# 1. GIỚI THIỆU VỀ PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chương này nhằm giới thiệu cho sinh viên các khái niệm, các đặc điểm của phương pháp lập trình Hướng đối tượng, phân biệt phương pháp hướng cấu trúc và hướng đối tượng, sử dụng một số thư viện được xây dựng sẵn.

## 1.1| Giới thiệu kiểu dữ liệu struct

### 1.1.1| 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.

### 1.1.2| 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.

### 1.1.3| 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;
    }
    //Ham 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;
}
```

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

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 . . .
```

### 1.1.5| 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
```

### 1.1.6| 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(" Nhap thong tin sinh vien: ");
            /*
             * Đây là hàm hỗ trợ nhập thông tin sinh viên.
             * Sử dụng từ khoá out để có thể cập nhật giá trị
nhập được ra biến SV1 bên ngoài
             */
            NhapThongTinSinhVien(out sV1);
            Console.WriteLine("*********");
            Console.WriteLine(" Thong tin sinh vien vua nhap la:
");
            XuatThongTinSinhVien(sV1);
            Console.WriteLine(" Diem TB cua sinh vien la: " +
DiemTBSinhVien(sV1));

            Console.ReadLine();
        }
        //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: 7.66666666666667
```

### 1.1.7| 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;
        }

        //Nhap mang Sinh vien
```

```
        static SinhVien[] NhapMang(int soPT)
        {
            SinhVien[] arrSV = new SinhVien[soPT];
            for(int i = 0; i < soPT; i++)
            {
                Console.WriteLine("Nhap thong tin SV thu {0} :",
                                    i + 1);
                NhapThongTinSinhVien(out arrSV[i]);
            }
            return arrSV;
        }

        //Xuat mang 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
```

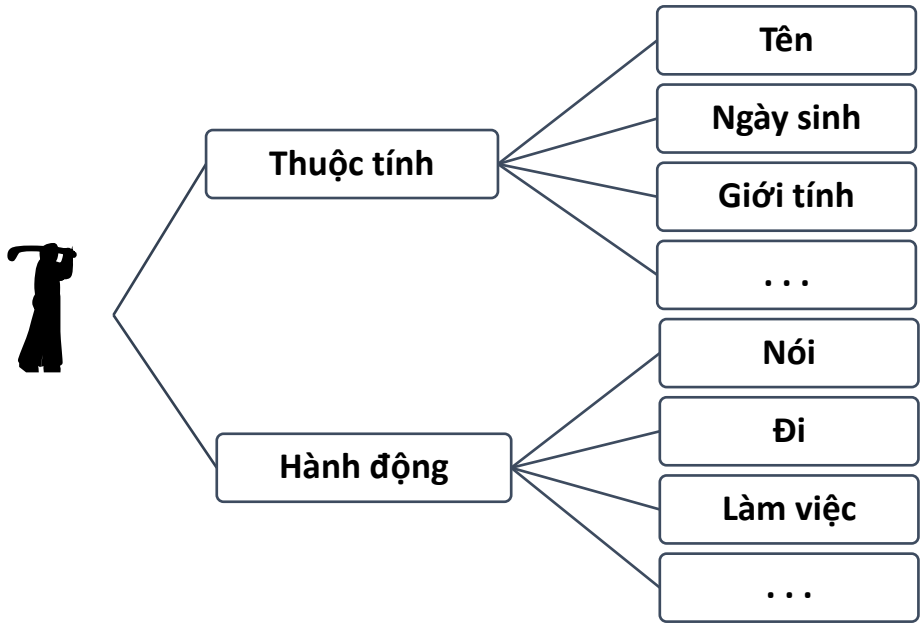
```
Diem ly: 6
Diem van: 8
Ma so: 2
Ho ten: Dang
Diem toan: 9
Diem ly: 7
Diem van: 6
```

## 1.2| Khái niệm lập trình Hướng đối tượng

Lập trình Hướng đối tượng có tên tiếng Anh là Object-Oriented Programming Languages, được viết tắt là OOP. Đây là một phương pháp lập trình sử dụng các đối tượng làm vai trò trung tâm để xây dựng chương trình. Lập trình hướng đối tượng giúp cho lập trình viên có thể tổ chức một chương trình lớn thành những đối tượng rời rạc, độc lập, ít phụ thuộc lẫn nhau, nhờ đó có thể dễ dàng kiểm soát, hiệu chỉnh, nâng cấp chương trình mà không ảnh hưởng đến các thành phần khác của chương trình.

Trong lập trình hướng đối tượng, một đối tượng được định nghĩa bao gồm hai thành phần chính là:

- Tập các thuộc tính dữ liệu (attributes): là những thông tin, đặc điểm của đối tượng. Ví dụ một người có những đặc điểm như họ tên, ngày sinh, giới tính nghề nghiệp, màu mắt, màu tóc, …
- Tập các phương thức (methods) là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ một người sẽ có thể thực hiện các hành động như nói, đi, làm việc, ăn, uống, …



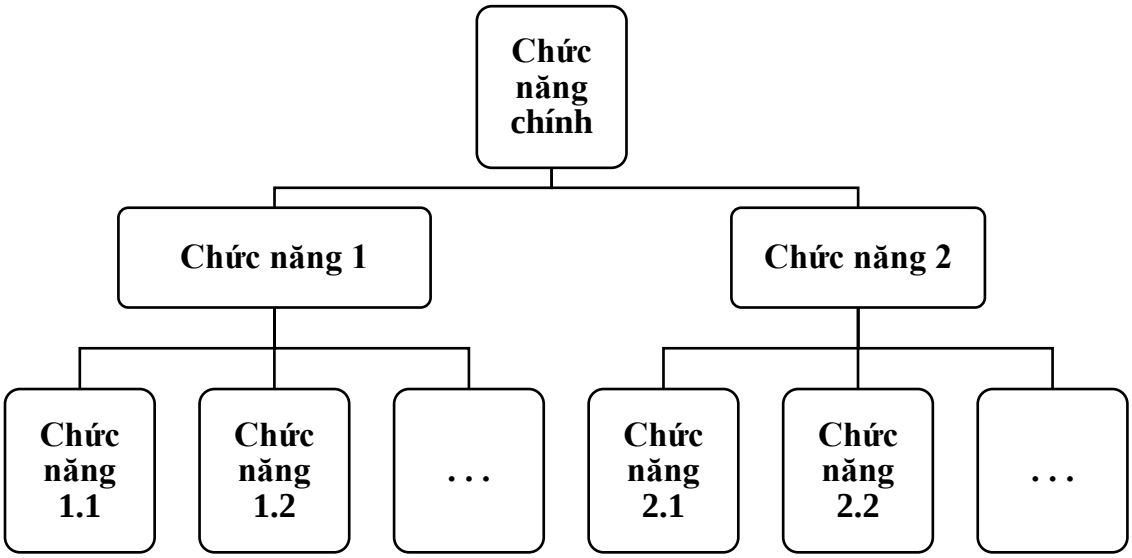

*Hình 1. Ví dụ về thuộc tính và hành động của đối tượng Người*

## 1.3| So sánh lập trình Hướng đối tượng và lập trình theo kiểu cấu trúc

### 1.3.1| Phương pháp lập trình kiểu cấu trúc

Phương pháp lập trình có cấu trúc (structured programming) hay còn gọi phương pháp lập trình hướng thủ tục (**P**rocedure-**O**riented **P**rogramming -**POP**) là phương pháp phân tích một nhiệm vụ lớn thành nhiều công việc nhỏ hơn – chia để trị. Cách thức phân tích và thiết kế theo nguyên lý lập trình từ trên xuống (top-down). Phương pháp này thể hiện quá trình suy diễn từ cái chung cho đến cái cụ thể, từ một chương trình lớn thành các chương trình con nhỏ hơn có chức năng độc lập. Chương trình con đóng vai trò trung tâm của việc lập trình hay còn gọi phương pháp Lập trình hướng thủ tục.

Chương trình có cấu trúc được tổ chức thành các chương trình con riêng lẻ (còn gọi là các module hay phương thức), sử dụng các cấu trúc tuần tự, cấu trúc rẽ nhánh và cấu trúc lặp. Mỗi chương trình con đảm nhận xử lý một công việc nhỏ trong toàn bộ hệ thống.




*Hình 2. Mô hình phân rã chức năng theo phương pháp lập trình có cấu trúc*

Trong lập trình hướng cấu trúc thường quan tâm đến việc phát triển các phương thức mà ít quan tâm tới dữ liệu, điều này khiến cho dữ liệu khó kiểm soát. Để liên kết giữa các phương thức với nhau, thường dùng biến toàn cục hoặc con trỏ.

Ưu điểm của phương pháp lập trình hướng thủ tục: triển khai các phần mềm dễ dàng, chương trình dễ hiểu và dễ bảo trì. Các nhược điểm của phương pháp lập trình hướng thủ tục như sau:

- Không thích hợp khi xây dựng các chương trình lớn và phức tạp.
- Phần lớn các phương thức sử dụng dữ liệu chung nên khi có một sự thay đổi dữ liệu phải thực hiện thay đổi ở tất cả phương thức liên quan đến dữ liệu đó. Đây là công việc tốn thời gian và kém hiệu quả.
- Bảo mật dữ liệu kém
- Cách tiếp cận đôi khi không phù hợp với thực tế, khó mô tả được các hoạt động của thế giới thực.

Để khắc phục các nhược điểm trên phương pháp lập trình hướng đối tượng ra đời.

### 1.3.2| Phương pháp lập trình Hướng đối tượng

Lập trình hướng đối tượng **(O**bject **O**riented **P**rogramming – **OOP**) là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng các phương thức và xây dựng chương trình. Theo mô hình đối tượng, chương trình có cấu trúc là một tập các đối tượng độc lập, tương tác với nhau khi cần thiết để hoàn thành nhiệm vụ của chương trình.

Trong phương pháp lập trình hướng đối tượng thì ta thường tư duy theo hướng thực hiện thao tác gì với các đối tượng đã có để giải quyết bài toán đặt ra. Với cách tư duy này, đối tượng là trung tâm của việc lập trình, người ta gọi là nguyên lý lập trình từ dưới lên (Bottom-up).

Ưu điểm của chương trình hướng đối tượng là cấu trúc thuần nhất chỉ chứa một loại thành phần là đối tượng nên dễ dàng quản lý code hơn khi có sự thay đổi chương trình, dễ mở rộng chương trình, có tính bảo mật dữ liệu cao và có tính tái sử dụng lại phần mềm.

Nhược điểm của chương trình hướng đối tượng là không phù hợp với mọi loại chương trình, một số chương trình có thể sử dụng nhiều bộ nhớ hơn và thực thi nhiều lệnh hơn so với lựa chọn phương pháp lập trình hướng thủ tục.

| | **POP**<br>(Lập trình hướng thủ tục) | **OOP**<br>(Lập trình hướng đối tượng) |
|---|---|---|
| Hướng thiết kế chương trình | Từ trên xuống (Top-down) | Từ dưới lên (Bottom-up) |
| Cách phân chia chương trình | Chia nhỏ theo các chức năng (functions).<br>Một phương thức có thể chứa nhiều dữ liệu khác nhau | Chia nhỏ theo các đối tượng (objects).<br>Một đối tượng chỉ điều khiển dữ liệu của nó |
| Thiết kế thuật toán | Tập trung xây dựng các thuật toán theo cách có hệ thống | Tập trung vào bảo mật dữ liệu, không phân biệt các thuật toán |

| | | |
|---|---|---|
| Tính bảo mật dữ liệu | Rất khó để che dấu dữ liệu | Dễ dàng cho phép hoặc giới hạn quyền truy cập đến các dữ liệu bằng các từ khóa: public, protected, private |
| Khả năng mở rộng chương trình | Rất khó chỉnh sửa, mở rộng khi chương trình có sự thay đổi về dữ liệu.<br>Tính tái sử dụng thấp | Dễ dàng chỉnh sửa, mở rộng khi chương trình có sự thay đổi về dữ liệu.<br>Tính tái sử dụng cao |
| Tính kế thừa ( tái sử dụng) | Không cho phép kế thừa | Cho phép kế thừa các phương thức và thuộc tính có sẵn |
| Đa năng hóa các toán tử, phương thức | Không cho phép | Các toán tử, phương thức có thể thực hiện với nhiều kiểu dữ liệu khác nhau, theo nhiều cách khác nhau |

*Bảng so sánh POP và OOP*

## 1.4| Mục tiêu của lập trình hướng đối tượng

Lập trình hướng đối tượng được thực hiện thông qua các ngôn ngữ lập trình hướng đối tượng để đạt được các mục tiêu sau:

- **Tính mạnh mẽ:**
  - Cho phép các chương trình phức tạp có thể hoạt động chính xác.
  - Nếu một thư viện có sẵn không phù hợp yêu cầu thì người lập trình có khả năng sửa đổi hoặc mở rộng một cách dễ dàng, không cần phải can thiệp đến mã nguồn thư viện.
  - Đơn giản hoá việc xây dựng và sử dụng các thư viện.
  - Nâng cao độ tin cậy và tính bền vững của phần mềm.

- **Khả năng thích nghi**
  - Hỗ trợ  mạnh các dự án phát triển phần mềm quy mô lớn, đòi hỏi nhiều người tham gia, phần mềm phát triển trong một thời gian dài.
  - Có thể chạy trên nhiều kỹ thuật và phần cứng khác nhau
- **Khả năng tái sử dụng**
  - Các đối tượng có thể sử dụng lại trong một ứng dụng hoặc nhiều ứng dụng khác nhau.

## 1.5| Các đặc điểm của lập trình Hướng đối tượng

Lập trình Hướng đối tượng có bốn đặc điểm sau:

- **Tính trừu tượng (Abtraction)**

Tính trừu tượng là một đặc điểm cơ bản của lập trình hướng đối tượng. Mục tiêu của nó là làm giảm độ phức tạp cho chương trình bằng cách loại bỏ những chi tiết không cần thiết của đối tượng và chỉ tập trung vào những chi tiết cốt lõi, quan trọng.

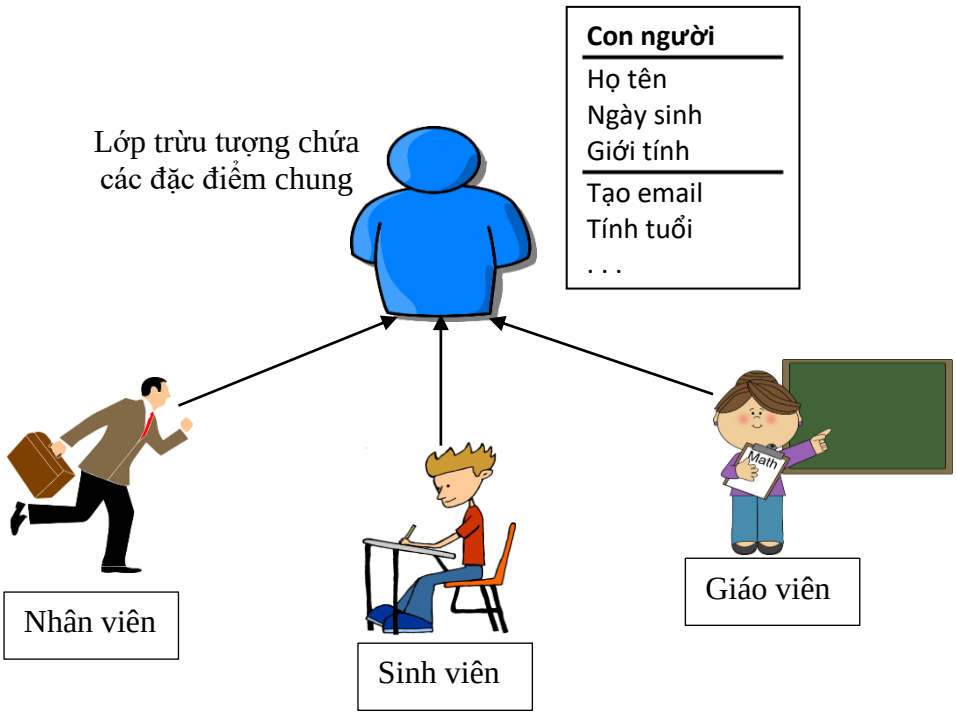
Trừu tượng còn là một quá trình ẩn các chi tiết triển khai và chỉ hiển thị chức năng cho người dùng. Mỗi đối tượng có thể thực hiện các công việc, tương tác với các đối tượng khác mà không cần phải biết làm cách nào đối tượng tiến hành được các thao tác.

Nâng cao hơn nữa là tính trừu tượng còn thể hiện qua việc một đối tượng ban đầu có thể có một số đặc điểm chung cho nhiều đối tượng khác như là sự mở rộng của nó nhưng bản thân đối tượng ban đầu này có thể không được định nghĩa chi tiết cách thực hiện. Tính trừu tượng này thường được xác định trong khái niệm gọi là lớp cơ sở trừu tượng (abstract class) và giao diện (interface).

Giả sử bạn cần tạo một ứng dụng quản lý kết quả học tập của sinh viên trong một trường học. Sinh viên là một trong các đối tượng của hệ thống này. Có rất nhiều thông tin về một sinh viên như: họ tên, ngày sinh, địa chỉ, giới tính, quốc tịch, sở thích, màu da, chiều cao, cân nặng, nhóm máu, … Tính trừu tượng thể hiện ở việc chúng ta chỉ chọn một số thông tin thực sự hữu ích cho hệ thống như họ tên, ngày sinh, giới tính và bỏ qua các thông tin khác không cần thiết.

Một số phương thức có thể thực hiện trên đối tượng sinh viên như truy cập, cập nhật, xóa các dữ liệu, tạo email tự động cho sinh viên từ thông tin họ tên và ngày sinh của sinh viên. Như vậy các hành động này được định nghĩa như là một hành động của đối tượng sinh viên. Khi người dùng gọi các phương thức này, ví dụ gọi phương thức tạo tạo email tự động, thì họ không cần quan tâm chi tiết phương thức được thực hiện bên trong như thế nào mà người dùng chỉ cần quan tâm đến việc sử dụng phương thức đó và kết quả trả về của nó.

Tuy nhiên, các đối tượng khác có tương tác với hệ thống như giáo viên, nhân viên cũng có các thông tin giống như đối tượng sinh viên như họ tên, ngày sinh, giới tính và phương thức tạo email nhưng việc thực hiện ở từng đối tượng khác nhau có thể khác nhau. Khi đó một lớp trừu tượng (abstract class) được tạo ra để chứa tất cả các thuộc tính và phương thức chung của các đối tượng giáo viên, sinh viên, nhân viên, có thể đặt tên lớp trừu tượng này là lớp "Con người". Chúng ta cũng có thể sử dụng một giao diện (interface), interface này không chứa định nghĩa chi tiết phương hoạt động như thế nào mà chỉ chứa khai báo khuôn mẫu cho các phương thức chung.

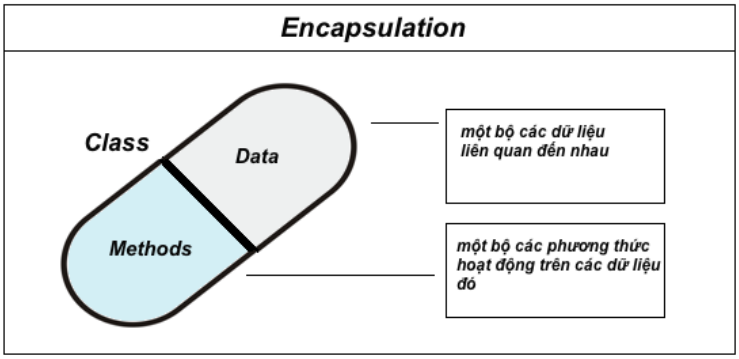



*Hình 3. Minh họa tính trừu tượng của OOP*

- **Tính đóng gói (Encapsulation)**

Tính đóng gói (encapsulation) và che giấu thông tin (information hidding) là tính chất không cho phép người sử dụng thay đổi trạng thái nội tại của đối tượng. Chỉ có các chức năng nội tại của đối tượng cho phép thay đổi trạng thái của nó. Việc cho phép môi trường bên ngoài tác động lên các dữ liệu nội tại của một đối tượng theo cách nào là hoàn toàn tùy thuộc vào người viết mã nguồn nhằm đảm bảo sự toàn vẹn của đối tượng.

Sự đóng gói còn là cơ chế ràng buộc dữ liệu và thao tác trên dữ liệu đó thành một thể thống nhất, tránh được các tác động bất ngờ từ bên ngoài. Thể thống nhất này gọi là lớp. Cơ chế đóng gói là phương thức tốt để thực hiện cơ chế che dấu thông tin so với các ngôn ngữ lập trình cấu trúc.




*Hình 4. Minh họa tính đóng gói của OOP*

Khi khai báo lớp các thành phần dữ liệu thông thường sẽ được giới hạn quyền truy cập là riêng tư để tránh việc truy cập trực tiếp từ bên ngoài lớp vào các dữ liệu của lớp. Việc truy cập vào các dữ liệu của lớp chỉ được thông qua việc gọi các phương thức, nhờ vậy các thao tác khi truy cập đến các dữ liệu của lớp sẽ được thực hiện thông qua các đoạn lệnh kiểm soát trong các phương thức. Ngoài ra các chi tiết phức tạp cài đặt bên trong mã nguồn của lớp cũng được che dấu.

**Ví dụ**: Trong hệ thống quản lý kết quả học tập sinh viên, người dùng lớp Sinh Viên có thể đọc nhưng không được phép sửa các thông tin họ tên, ngày sinh của một đối tượng sinh viên đã được tạo ra trong hệ thống, nhưng người dùng có thể được toàn quyền thêm, xóa, sửa đối với dữ liệu địa chỉ của sinh viên. Hơn nữa, người dùng sẽ không thấy được các mã nguồn chi tiết cài đặt bên trong lớp Sinh viên cũng như các thuộc tính và phương thức được

gán quyền riêng tư. Tính đóng gói này giúp tăng sự toàn vẹn dữ liệu và che dấu sự phức tạp mã nguồn bên trong lớp.

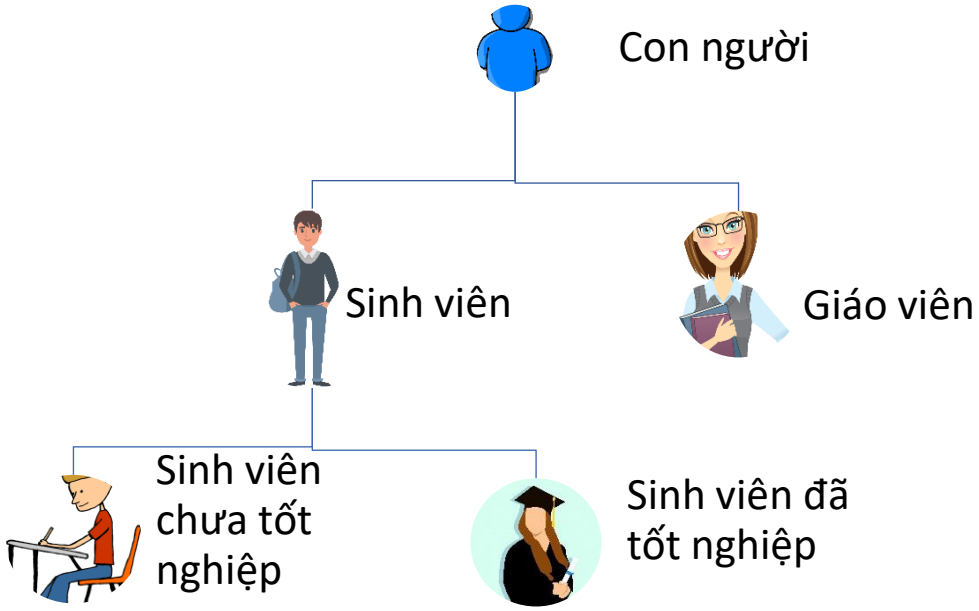
- **Tính kế thừa (Inheritance)**

Tính kế thừa trong OOP cho phép chúng ta xây dựng các lớp mới từ các lớp đã có. Một lớp mới gọi là lớp dẫn xuất (derived class) có thể thừa hưởng dữ liệu và các phương thức của lớp cơ sở (base class) ban đầu và cho phép định nghĩa thêm các dữ liệu, phương thức mới.

Mỗi lớp cơ sở không có giới hạn về số lượng các lớp dẫn xuất được tạo ra. Qua cơ chế kế thừa này, dạng hình cây của các lớp được hình thành. Dạng cây của các lớp trông giống như các cây gia phả vì thế các lớp cơ sở còn được gọi là lớp cha (parent class) và các lớp dẫn xuất được gọi là lớp con (child class).

**Ví dụ**: Trong hệ thống quản lý kết quả học tập của sinh viên ở trên cần mở rộng quản lý cho hai loại đối tượng là sinh viên chưa tốt nghiệp và sinh viên đã tốt nghiệp, Quan sát sơ đồ lớp sau:

Trong hệ thống có Lớp cơ sở / lớp cha (base class / parent class) Người có hai lớp dẫn xuất / lớp con (derived class / child class) là lớp SinhVien và lớp Giáo viên. Hai lớp con được thừa hưởng tất cả các thuộc tính và phương thức mà lớp cha Người có mà không cần phải định nghĩa lại.

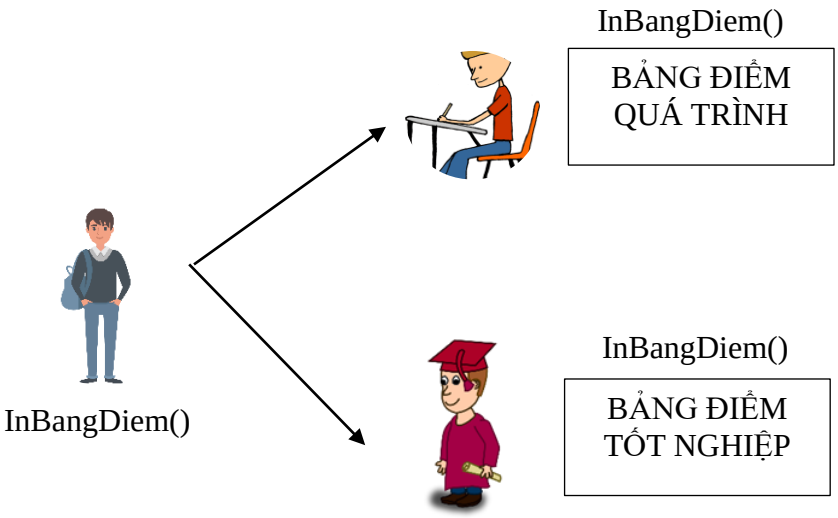



*Hình 5. Minh họa tính kế thừa*

- **Tính đa hình (Polymorphism)**

Tính đa hình thể hiện thông qua việc gửi các thông điệp (message). Việc gửi các thông điệp này có thể so sánh như việc gọi các phương thức bên trong của một đối tượng. Các phương thức dùng trả lời cho một thông điệp sẽ tùy theo đối tượng mà thông điệp đó được gửi tới, sẽ có phản ứng khác nhau. Vậy tính đa hình là khả năng một thông điệp có thể có cách thực hiện khác nhau tùy theo đối tượng nhận thông điệp.

**Ví dụ:** cùng một hành động in bảng điểm cho Sinh viên. Nhưng đối với mỗi loại sinh viên khác nhau thì thông tin chi tiết trong bảng điểm sẽ khác nhau. Như vậy cùng một đối tượng sinh viên, cùng một hành động là in bảng điểm nhưng tại các thời điểm khác nhau (đang học hoặc đã tốt nghiệp) thì cách thực hiện của hành động in bảng điểm cũng khác nhau. Người dùng chỉ cần truy cập một phương thức duy nhất nhưng kết quả trả về sẽ khác nhau tại các thời điểm khác nhau. Điều này thể hiện tính đa hình của các phương thức trong lập trình hướng đối tượng.




*Hình 6. Minh họa tính đa hình*

## 1.6| Lớp và Đối tượng

### 1.6.1| Khái niệm Đối tượng (Objects)

Đối tượng là người, vật, hiện tượng mà con người tác động tới (trong suy nghĩ, hành động). Mỗi phần tử trong thế giới thực xung quanh chúng ta là một đối tượng, ví dụ: giáo viên A, sinh viên B, máy tính MT, xe X….

Một thực thể (instance) là một vật  thể có thực bên trong bộ nhớ, thực chất đó là một đối tượng (nghĩa là một đối tượng được cấp phát vùng nhớ).

Trong lập trình hướng đối tượng, đối tượng được hiểu như là một thực thể (instance): người, vật hoặc một bảng dữ liệu, . . .Khi một phần tử dữ liệu được khai báo là từ một lớp thì nó được gọi là một đối tượng.

Một đối tượng bao gồm hai thông tin: thuộc tính và phương thức.

- Thuộc tính chính là những thông tin, đặc điểm của đối tượng. Ví dụ: một người sẽ có họ tên, ngày sinh, màu da, kiểu tóc, . . .
- Phương thức là những thao tác, hành động mà đối tượng đó có thể thực hiện. Ví dụ: một người sẽ có thể thực hiện hành động nói, đi, ăn, uống, . . .

### 1.6.2| Khái niệm lớp

Lớp (còn được gọi là "lớp đối tượng", class) được dùng để mô hình hóa một nhóm các thực thể cùng loại trong thế giới thực.

Ví dụ: lớp Sinh viên (là tập hợp các đối tượng sinh viên đều có dữ liệu mã sinh viên, họ tên, lớp, điểm trung bình, ăn, ngủ, đi học...).

### 1.6.3| Thuộc tính, phương thức

Các đối tượng đều có hai thành phần chính đó là thành phần thuộc tính và thành phần phương thức. Thuộc tính (attributes) bao gồm những thông tin mô tả về đối tượng còn phương thức chính là các hành động của đối tượng hay các phương thức thao tác, tác động lên dữ liệu (thuộc tính) của đối tượng.

*Dấu hiệu nhận diện (indentity): khi đọc mô tả của một bài toán, để xác định thuộc tính và phương thức của đối tượng chúng ta dựa vào thông tin sau:*

*Tìm các danh từ* → Thuộc tính mô tả đối tượng, ta cần biết thông tin gì về đối tượng

*Tìm các động từ* → Hành vi (phương thức) của đối tượng, xét xem đối tượng cần có xử lý gì.

Tất cả các đối tượng cùng loại đều có thuộc tính và phương thức giống nhau. Do đó *lớp đối tượng* cũng có hai thành phần đó là thành phần thuộc tính và thành phần phương thức trong đó thuộc tính của các đối tượng này chính là thuộc tính của lớp và phương thức của các đối tượng cũng chính là phương thức của lớp. Mỗi phương thức của lớp thực chất là một phương thức được viết riêng cho các đối tượng của lớp, chỉ được phép gọi để tác động lên chính các đối tượng của lớp này.

Ví dụ: Lớp Sinh viên

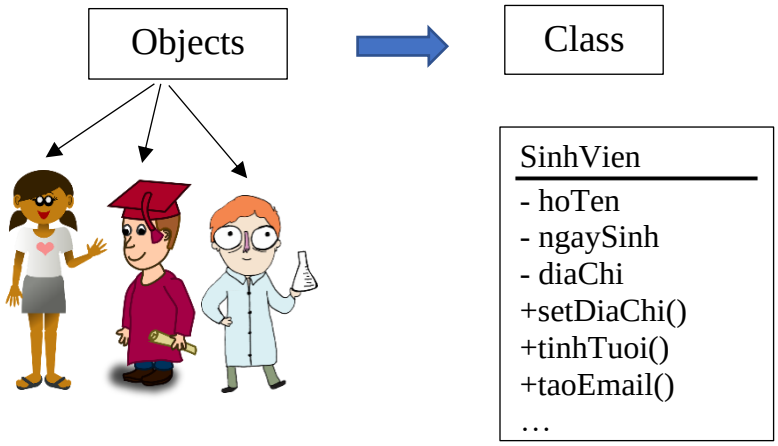
+ Thuộc tính: mã sinh viên, tên sinh viên, điểm trung bình, xếp loại
+ Phương thức: khởi tạo, gán giá trị cho các thành phần thông tin, cập nhật giá trị cho các dữ liệu, in thông tin, tạo email tự động, tính tuổi …

### 1.6.4| Phân biệt Lớp và Đối tượng

Lớp là một khái niệm bao quát nói lên những đặc điểm chung nhất của các đối tượng, còn đối tượng chính là một phần tử tồn tại thực tế của lớp.

Ví dụ:

Sinh viên Lan, Tuấn, Hoa, ... đều là các đối tượng thuộc lớp Sinh viên nói chung.




*Hình 7. Minh họa class và objects*

Lớp là khuôn mẫu để tạo các đối tượng. Mỗi đối tượng có cấu trúc và hành vi giống như lớp đối tượng mà nó được tạo từ đó.

Lớp là cái chúng ta thiết kế và lập trình còn đối tượng được tạo (từ một lớp) tại thời gian chạy. Ví dụ lớp Student tạo ra nhiều đối tượng như *Jenna, John*, ...

## 1.7| Một số lớp được xây dựng sẵn trong C#:,

### 1.7.1| String class

#### 1.7.1.1| 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 tuừ U+0000 đến U+FFFF. String class được định nghĩa trong thư viện chuẩn .NET. Hay nói cách khác đối tượng String là tập hợp dãy

System.Char. Kích thước lớn nhất của một string objects khoảng 2GB hay khoảng một tỷ ký tự.

Các đặc điểm của lớp String:

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

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

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

// Sử dụng hàm tạo khi khai báo string với số ký tự lặp lại
```

```
string alphabet = new string('A', 5); // chuỗi AAAAA

// 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}";
```

### 1.7.1.3| 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 tuỳ chỉnh:**

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

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

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

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

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

Kết quả:

```
Chieu dai chuoi : 8
C a o   D a n g
```

**Một số phương thức xử lý chuỗi:**

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

| | |
|---|---|
| bool **EndsWith**(string value)<br>Vd: str1.EndsWith(str2) | Trả về true nếu str1 có chứa str2 cuối chuỗi. Ngược lại trả về false |
| bool **Equals**(string value)<br>bool **Equals**(string a, string b)<br><br>Vd: str1.Equals(str2)<br>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)<br>Vd: str1 .IndexOf(str2) | Trả về vị trí bắt đầu chuỗi str2 trong chuỗi str1. Nếu không có trả về -1 |
| Public int I**ndexOfAny**(char[] anyof)<br>Char[] ch = { 'h', 'T' };<br>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)<br>Vd: str1.Insert(3, str2) | Chèn chuỗi str2 vào str1 tại vị trí startIndex |
| Bool **IsNullOrEmpty** (string str)<br><br>Vd: String .IsNullOrEmpty (str1) | Trả về true nếu chuooix str null hoặc empty |
| int **LastIndexof**(string str)<br><br>Vd: str1.lastindexof("u") | Trả về chỉ mục cho sự xuất hiện cuối cùng của<br><br>một chuỗi str đã cho bên trong đối tượng hiện t<br><br>ại |
| Public string remove(int startindex,<br><br>int count) | Xoá trong chuỗi str1 từ vị trí thứ startindex xoá<br><br>count ký tự |

| | |
|---|---|
| Vd: str1.Remove(8, 4) | |
| String **replace**(string oldvalue, string newvalue)<br>Vd: str.Replace(str1, str2) | Thay thế tất cả chuỗi str1 xuất hiện trong chuỗi bằng chuỗi str2. Trả về chuỗi mới. |
| string ToLower()<br>Vd: str1.ToLower() | Chuyển chuỗi str1 thành chữ thường. Trả về chuỗi mới |
| string ToUpper()<br>Vd: str1.ToUpper() | Chuyển chuỗi str1 thành chữ hoa. Trả về chuỗi mới |
| string Trim()<br>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
```

### 1.7.2| DateTime

#### 1.7.2.1| 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.

#### 1.7.2.2| 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 . . .
```

### 1.7.2.3| 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 tri 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.Write(myDate);
    }
  }
}
```

**Kết quả:**

```
12/31/9999 11:59:59 PM
```

### 1.7.2.4| 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)

### 1.7.2.5| 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 substractResult = 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:  " +
          substractResult);
      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
```

### 1.7.2.6| 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.
```

### 1.7.2.7| 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
```

### 1.7.3| Array class

Lớp Array cung cấp các phương thức để tạo, thao tác, tìm kiếm và sắp xếp mảng, do đó đóng vai trò là lớp cơ sở cho tất cả các mảng theo thời gian thực.

#### 1.7.3.1| Các thuộc tính:

- **IsFixedSize**: trả về true, giá trị chỉ định liệu mảng có kích thước cố định hay không.
- **IsReadOnly**: trả về giá trị true nếu mảng đang xét là mảng chỉ đọc.
- **IsSynchronized**: trả về giá trị true nếu việc truy cập mảng là đồng bộ.
- **Length**: trả về tổng số phần tử của tất cả các chiều trong mảng.
- **LongLength**: trả về giá trị số phần tử của tất cả các chiều trong mảng ở dạng số nguyên 64 bit.
- **MaxLength**: trả về số phần tử tối đa có thể chứa trong mảng.

- **Rank**: trả về số chiều của mảng. Mảng 1 chiều thì Rank sẽ bằng 1. Mảng n chiều thì Rank là n.

### 1.7.3.2| Một số phương thức:

| Phương thức | Mô tả |
|---|---|
| **Clear**(Array arr, int index, int num) | Xóa mảng arr từ vị trí thứ **index**, xóa **num** phần tử |
| **LastIndexOf** (Array arr, object value) | Tìm vị trí xuất hiện cuối cùng của phần tử **value** trong mảng **arr** |
| **Copy**(Array source, int fromIndex, Array dest, int destIndex, int length) | Sao chép length phần tử từ mảng **source** sang mảng **dest** từ vị trí **fromIndex** trong mảng source tới vị trí **destIndex** trong mảng dest. |
| **Clone**() | Tạo ra một bản sao của mảng |
| **Equals**(object obj1, object obj2) | So sánh xem đối tượng **obj1** có bằng với đối tượng **obj2**. Trả về false khi hai đối tượng không cùng tham chiếu |
| **Sort(Array arr)** | Sắp xếp mảng arr theo thứ tự tăng dần |
| **Reverse( Array arr)** | Đảo ngược mảng các phần tử của mảng arr |

Ví dụ :

Ví dụ sử dụng Array class

```
using System;
namespace ArrayClass_Example
{

    class Program
    {
```

```
        public static void Main(string[] args)
        {
            string[] arrS1 = new string[0];
            arrS1 = new String[]{ "Nguyen An", "Tran Binh", "Ly
Chau",};

            //Properties
            //Kiểm tra mảng cố định số phần tử
            Console.WriteLine("1." + arrS1.IsFixedSize); //True
            //Kiểm tra mảng chỉ đọc
            Console.WriteLine("2." + arrS1.IsReadOnly); //False
            //Số phần tử của mảng
            Console.WriteLine("3." + arrS1.Length); //3
            //Số chiều của mảng
            Console.WriteLine("4." + arrS1.Rank); //1

            //Methods trong Array class
            //Clear(): Xóa mảng
            Array.Clear(arrS1, 2, 1);//Xóa từ index thứ 2, xóa 1 phần
tử
            Console.WriteLine("5.");
            InMang(arrS1); //In phần tử rỗng

            //LastIndexOf(): Lấy vị tríphần tử cuối xuất hiện trong
mảng                            //
            arrS1[2] = "Nguyen An";// Gán giá trị tại phần tử cuối bằng
phần tử đầu
            Console.WriteLine("6." + Array.LastIndexOf(arrS1, "Nguyen
An")); //2

            //Copy(): Sao chép mảng
            string[] arrS2 = new string[arrS1.Length];
            Array.Copy(arrS1, arrS2, arrS1.Length);
            Console.WriteLine("7.");
            InMang(arrS2);

            //Clone(): Tạo bản sao của mảng
            string[] arrS3 = (string[])arrS1.Clone();
            Console.WriteLine("8.");
            InMang(arrS3);

            //So sánh hai mảng bằng Equals
            Console.WriteLine("9."+(arrS1 == arrS2));
            String[] arrS4 = arrS1; // cùng tham chiếu
            Console.WriteLine("10."+Array.Equals(arrS1, arrS4));//True
            Console.WriteLine("11."+(arrS1 == arrS4)); //True
            Console.WriteLine("12."+Array.Equals(arrS1, arrS2));//False
            Console.WriteLine("13."+Array.Equals(arrS1, arrS3));//False

            //Sort(): Sắp xếp mảng
```

```
            Array.Sort(arrS3);
            Console.WriteLine("14.");
            InMang(arrS3);
            //Reverse(): Đảo ngược mảng
            Array.Reverse(arrS3);
            Console.WriteLine("15.");
            InMang(arrS3);
            //Thay đổi số phần tử của mảng
            Array.Resize(ref arrS1, 10);
            Console.WriteLine("So phan tu S1 = " + arrS1.Length);
            Console.WriteLine("So phan tu S4 = " + arrS4.Length);
            arrS4 = arrS1;
            Console.WriteLine("So phan tu S4 =" + arrS4.Length);
        }

        //In mảng
        public static void InMang(string[] arr)
        {
            foreach (var item in arr)
            {
                Console.Write(item + " - ");
            }
            Console.WriteLine();
        }
    }
}
```

Kết quả :

```
1.True
2.False
3.3
4.1
5.
Nguyen An - Tran Binh -  -
6.2
7.
Nguyen An - Tran Binh - Nguyen An -
8.
Nguyen An - Tran Binh - Nguyen An -
```

```
9.False
10.True
11.True
12.False
13.False
14.
Nguyen An - Nguyen An - Tran Binh -
15.
Tran Binh - Nguyen An - Nguyen An -
So phan tu S1 = 10
So phan tu S4 = 3
So phan tu S4 =10
Press any key to continue . . .
```

## 1.8| Sơ Đồ lớp

Class được mô tả gồm tên Class, thuộc tính và phương thức.

| Class name (tên lớp) |
|---|
| Attributes (Các thuộc tính / dữ liệu) |
| Methods (Các phương thức/ hành động) |

Ví dụ: class SinhVien

| **Student** |
|---|
| - studentName: string<br>- idNumber: int<br>- birthday: DateTime<br>- address: string<br>+ numOfStudent : static int |
| + Student()<br>+ Student(string, int, DateTime, string)<br>+ GetAge(): int |

| |
|---|
| +CreateEmail(): string<br>+ Print(): void |

## 1.9| Một số ngôn ngữ hỗ trợ lập trình Hướng đối tượng

Xuất phát từ tư tưởng của ngôn ngữ SIMULA67, trung tâm nghiên cứu Palo Alto (PARC) của hãng XEROR đã tập trung nghiên cứu để hoàn thiện ngôn ngữ OOP đầu tiên với tên gọi là Smalltalk. Sau đó các ngôn ngữ OOP lần lượt ra đời như Eiffel, Clos, Loops, Flavors, Object Pascal, Object C, C++, Delphi, Java…

Chính XEROR trên cơ sở ngôn ngữ OOP đã đề ra tư tưởng “icon base screen interface”, kể từ đó Apple Macintosh cũng như Microsoft Windows phát triển giao diện đồ họa như ngày nay. Trong Microsoft Windows, tư tưởng OOP được thể hiện một cách rõ nét nhất đó là "click vào đối tượng", mỗi đối tượng có thể là control menu, control menu box, menu bar, scroll bar, button, minimize box, maximize box, … sẽ đáp ứng công việc tùy theo đặc tính của đối tượng. Turbo Vision của hãng Borland là một ứng dụng OOP tuyệt vời, giúp lập trình viên không quan tâm đến chi tiết của chương trình giao diện mà chỉ cần thực hiện các nội dung chính của vấn đề.

## 1.10| Bài tập áp dụng

# BÀI TẬP THỰC HÀNH SỐ 1

## I. Thông tin chung:

- Mã số bài tập : HW1-LTHDT
- Hình thức nộp bài : Nộp qua hệ thống EL môn học
- Thời hạn nộp bài : … / … / ……
- Nội dung : Chương 1: Giới thiệu lập trình hướng đối tượng

**Chuẩn đầu ra cần đạt:**

| | |
|---|---|
| L.O.1 | Phân tích, thiết kế chương trình theo hướng đối tượng. Cài đặt được sơ đồ lớp cho các chương trình vừa và nhỏ; |
| L.O.8 | Làm bài tập và nộp bài đúng quy định. |

**Sinh viên thực hành các yêu cầu sau:**

## BÀI TẬP CƠ BẢN

### Bài 1. Sử dụng kiểu dữ liệu struct để viết ứng dụng sau:

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

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

Xây dựng và thực thi các chức năng sau:

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

### Bài 2. Sử dụng kiểu dữ liệu struct để viết ứng dụng sau:

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 <u>Gói cước</u> gồm:***

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

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

**Yêu cầu:**

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

**Bài 3. Sử dụng thư viện String để viết chương trình chứa các hàm sau:**

1. Hàm đếm số ký tự là chữ hoa có trong chuỗi.
2. Hàm đếm số ký tự là chữ số có trong chuỗi.
3. Hàm đếm số ký tự không phải là chữ số có trong chuỗi
4. Hàm cho phép kiểm tra Chuỗi có tồn tại ký tự x hay không, biết x do người dùng nhập vào từ bàn phím.
5. Hàm đảo ngược chuỗi.
6. Hàm đếm số từ trong một chuỗi.

**Bài 4. Sử dụng thư viện DateTime viết chương trình chứa các hàm sau:**

1. Hàm tính ngày trước hoặc sau của một ngày
2. Hàm trả trả về chuỗi thứ trong tuần của ngày hiện tại.
3. Hàm kiểm tra xem năm hiện tại có phải năm nhuận không
4. Hàm tính số ngày của tháng hiện tại
5. Hàm tính số khoảng cách giờ giữa hai giá trị ngày giờ

**Bài 5. Sinh viên trả lời các câu hỏi sau**

a. Giải thích sự khác nhau của phương pháp lập trình hướng đối tượng và phương pháp lập trình hướng thủ tục

b. Đối tượng là gì? Lớp là gì? Phân biệt lớp và đối tượng
c. Nêu các thành phần của một lớp và cách xác định chúng.

**Bài 6. Hãy xác định thuộc tính và hành vi (phương thức) của các lớp đối tượng**

1. Sinh viên (Student)
2. Lớp học (Class)
3. Cuốn sách (Book)
4. Nhân viên (Employee)
5. Tài khoản ngân hàng (Bank Account)
6. Điện thoại di động (Mobile)
7. Thời gian (Time)
8. Phân số (Fraction)
9. Xe hơi (Car)
10. Điểm trên trục tọa độ (Point)

## BÀI TẬP NÂNG CAO

**Bài 7. Sử dụng kiểu dữ liệu struct để viết ứng dụng sau:**

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.

**Bài 8. Sử dụng thư viện String để viết chương trình chứa các hàm sau:**

1. Hàm xóa khoảng trắng thừa trong chuỗi. Biết chuỗi không chứa khoảng trắng thừa là chuỗi không có khoảng trắng đầu hoặc cuối chuỗi, hai từ chuỗi chỉ cách nhau tối đa một khoảng trắng.
2. 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.
3. Hàm tạo email từ chuỗi họ tên của người dùng. Biết email được tạo bằng cách xoá các khoảng trắng thừa trong chuỗi và thêm chuỗi "@tdc.edu.vn".
4. 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.
5. Hàm kiểm tra chuỗi họ tên một người có hợp lệ hay không. Biết chuỗi hợp lệ là chuỗi có chiều dài tối thiểu 3 ký tự, không có khoảng trắng thừa trong chuỗi, bắt đầu bằng ký tự chữ hoa, các từ sau viết hoa đầu từ

**Bài 9. Viết chương trình theo mô tả sau:**

Chương trình cho phép người dùng nhập vào một chuỗi họ tên. Chương trình tạo username và password tự động cho người dùng bằng các 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 du: Họ tên: Nguyen Van Anh, username: AnhNguyen, password: nva261782

**Bài 10. Viết chương trình theo mô tả sau:**

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.

**Bài 11.** 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, trên 3 ký tự, không chứa khoảng trắng thừa, viết hoa đầu từ
- Ngày sinh: DateTime, tính tới thời điểm hiện tại nhân viên phải từ 18 tuổi trở lên.
- Lương cơ bản: double, tối thiểu là 3.000.000
- Hệ số lương: double

Các chức năng của chương trình:

1. Nhập danh sách nhân viên bao gồm các thông tin cá nhân theo các ràng buộc dữ liệu được mô tả ở trên
2. Đếm số lượng nhân viên có mức lương cao hơn mức lương trung bình của các nhân viên trong danh sách
3. Thống kê số lượng nhân viên có họ "Nguyen"
4. Liệt kê danh sách họ tên nhân viên có sinh nhật trong một tháng x. Với x là số tháng do người dùng nhập vào.
5. Sắp xếp danh sách họ tên nhân viên trong câu 4 theo thứ tự giảm dần.

**Bài 12.** **Hãy vẽ sơ đồ lớp của các lớp cho 3 đối tượng bất kỳ**

**----- Hết-----**