồng (stream).

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


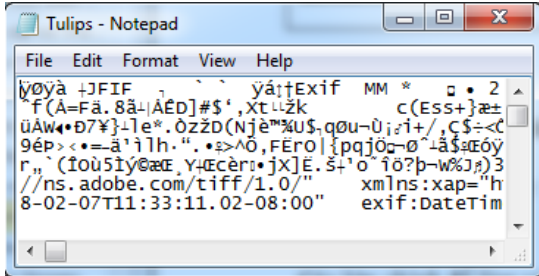
- Mở một luồng đến tập tin tương ứng.
- Thực hiện đọc hoặc viết.
- Đóng luồng.

4.2| PHÂN LOẠI TẬP TIN

Tất cả các tập tin đều tồn tại ở một trong hai dạng là Tập tin dạng văn bản và tập tin dạng nhị phân. Ngôn ngữ lập trình C# hỗ trợ hai loại luồng là luồng văn bản (text stream) và luồng nhị phân (binary stream).

Tập tin nhị phân thường chứa dữ liệu dạng dãy các byte. Khi tạo định dạng tập tin tùy chỉnh cho chương trình, lập trình viên sắp xếp các byte này thành định dạng lưu trữ thông tin cần thiết cho ứng dụng.

Các định dạng tập tin nhị phân có thể bao gồm nhiều loại dữ liệu trong cùng một tập tin, chẳng hạn như dữ liệu hình ảnh, video và âm thanh. Dữ liệu này chỉ có thể đọc được bằng các chương trình hỗ trợ đọc loại tập tin tương ứng, khi hiển thị dưới dạng văn bản sẽ bị cắt xén hoặc hiển thị mã vô nghĩa. Dưới đây là một ví dụ về tập tin hình ảnh .PNG được mở trong trình xem ảnh và trình soạn thảo văn bản.

Xem tập tin Tulips.png bằng ứng dụng Windows Photo Viewers	Xem tập tin Tulips.png bằng ứng dụng Notepad
	

Các tập tin văn bản hạn chế hơn các tập tin nhị phân vì chúng chỉ có thể chứa dữ liệu văn bản. Tuy nhiên, không giống như các tập tin nhị phân, chúng ít có khả năng bị hỏng. Mặc dù một lỗi nhỏ trong tập tin nhị phân có thể khiến nó không thể đọc được, nhưng một lỗi nhỏ trong tập tin văn bản có thể chỉ hiển thị khi tập tin được mở.

Các tập tin văn bản có thể được lưu ở định dạng văn bản thuần (.TXT) và định dạng văn bản có định dạng (.RTF). Một tập tin văn bản đơn giản điển hình chứa các dòng văn bản, mỗi dòng được theo sau bởi một ký tự Cuối dòng (EOL). Điểm đánh dấu kết thúc tập tin (EOF) được đặt sau ký tự cuối cùng, báo hiệu kết thúc tập tin. Cả văn bản thuần túy và tập tin văn bản có định dạng theo lược đồ mã hóa ký tự để xác định cách các ký tự được diễn giải và ký tự nào có thể được hiển thị.

Vì các tập tin văn bản sử dụng định dạng chuẩn, đơn giản, nhiều chương trình có khả năng đọc và chỉnh sửa tập tin văn bản. Các trình soạn thảo văn bản phổ biến như Microsoft Notepad, Notepad++, Wordpad, Microsoft Word, ...

C# hỗ trợ 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

4.3| FILESTREAM CLASS

Thư viện .NET cung cấp lớp cơ sở Stream (System.IO.Stream) để hỗ trợ làm việc với các stream, từ lớp cơ sở này một loạt lớp kế thừa cho những stream đặc thù như: FileStream, BufferStream, MemoryStream, TextReader , StreamReader ...

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 đọc/ ghi trên tập tin gồm ba bước sau:**

Bước 1 : Khai báo, khởi tạo đối tượng, cài đặt các mode

Bước 2 : Thao tác đọc hoặc ghi trên tập tin

Bước 3 : 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 (tắc nghẽn). 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.

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 tập tin 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
{
    //Khởi lệnh;
}Catch (Exception)
{
    //Xử lý
}
```

4.4| SỬ DỤNG STREAMREADER ĐỂ ĐỌC FILE

 **Đọc nội dung tập tin bằng StreamReader :**

Lớp StreamReader hỗ trợ các phương thức sau:

Phương thức	Mô tả
Close()	Đóng StreamReader hiện tại và các stream có liên quan
Peek()	Di chuyển đến ký tự có sẵn tiếp theo để đọc. Trả về -1 nếu cuối file.
Read(char buffer, int index, int count)	Đọc từng byte vào buffer , từ vị trí index, mỗi lần đọc tối đa count ký tự, kết quả đọc lưu vào mảng. Trả về số lượng byte đọc được, 0 nếu cuối stream.

ReadLine()	Đọc từng dòng từ input stream và trả về data dạng chuỗi
ReadToEnd()	Đọc từ vị trí hiện tại đến cuối file
Seek()	Được dùng để đọc hoặc ghi từ một vị trí trong file

Một số ví dụ đọc nội dung file StreamReader :

Ví dụ 1: Sử dụng phương thức Peek() để đọc từ byte:

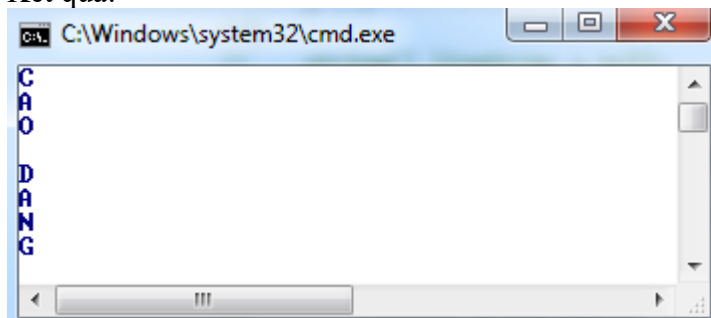
File: D: \ Example.txt
CAO DANG

Chương trình đọc tập tin văn bản và hiển thị nội dung lên màn hình console như sau:

File: Program.cs
<pre> /* Ví dụ sử dụng StreamReader để đọc tập tin */ using System; using System.IO; namespace FileExample { class Program { static void Main(string[] args) { //Khai báo đường dẫn file String path = @"D:\Example.txt"; try { //Bước 1: Tạo đối tượng StreamReader để đọc file StreamReader myStrReader = File.OpenText(path); //Bước 2: Đọc nội dung file theo từng dòng đến ký tự # // sử dụng ReadLine() using (myStrReader) { while (myStrReader.Peek() != -1) { Console.Write((char)myStrReader.Read()+"\n"); } } //Bước 3: Đóng file myStrReader.Close(); } catch (Exception) { Console.WriteLine("Không đọc được file"); } } } } </pre>

```
}  
}  
}
```

Kết quả:



Ví dụ 2: Sử dụng phương thức Read() để đọc nhóm byte:

File: D: \ Example.txt

CAO DANG

Chương trình đọc tập tin văn bản và hiển thị nội dung lên màn hình console như sau:

File: Program.cs

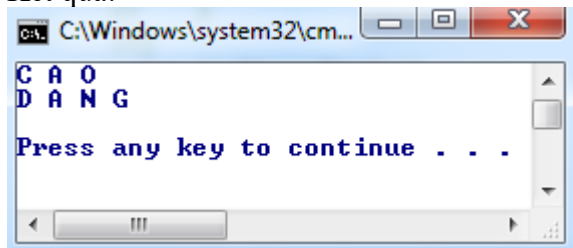
```
/* Ví dụ sử dụng StreamReader để đọc tập tin */  
using System;  
using System.IO;  
namespace FileExample  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            //Khai báo đường dẫn file  
            String path = @"D:\Example.txt";  
            try  
            {  
                //Bước 1: Tạo đối tượng StreamReader để đọc file  
                StreamReader myStrReader = File.OpenText(path);  
  
                //Bước 2: Đọc nội dung file theo từng dòng đến ký tự #  
                // sử dụng ReadLine()  
                using (myStrReader)  
                {  
                    //Đọc từng 4 bytes  
                    char[] buffer = new char[4];  
                    int count = 0;  
  
                    while((count = myStrReader.Read(buffer,0,4)) > 0)  
                    {  
                        foreach (char e in buffer)
```

```

        {
            Console.Write(e + " ");
        }
        Console.WriteLine();
    }
}
}
catch (Exception)
{
    Console.WriteLine("Khong doc duoc file");
}
}
}
}

```

Kết quả:



Ví dụ 3: Sử dụng phương thức ReadLine() để đọc từng dòng:

File: D:\Example.txt

CAO DANG

Chương trình đọc tập tin văn bản và hiển thị nội dung lên màn hình console như sau:

File: Program.cs

```

/* Ví dụ sử dụng StreamReader để đọc tập tin
 */
using System;
using System.IO;
namespace FileExample
{
    class Program
    {
        static void Main(string[] args)
        {
            //Khai báo đường dẫn file
            String path = @"D:\Example.txt";
            try
            {
                //Bước 1: Tạo đối tượng StreamReader để đọc
                file
                StreamReader myStrReader =
                File.OpenText(path);
            }
            catch { }
        }
    }
}

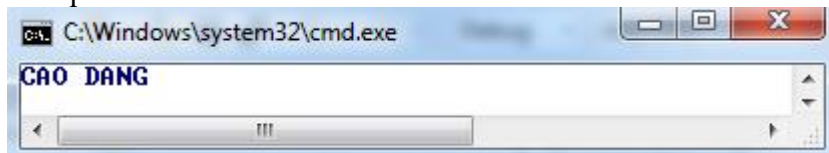
```

```

//Bước2: Đọc nội dung file theo từng dòng đến
ký tự #
// sử dụng ReadLine()
using (myStrReader)
{
    //Đọc từng dòng
    string s = null;
    while ((s = myStrReader.ReadLine()) !=
null)
    {
        Console.WriteLine(s);
    }
}
catch (Exception)
{
    Console.WriteLine("Khong doc duoc file");
}
}
}

```

Kết quả:



Ví dụ 3: Sử dụng phương thức ReadLine() để đọc đến ký tự phân tách:

File: D:\Example.txt
CAO DANG#THU DUC#

Chương trình đọc tập tin văn bản theo từng dòng với ký hiệu kết thúc dòng là ký tự # và hiển thị nội dung lên màn hình console như sau:

File: Program.cs
<pre> /* Ví dụ sử dụng StreamReader để đọc tập tin */ using System; using System.IO; namespace FileExample { class Program { static void Main(string[] args) </pre>

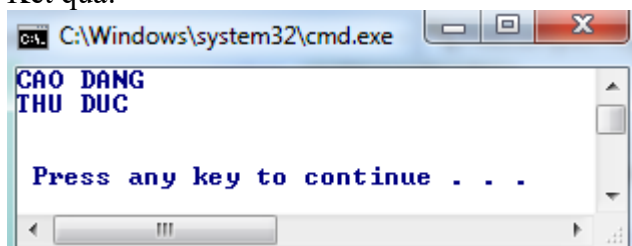
```

{
    //Khai báo đường dẫn file
    String path = @"D:\Example.txt";
    try
    {
        //Bước 1: Tạo đối tượng StreamReader để đọc file
        StreamReader myStreamReader =
File.OpenText(path) ;

        //Bước2: Đọc nội dung file theo từng dòng đến ký tự #
        // sử dụng ReadLine()
        using (myStreamReader)
        {
            string[] arrL = null;
            while ((myStreamReader.Peek() > -1) &&
(arrL = myStreamReader.ReadLine().Split('#')) != null)
            {
                foreach (string str in arrL)
                {
                    Console.Write(str + " ");
                    Console.WriteLine();
                }
                Console.WriteLine();
            }
        }
    } catch (Exception)
    {
        Console.WriteLine("Khong doc duoc file");
    }
}
}
}

```

Kết quả:



4.5| SỬ DỤNG STREAMWRITER ĐỂ GHI FILE

Lớp StreamWriter hỗ trợ các phương thức sau:

Phương thức	Mô tả
-------------	-------

Close()	Đóng StreamWriter hiện tại và các stream có liên quan
Flush()	Xóa tất cả buffers của StreamWriter hiện tại
Write(char)	Ghi một ký tự vào stream
Write(char[])	Ghi một mảng ký tự vào stream
Write(string)	Ghi một chuỗi vào stream
WriteLine()	Ghi một ký tự kết thúc dòng vào stream (kế thừa từ TextWriter)
WriteLine(char)	Ghi một ký tự vào một dòng trong stream (kế thừa từ TextWriter)
WriteLine(string)	Ghi một chuỗi vào một dòng trong stream

Ví dụ 1: Sử dụng `Console.WriteLine(string)` để **Ghi file**

```
using System;
using System.IO;

namespace FileExample
{
    class StreamWriterApp
    {
        static void Main(string[] args)
        {
            //Bước 1: Khai báo đường dẫn file
            String path = @"D:\OuputFile1.txt";
            try
            {
                //Bước 2: Tạo đối tượng StreamReader để ghi vào file
                StreamWriter myWriter = new
StreamWriter(path);
                //Bước 3: ghi vào file
                using (myWriter)
                {
                    myWriter.Write("Cao Dang Cong Nghe Thu Duc");
                }
                Console.WriteLine("Ghi file thanh cong");
            }
            catch (Exception)
            {
                Console.WriteLine("Khong the tao file");
            }
        }
    }
}
```

```
}
```

Kết quả:

```
Luu file thanh cong
```

Nếu đường dẫn không tồn tại thì màn hình kết quả sau sẽ hiển thị
Luu file thanh cong

```
Khong the tao file
```

4.6| BINARY STREAMS

Các lớp `BinaryReader` và `BinaryWriter` dùng để đọc và ghi dữ liệu vào tập tin nhị phân. Hàm tạo các đối tượng để đọc hoặc ghi được gọi thông qua hàm tạo đối tượng `FileStream`. Các thao tác đọc ghi theo các bước sau:

Bước 1 : Gọi hàm tạo đối tượng `FileStream`

Bước 2 : Gọi hàm tạo đối tượng `BinaryReader` hoặc `BinaryWriter`

Bước 3 : Thực hiện thao tác đọc hoặc ghi trên tập tin nhị phân

Bước 4 : Đóng tập tin

4.7| 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.

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:

➤ **Tạo đối tượng để ghi file**

```
FileStream output=File.Open(fileName, FileMode,fileAccess)
```

```
BinaryWriter binWriter = new BinaryWriter( output) ;
```

➤ **Ghi dữ liệu**

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

```
binWriter.Write("content");
```

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

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

Ví dụ: Chương trình ghi mảng số nguyên vào file dạng binary

```
File: Program.cs
using System;
using System.IO;

namespace BinaryFileApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            //mang cac so nguyen
            int[] arrNum = new int[] { 10, 2, 300, 4, 5000 };
            //Khai báo đường dẫn của file
            string filePath = @"D:\output.txt";

            try
            {
                //Mở file, Tạo đối tượng BinaryWriter để ghi
                FileStream outputFile = File.Open(filePath,
                    FileMode.Append, FileAccess.Write);
                BinaryWriter binWriter =
                    new BinaryWriter(outputFile);
                //Ghi vào file
                using (binWriter)
                {
                    foreach(int e in arrNum)
                    {
                        binWriter.Write(e);
                    }

                    Console.WriteLine("Ghi file thanh cong");
                }
            }
            catch(Exception)
            {
                Console.WriteLine("Khong ghi duoc vao file");
            }
        }
    }
}
```

```
}  
}
```

Kết quả dãy số nguyên 10, 2, 300, 4, 5000 được ghi vào file output.txt thành công. Tuy nhiên khi mở file này bằng một text editor thông thường thì sẽ không thể đọc được dữ liệu. Muốn đọc được dữ liệu của file này cần có đoạn chương trình để đọc dữ liệu từ file nhị phân.

4.8| 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)*

```
FileStream output=File.Open(fileName, FileMode,fileAccess)  
BinaryWriter binWriter = new BinaryWriter( output);
```

➤ *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 vị trí pos đọc num ký tự
Read(Char[], int pos, int num)	Đọc mảng số Char từ stream bắt đầu từ vị trí pos đọc num ký tự

➤ Đóng file: Sử dụng phương thức Close()

Ví dụ: Chương trình sử dụng phương thức ReadInt32() để đọc dãy các số nguyên từ file nhị phân đã ghi trong ví dụ trên vào mảng arrNum

```
/* Ví dụ sử dụng BinaryReader để đọc mảng số nguyên  
*/  
using System;  
using System.IO;  
  
namespace BinaryFileApplication  
{  
    class Program
```

```

{
    static void Main(string[] args)
    {
        //mang cac so nguyen
        int[] arrNum = new int[0];
        //Khai báo đường dẫn của file
        string filePath = @"E:\output.txt";

        try
        {
            //Mở file, Tạo đối tượng BinaryWriter để ghi
            FileStream outputFile = File.Open(filePath,
                FileMode.Open, FileAccess.Read);
            BinaryReader binReader = new
            BinaryReader(outputFile);
            //Ghi vào file
            using (binReader)
            {
                int i = 0;
                while (binReader.BaseStream.Position !=
                    binReader.BaseStream.Length)
                {
                    var integerFromFile = binReader.ReadInt32();
                    Console.WriteLine(integerFromFile);
                    Array.Resize(ref arrNum, arrNum.Length + 1);
                    arrNum[i++] = integerFromFile;
                }

                Console.WriteLine("Doc file thanh cong");
            }
        }
        catch (Exception)
        {
            Console.WriteLine("Khong doc duoc file");
        }
        foreach (int e in arrNum)
        {
            Console.Write(e + "\t");
        }
    }
}

```

Kết quả:

```

10, 2, 300, 4, 5000
Doc file thanh cong

```


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 bài 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
5. Hãy viết chương trình thực hiện ghi nội dung của struct trong Bài tập 3 dưới dạng file nhị phân. Sau đó chương trình đọc nội dung từ file nhị phân và in ra màn hình.

4.10| BÀI TẬP TỔNG HỢP

Viết chương trình quản lý Sinh viên. Biết mỗi Sinh Viên cần lưu trữ các thông tin cá nhân, điểm của từng học kỳ trong 3 năm. Chi tiết như sau:

- Mã sinh viên : Kiểu chuỗi gồm 11 ký tự, chỉ cho phép ký tự chữ và số và không chứa khoảng trắng trong chuỗi
- Họ tên : Kiểu chuỗi không quá 30 ký tự, không chứa khoảng trắng đầu và cuối chuỗi, viết hoa đầu từ.
- Ngày sinh: kiểu số DateTime với năm lớn hơn 1900, dữ liệu ngày tháng hợp lệ
- Điểm trung bình tốt nghiệp : kiểu số thực (tính tự động, không nhập)
- Bảng điểm trung bình : là bảng gồm 3 dòng và 2 cột chứa các phần tử kiểu số thực trong khoảng từ 0 đến 10, định dạng kiểu 0.0:

	Học kỳ 1	Học kỳ 2
Năm 1		
Năm 2		
Năm 3		

Chương trình cần xây dựng có chức năng sau:

- Khai báo và khởi tạo dữ liệu cho một Sinh viên trong hệ thống với các dữ liệu về thông tin cá nhân bắt buộc người dùng nhập vào theo các ràng buộc mô tả ở trên. Điểm trung bình năm và các cột điểm trong bảng điểm mặc định là 0.
- In tất cả thông tin cá nhân của sinh viên gồm Mã sinh viên, Họ tên, Năm sinh ra màn hình.
- In bảng điểm của sinh viên ra màn hình gồm thông tin: Mã sinh viên , Bảng điểm (thể hiện theo Năm – cột và học kỳ - dòng)
- Hàm nhập điểm cho sinh viên với tham số đầu vào là số năm trên cột (1, 2, hoặc 3) và số học kỳ trên dòng (1 hoặc 2). Điểm được gán vào bảng với vị trí tương ứng và phải theo ràng buộc về giá trị như mô tả ở trên.
- Hàm tính Điểm trung bình theo từng năm học cho sinh viên theo công thức sau:
$$\text{Điểm trung bình năm} = (\text{Điểm học kỳ 1} + \text{Điểm học kỳ 2}) / 2$$

Tham số đầu vào là số năm (1, 2 hoặc 3)
- Hàm tính điểm Trung bình tốt nghiệp cho Sinh viên như sau:
$$\text{Điểm trung bình tốt nghiệp} = (\text{ĐTB năm 1} + \text{ĐTB năm 2} + \text{ĐTB năm 3}) / 3$$

Điểm trung bình tốt nghiệp được ghi vào dữ liệu của Sinh viên.

g. Hàm xếp loại cho Sinh viên theo cách sau:

- Từ 0 đến dưới 5 : Yếu
- Từ 5 đến dưới 7 : Trung bình
- Từ 7 đến dưới 8 : Khá
- Từ 8 đến 10 : Giỏi

h. Hàm có chức năng in Bảng điểm cho sinh viên vào tập tin dạng .txt như sau:

Mẫu	Ví dụ:
<pre>***** BANG DIEM TONG KET ***** MSSV : <mã sinh viên> Ho ten : <họ tên sinh viên> Ngay sinh : <năm sinh> Diem trung binh : <ĐTB tốt nghiệp> Xep loai : <loại> Diem trung binh chi tiet : <bảng điểm trung bình></pre>	<pre>***** BANG DIEM TONG KET ***** MSSV : 19211TT1234 Ho ten : Nguyen Van Anh Ngay sinh : 31/12/2000 Diem trung binh : 8.2 Xep loai : Gioi Diem trung binh chi tiet : 8.0 9.5 7.5 7.0 8.3 8.7</pre>

i. Hàm đọc thông tin cá nhân và bảng điểm của n sinh viên. Với n là số lượng sinh viên và thông tin của một sinh viên được đọc theo từng dòng từ file. Ví dụ dữ liệu trong file như sau :

MSSV#Ho ten#ngay/thang/nam# d1 d2 d3 d4 d5 d6

Với các điểm sẽ được đọc vào bảng hai chiều theo hướng từ trái qua phải, từ trên xuống dưới

Dòng 1	3
Dòng 2	19211TT1231#Nguyen Van An#25/3/2000#5.0 7.5 6.3 9.2 7.5 8.5
Dòng 3	19211TT1232#Nguyen Van Binh#2001#7.5 6.5 5.3 8.2 6.5 9.5
Dòng 4	19211TT1233#Nguyen Van Chau#2000#7.0 6.5 4.5 8.0 6.2 7.0

j. Tìm và in thông tin của sinh viên có Điểm trung bình tốt nghiệp lớn nhất

TÀI LIỆU THAM KHẢO:

- [1] Kỹ Thuật lập trình , Nguyễn Trung Trực, NXB ĐH Quốc Gia TP.HCM, 2018
- [2] Joyce Farrell, **Microsoft Visual C# 2015: An Introduction to Object-Oriented Programming**, Cengage Learning, 201, 864 trang