LAB 2. HÀM VÀ MỘT SỐ KIỂU DỮ LIỆU CƠ BẢN

THỜI LƯỢNG : 4 TIẾT

A. Mục tiêu

- Giúp sinh viên học cách định nghĩa và sử dụng kiểu liệt kê, kiểu dữ liệu chuỗi
- Học cách định nghĩa và gọi hàm (phương thức) trong ngôn ngữ C#
- Sau khi hoàn thành bài thực hành này, sinh viên cần:
  - Phân biệt rõ lời gọi hàm và định nghĩa hàm
  - Biết cách định nghĩa hàm theo các yêu cầu khác nhau
  - Phân biệt được tham số và đối số
  - Phân biệt được cách truyền tham trị và truyền tham biến
  - Hiểu rõ và cài đặt được các hàm đệ quy
  - Biết cách vận dụng hàm đệ quy và khử đệ quy
  - Sử dụng được kiểu liệt kê, các lệnh điều khiển và hàm để tạo menu cho chương trình

B. Yêu cầu

- Sinh viên phải trả lời đầy đủ các câu hỏi (nếu có) hoặc chụp màn hình kết quả, ghi lại vào tập tin Word theo yêu cầu ghi trong phần hướng dẫn thực hành.
- Đặt tên tập tin Word theo dạng: Lab02_MSSV_HoVaTen.docx
- Tạo thư mục, đặt tên là MSSV_Lab02 để lưu tất cả bài làm trong bài thực hành này.
- Sinh viên phải hoàn thành tối thiểu 5 bài tập bắt buộc.
- Phải tạo một dự án riêng cho mỗi phần hoặc mỗi bài tập.
- Cuối buổi thực hành, nén thư mục trên & nộp bài qua email: phucnguyen5555@gmail.com

C. Hướng dẫn thực hành

1. Định nghĩa (tạo) hàm đơn giản

Phần này minh họa cách tạo các hàm đơn giản. Từ hàm không có tham số, không có giá trị trả về đến hàm có nhiều tham số, có giá trị trả về.

Bước 1. Tạo một thư mục mới, đặt tên theo dạng MSSV_Lab02. Ví dụ: 1210234_Lab02.

Bước 2. Tạo một dự án mới, đặt tên là Lab02_Bai01_HamDonGian và lưu vào thư mục trên

Bước 3. Mở tập tin Program.cs, trước hàm Main, nhập đoạn mã sau để định nghĩa một hàm.

```
// Định nghĩa hàm xuất một chuỗi ra màn hình
// Hàm này không có tham số, không trả về giá trị
static void XuatThongBao()
{
    Console.WriteLine("Thong bao: ban dang doc thong bao :)");
}
```

Bước 4. Trong hàm Main, nhập đoạn mã sau để gọi hàm vừa tạo

```
// Lời gọi hàm xuất thông báo
XuatThongBao();

Console.ReadKey();
```

Bước 5. Chạy chương trình và ghi nhận lại kết quả

Bước 6. Trước hàm Main, tiếp tục định nghĩa hàm xuất câu chào mừng như sau

```
// Hàm này có tham số, không trả về giá trị
// hoTen nằm trong dấu ngoặc gọi là tham số của hàm
static void XuatCauChao(string hoTen)
{
    Console.WriteLine("Xin chao ban " + hoTen);
    // Hoặc viết
    // Console.WriteLine("Xin chao ban {0}", hoTen);
}
```

Bước 7. Trong hàm Main, thay đoạn mã trong bước 4 bởi đoạn mã sau để gọi hàm vừa tạo

```
// Lời gọi hàm xuất câu chào. Giá trị được đặt
// trong cặp dấu ngoặc tròn gọi là đối số của hàm
XuatCauChao("Nguyen Van Phuc");

string tenCuaToi = "Nguyen Van Phuc";
XuatCauChao(tenCuaToi);
```

Bước 8. Chạy chương trình và ghi nhận lại kết quả

Bước 9. Trước hàm Main, tiếp tục định nghĩa hàm nhập một số nguyên như sau

```
// Hàm này không có tham số nhưng có trả về giá trị
static int NhapSoNguyen()
{
    int number = 0;
    number = int.Parse(Console.ReadLine());
    return number;
}
```

Bước 10. Trong hàm Main, thay đoạn mã trong bước 7 bởi đoạn mã sau để gọi hàm vừa tạo

```
// Lời gọi hàm nhập số nguyên
int n = NhapSoNguyen();
Console.WriteLine("So nguyen vua nhap : {0}", n);
```

Bước 11. Chạy chương trình và ghi nhận lại kết quả

Bước 12. Trước hàm Main, tiếp tục định nghĩa hàm nhập một số thực trong miền giá trị cho trước như sau:

```
// Hàm này có tham số và có trả về giá trị
static double NhapSoThuc(double min, double max)
{
    double number = 0;
    do
    {
        number = double.Parse(Console.ReadLine());
    } while (number < min || number > max);

    return number;
}
```

Bước 13. Trong hàm Main, thay đoạn mã trong bước 10 bởi đoạn mã sau để gọi hàm vừa tạo

```
// Lời gọi hàm nhập số thực trong đoạn 10.5 đến 19.1
// Truyền giá trị trực tiếp làm đối số
double soThuc = NhapSoThuc(10.5, 19.1);

Console.WriteLine("So thuc vua nhap: {0}", soThuc);

// Khai báo biến và truyền biến làm đối số của hàm
double nho = 10.5, lon = 19.1;
soThuc = NhapSoThuc(nho, lon);

Console.WriteLine("So thuc vua nhap: {0}", soThuc);
```

Bước 14. Chạy chương trình và ghi nhận lại kết quả

2. Truyền tham trị và truyền tham chiếu (tham biến)

Phần này minh họa cách gọi hàm và truyền đối số vào hàm theo hai cách: truyền tham trị và truyền tham chiếu. Khi truyền tham trị, giá trị của đối số sẽ không bị thay đổi sau khi ra khỏi hàm. Ngược lại, khi truyền tham chiếu (sử dụng từ khóa ref hoặc out), nếu trong hàm có thay đổi giá trị của đối số thì giá trị đó sẽ được giữ lại khi ra khỏi hàm.

Bước 1. Tạo dự án mới, đặt tên là Lab02_Bai02_TruyenThamSo và lưu vào thư mục đã tạo

Bước 2. Mở tập tin Program.cs, trước hàm Main, nhập đoạn mã sau để định nghĩa một hàm.

```
// Truyền tham trị
// Hàm này sẽ không làm thay đổi giá trị biến x
static void NhanDoi(double x)
{
    x *= 2;
}
```

Bước 3. Trong hàm Main, nhập đoạn mã sau để gọi hàm vừa tạo

```
// Khai báo và khởi tạo một biến số thực
double so = 10;

// Lời gọi hàm nhân đôi
NhanDoi(so);
Console.WriteLine("Khong co ref: so = {0}", so);

Console.ReadKey();
```

Bước 4. Chạy chương trình và cho biết kết quả. Giá trị của biến "so" trước và sau khi gọi hàm có giống nhau hay không? Tại sao?

Bước 5. Tiếp tục định nghĩa một hàm mới như sau:

```
// Truyền tham chiếu
// Hàm này sẽ làm thay đổi giá trị biến x
static void GapDoi(ref double x)
{
    x *= 2;
}
```

Bước 6. Thay đổi lời gọi hàm trong bước 3 bởi đoạn mã sau để gọi hàm GapDoi (gấp đôi).

```
GapDoi(ref so);
Console.WriteLine("Co dung ref : so = {0}", so);
```

Bước 7. Chạy chương trình và ghi nhận kết quả. Giá trị của biến "so" trước và sau khi gọi hàm có giống nhau hay không? Có nhận xét gì về 2 hàm NhanDoi và GapDoi cũng như lời gọi hai hàm đó?

Bước 8. Tiếp tục định nghĩa hai hàm khác để thực hiện việc hoán vị hai số nguyên như sau:

```
// Truyền tham trị
// Hàm hoán vị hai số nguyên
static void HoanDoi(int x, int y)
{
    int tam = x;
    x = y;
    y = tam;
}
```

```
// Truyền tham chiếu
static void HoanVi(ref int x, ref int y)
{
    int tam = x;
    x = y;
    y = tam;
}
```

Bước 9. Thay đoạn mã trong bước 6 bởi đoạn mã sau để gọi hàm HoanDoi và HoanVi

```
// Khai báo và khởi tạo hai biến số nguyên
int p = 10, q = 15;
Console.WriteLine("Ban dau : p = {0}, q = {1}", p, q);

// Lời gọi hàm Hoán vị
HoanDoi(p, q);
Console.WriteLine("Hoan vi khong dung ref : p = {0}, q = {1}", p, q);

HoanVi(ref p, ref q);
Console.WriteLine("Hoan vi co dung ref : p = {0}, q = {1}", p, q);
```

Bước 10. Chạy chương trình và cho biết kết quả. Hàm nào thực sự hoán vị 2 số nguyên?

Bước 11. Từ khóa ref bắt buộc đối số phải được gán giá trị trước khi truyền vào hàm. Tuy nhiên, có những trường hợp, giá trị này ta chưa biết. C# cung cấp một từ khóa khác để giải quyết vấn đề này. Trong các bước tiếp theo, ta sẽ định nghĩa một hàm để kiểm tra số a có chia hết cho số b hay không. Nếu có, cho biết thương của phép chia là bao nhiêu. Vì kết quả cuối cùng phải là 2 giá trị (chia hết hay không và thương số) nhưng hàm chỉ trả về một giá trị nên ta phải dùng thêm một tham số. Hàm này được định nghĩa như sau:

```
// Hàm kiểm tra phép chia hết
// Minh họa cách sử dụng từ khóa out
static bool ChiaHet(int soBiChia, int soChia, out int thuong)
{
    thuong = soBiChia / soChia;
    return soBiChia == thuong * soChia;
}
```

Bước 12. Trong hàm Main, thay đoạn mã ở bước 9 bởi đoạn mã sau:

```
// Lời gọi hàm Chia hết
int a = 20, b = 4, t;
bool kq = ChiaHet(a, b, out t);

if (kq)
{
    Console.WriteLine("{0} chia het cho {1}", a, b);
    Console.WriteLine("Thuong cua phep chia la : {0}", t);
}
```

Bước 13. Chạy chương trình và ghi nhận kết quả.

Bước 14. Thử gán t = 1 rồi chạy lại chương trình. So sánh kết quả với lần chạy trước.

Bước 15. Nếu không gán giá trị cho t và thay out bằng ref ở cả phần định nghĩa hàm và lời gọi hàm thì kết quả chương trình là gì?

3. Hàm đệ quy

Phần này hướng dẫn cách định nghĩa và sử dụng các hàm đệ quy để tính giai thừa và tìm số Fibonacci. Sau đó khử đệ quy bằng cách sử dụng các lệnh lặp.

Bước 1. Tạo dự án mới, đặt tên là Lab02_Bai03_HamDeQuy và lưu vào thư mục đã tạo

Bước 2. Mở tập tin Program.cs, định nghĩa hàm tính giai thừa bằng cách dùng đệ quy như sau

```
// Hàm tính giai thừa của một số
// N! = 1.2.3...(N-1).N = (N-1)!.N
static long GiaiThuaDeQuy(int n)
{
    if (n < 2) return 1;
    else return n * GiaiThuaDeQuy(n - 1);
}
```

Bước 3. Trong hàm Main, bổ sung đoạn mã sau để gọi hàm tính 10!

```
// Khai báo biến số nguyên n
int so = 10;

// Lời gọi hàm tính giai thừa
long gtn = GiaiThuaDeQuy(so);
Console.WriteLine("{0}! = {1}", so, gtn);
```

Bước 4. Chạy chương trình và cho biết kết quả.

Bước 5. Sau đoạn mã ở bước 3, hãy viết mã lệnh để xuất giai thừa của các số từ 0 tới 10.

Bước 6. Tiếp theo ta sẽ khử đệ quy bằng cách sử dụng vòng lặp. Bổ sung hàm sau:

```
// Khử đệ quy phương thức tính giai thừa
static long GiaiThuaLap(int n)
{
    if (n < 2) return 1;
    long tich = 1;
    for (int i = 2; i <= n; i++)
    {
        tich *= i;
    }
    return tich;
}
```

Bước 7. Thay thế lời gọi hàm GiaiThuaDeQuy trong hàm Main bằng hàm GiaiThuaLap. Chạy chương trình và ghi nhận kết quả. So với kết quả ở bước 4 có khác gì hay không?

Để hiểu rõ hơn về hàm đệ quy, ta tiếp tục với một ví dụ khác. Đó là số Fibonacci, chúng được định nghĩa như sau: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$. Bài toán đặt ra là tìm số Fibonacci thứ n và xuất n số Fibonacci đầu tiên.

Bước 8. Định nghĩa hàm tính số Fibonnaci thứ n theo dạng đệ quy như sau

```
// Hàm tìm số Fibonacci thứ n
// F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)
static long FiboDeQuy(int n)
{
    if (n <= 0) return 0;
    else if (n == 1) return 1;
    else return FiboDeQuy(n - 1) + FiboDeQuy(n - 2);
}
```

Bước 9. Trong hàm Main, thay thế đoạn mã ở bước 7 bởi đoạn mã sau:

```
// Khai báo biến số nguyên n
int so = 10;

// Lời gọi hàm tính số Fibonacci
long fibo = FiboDeQuy(so);
Console.WriteLine("Fibonacci({0}) = {1}", so, fibo);

// In n số Fibonacci đầu tiên
for (int i = 0; i <= so; i++)
{
    fibo = FiboDeQuy(i);
    Console.WriteLine("Fibonacci({0}) = {1}", i, fibo);
}
```

Bước 10. Chạy chương trình và ghi nhận kết quả.

Bước 11. Tiếp theo, ta sẽ khử đệ quy hàm tính số Fibonacci bằng cách dùng vòng lặp. Định nghĩa một hàm mới như sau:

```
// Khử đệ quy hàm tính số Fibonacci thứ n
static long FiboLap(int n)
{
    if (n <= 0) return 0;
    if (n == 1) return 1;

    // Trường hợp n >= 2
    long truoc = 0, giua = 1, buoc = 2, sau = 0;
    while (buoc <= n)
    {
        sau = truoc + giua;
```

```
            truoc = giua;
            giua = sau;
            buoc++;
        }
        return sau;
    }
```

Bước 12. Trong hàm Main, thay thế lời gọi hàm FiboDeQuy trong bước 9 bởi lời gọi hàm FiboLap. Chạy lại chương trình và so sánh kết quả với bước 10.

4. Các thao tác trên mảng một chiều

Phần này hướng dẫn cách định nghĩa các hàm tính toán hay thực hiện các thao tác cơ bản trên mảng một chiều chứa các số nguyên. Sinh viên xem lại Lab01_Bai06 để cài đặt các hàm còn để trống.

Bước 1. Tạo dự án mới, đặt tên là Lab02_Bai04_CacThaoTacMang và lưu vào thư mục đã tạo

Bước 2. Mở tập tin Program.cs, trước hàm Main, bổ sung đoạn mã sau:

```
// Khai báo và khởi tạo một mảng một chiều
// chứa tối đa 1000 số nguyên, đặt tên là mang
static int[] mang = new int[1000];

// Khai báo và khởi tạo biến lưu số lượng
// phần tử đã có trong mảng
static int soLuong = 0;
```

Bước 3. Tiếp theo, định nghĩa hàm nhập giá trị (từ bàn phím) cho các phần tử của mảng:

```
// Định nghĩa hàm nhập giá trị cho các phần tử
// của mảng từ bàn phím
static void NhapMang()
{
    Console.Write("Nhap so luong phan tu cua mang : ");
    soLuong = int.Parse(Console.ReadLine());

    for (int i = 0; i < soLuong; i++)
    {
        Console.Write("mang[{0}] = ", i);
        mang[i] = int.Parse(Console.ReadLine());
    }
}
```

Bước 4. Trong hàm Main, nhập đoạn mã sau để gọi hàm NhapMang. Sau đó, chạy chương trình để xem và ghi nhận lại kết quả

```
// Gọi hàm nhập bằng tay
NhapMang();

Console.ReadKey();
```

Bước 5. Cài đặt hàm XuatMang để xuất các phần tử của mảng ra màn hình như sau:

```
static void XuatMang()
{
    Console.WriteLine("Cac phan tu cua mang : ");
    for (int i = 0; i < soLuong; i++)
    {
        Console.Write("{0}\t", mang[i]);
    }
    Console.WriteLine();
}
```

Bước 6. Trong hàm Main, sau lời gọi hàm NhapMang(); bổ sung lời gọi hàm XuatMang();

Bước 7. Chạy chương trình, ghi nhận lại kết quả và mô tả lại hoạt động của chương trình.

Bước 8. Để tránh mất thời gian vì mỗi lần chạy chương trình phải nhập các phần tử của mảng, ta có thể định nghĩa hàm nhập giá trị ngẫu nhiên như sau:

```
static void NhapNgauNhien()
{
    Console.Write("Nhap so luong phan tu cua mang : ");
    soLuong = int.Parse(Console.ReadLine());

    // Tạo bộ sinh số ngẫu nhiên
    Random rd = new Random();

    for (int i = 0; i < soLuong; i++)
    {
        // Sinh ra một số ngẫu nhiên trong đoạn [0..999]
        // rồi gán số đó vào phần tử i của mảng
        mang[i] = rd.Next(1000);
    }
}
```

Bước 9. Thay lời gọi hàm NhapMang() trong bước 4 thành lời gọi hàm NhapNgauNhien();

Bước 10. Chạy chương trình ít nhất 3 lần và cho biết kết quả có giống nhau hay không?

Bước 11. Tiếp theo, định nghĩa hàm tính tổng các phần tử của mảng như sau:

```
static int TinhTong()
{
    int sum = 0;
    for (int i = 0; i < soLuong; i++)
        sum += mang[i];
    return sum;
}
```

Bước 12. Trong hàm Main, cập nhật lại đoạn mã như sau để gọi hàm tính tổng

```
// Gọi hàm nhập ngẫu nhiên
NhapNgauNhien();

// Gọi hàm xuất mảng
XuatMang();

// Gọi hàm tính tổng
int tong = TinhTong();
Console.WriteLine("Tong cac phan tu cua mang la : {0}", tong);
```

Bước 13. Chạy chương trình và ghi nhận lại kết quả.

Bước 14. Tiếp tục cài đặt các hàm sau và bổ sung lời gọi hàm tương ứng vào hàm Main

```
// Định nghĩa hàm tìm phần tử lớn nhất trong mảng
static int TimSoLonNhat()...

// Định nghĩa hàm tìm vị trí của phần tử X cho trước
// Nếu tìm thấy, trả về vị trí, nếu không, trả về -1
static int TimViTri(int x)...

// Định nghĩa hàm tìm vị trí của phần tử lớn nhất
static int TimViTriMax()...

// Định nghĩa hàm xóa phần tử nằm tại vị trí cho trước
static void XoaTaiViTri(int viTri)...

// Định nghĩa hàm xóa phần tử có giá trị x
static bool XoaPhanTuX(int x)...
```

Bước 15. Chạy chương trình và ghi nhận lại kết quả.

5. Kiểu liệt kê (enum) và tạo menu

Phần này hướng dẫn cách tạo một menu chức năng, cho phép người dùng chọn chức năng và chương trình sẽ gọi hàm xử lý tương ứng. Để tạo menu, trước tiên, ta sẽ tạo một kiểu liệt kê cho biết các chức năng mà chương trình có thể thực hiện. Ta sẽ tiếp tục với dự án Lab02_Bai04.

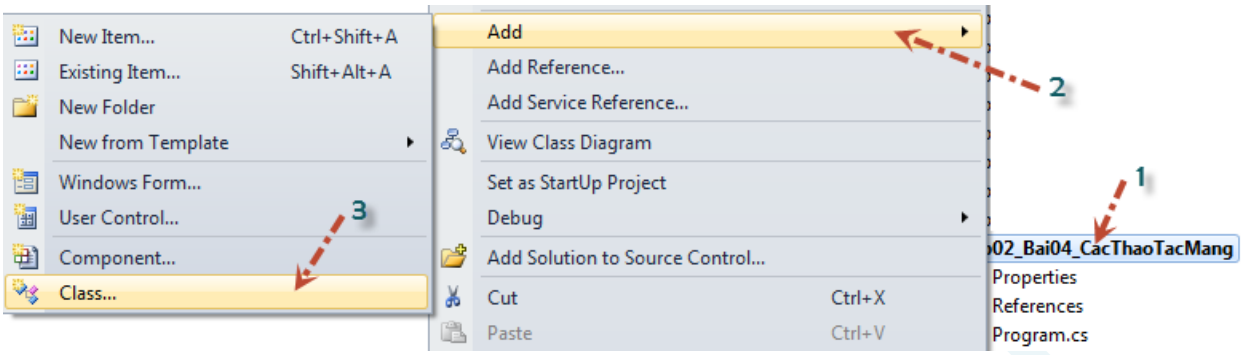
Bước 1. Trong dự án Lab02_Bai04_CacThaoTacMang, mở cửa sổ Solution Explorer, nhắp phải chuột vào tên dự án, chọn Add > Class … để mở hộp thoại Add New Item

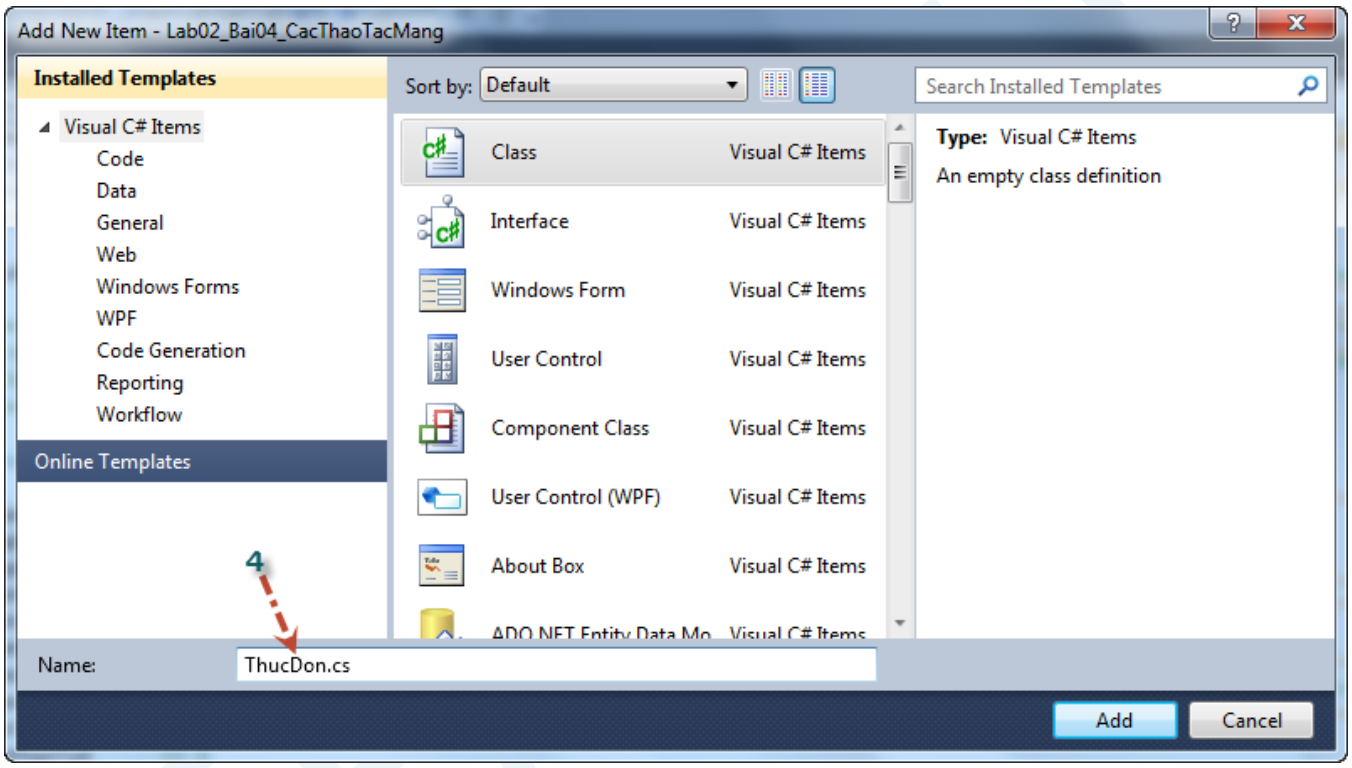
Bước 2. Trong mục Name, đặt tên tập tin là ThucDon.cs

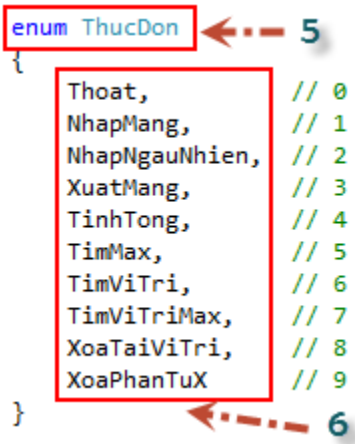
Bước 3. Trong cửa sổ soạn thảo, thay đổi class ThucDon thành enum ThucDon.

Bước 4. Bổ sung các trường vào kiểu liệt kê (enum) ThucDon như hình (cuối trang sau)

Mỗi trường của enum ThucDon tương ứng với 1 chức năng mà chương trình sẽ thực hiện.







Bước 5. Trong lớp Program, cài đặt hàm in các chức năng của chương trình ra màn hình như sau:

```
// Định nghĩa hàm xuất thực đơn ra màn hình
static void InMenu()
{
    Console.WriteLine("====================== MENU ======================");
    Console.WriteLine("{0}. Nhap mang bang tay", (int)ThucDon.NhapMang);
    Console.WriteLine("{0}. Nhap ngau nhien", (int)ThucDon.NhapNgauNhien);
    Console.WriteLine("{0}. Xuat cac phan tu cua mang", (int)ThucDon.XuatMang);
    Console.WriteLine("{0}. Tinh tong cac phan tu", (int)ThucDon.TinhTong);
    Console.WriteLine("{0}. Tim phan tu lon nhat", (int)ThucDon.TimMax);
    Console.WriteLine("{0}. Tim vi tri cua phan tu X", (int)ThucDon.TimViTri);
    Console.WriteLine("{0}. Tim vi tri phan tu lon nhat", (int)ThucDon.TimViTriMax);
    Console.WriteLine("{0}. Xoa phan tu tai vi tri cho truoc", (int)ThucDon.XoaTaiViTri);
    Console.WriteLine("{0}. Xoa phan tu X khoi mang", (int)ThucDon.XoaPhanTuX);
    Console.WriteLine("{0}. Thoat", (int)ThucDon.Thoat);
    Console.WriteLine("==================================================");
}
```

Bước 6. Cài đặt hàm cho phép người dùng chọn một chức năng của chương trình

```
// Định nghĩa hàm chọn một thực đơn
static ThucDon ChonMenu()
{
    // Lấy số lượng menu
    int soMenu = Enum.GetNames(typeof (ThucDon)).Length;

    // Khai báo biến để lưu số thứ tự menu mà người dùng chọn
    int menu = 0;
    do
    {
        InMenu();       // Lời gọi hàm xuất danh sách chức năng
        Console.Write("Nhap so de chon menu (0..{0}) : ", soMenu);

        menu = int.Parse(Console.ReadLine());
    } while (menu < 0 || menu >= soMenu);

    return (ThucDon) menu;
}
```

Bước 7. Tiếp theo, cài đặt hàm xử lý ứng với từng chức năng được chọn

```
// Định nghĩa hàm xử lý chức năng tương ứng với menu đã chọn
static void XuLy(ThucDon menu)
{
    switch (menu)
    {
        case ThucDon.NhapMang: NhapMang(); break;
        case ThucDon.NhapNgauNhien: NhapNgauNhien(); break;
        case ThucDon.XuatMang: XuatMang(); break;
        default:  break;
    }
}
```

Bước 8. Cuối cùng, định nghĩa hàm để lặp lại việc chọn các chức năng và xử lý menu

```
// Định nghĩa hàm cho phép người dùng chọn menu
// và thực hiện các chức năng tương ứng
static void ChayChuongTrinh()
{
    ThucDon menu = ThucDon.Thoat;
    do
    {
        menu = ChonMenu();
        XuLy(menu);
    } while (menu != ThucDon.Thoat);
}
```

Bước 9. Cập nhật lại hàm Main như sau:

```
static void Main(string[] args)
{
    ChayChuongTrinh();
    Console.ReadKey();
}
```

Bước 10. Chạy chương trình, chọn các chức năng 0, 1, 2 hoặc 3 để xem kết quả.

Bước 11. Mô tả lại hoạt động của chương trình.

Bước 12. Trong hàm XuLy(menu), bổ sung các mệnh đề case để gọi các hàm tương ứng với các chức năng (các trường trong enum ThucDon) khác của chương trình.

6. Kiểu dữ liệu chuỗi và các thao tác cơ bản trên chuỗi

Phần này minh họa cách khai báo, khởi tạo các biến kiểu chuỗi (string) và sử dụng các hàm cơ bản đã được định nghĩa để thực hiện các thao tác trên chuỗi như tính chiều dài, ghép chuỗi, thay thế chuỗi, lấy chuỗi con, …

Bước 1. Tạo một dự án mới, đặt tên là Lab02_Bai05_CacThaoTacTrenChuoi.

Bước 2. Trong tập tin Program.cs, nhập đoạn mã sau vào hàm Main.

```
// Khai báo ba chuỗi
string khoa = "Cong nghe Thong tin",
       dhoc = "  Dai hoc Da Lat",
       con = "Da";

// 1. Lấy chiều dài của chuỗi
int cd = khoa.Length;
Console.WriteLine("Chieu dai chuoi {0} la {1}", khoa, cd);
```

Bước 3. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 4. Tiếp tục bổ sung đoạn mã sau vào hàm Main

```
// 2. Nối hai chuỗi với nhau
string ketQua = khoa + dhoc;
Console.WriteLine(ketQua);

// 3. Chuyển sang chữ hoa
ketQua = khoa.ToUpper();
Console.WriteLine(ketQua);

// 4. Chuyển sang chữ thường
ketQua = dhoc.ToLower();
Console.WriteLine(ketQua);

// 5. Xóa ký tự trắng ở đầu và đuôi
ketQua = dhoc.Trim();
Console.WriteLine(ketQua);
```

Bước 5. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 6. Thay đoạn mã trong bước 4 bởi đoạn mã sau để lấy (cắt) một chuỗi con từ chuỗi "Cong nghe Thong tin"

```
// 6. Cắt 4 ký tự bên trái chuỗi khoa
ketQua = khoa.Substring(0, 4);
Console.WriteLine(ketQua);

// 7. Cắt 5 ký tự giữa chuỗi khoa, bắt đầu từ vị trí thứ 10
ketQua = khoa.Substring(10, 5);
Console.WriteLine(ketQua);
```

Bước 7. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 8. Thay đoạn mã trong bước 6 bởi đoạn mã sau để chèn hoặc thay thế các ký tự

```
// 8. Chèn chuỗi "Trường" vào đầu chuỗi dhoc
ketQua = dhoc.Insert(0, "Truong");
Console.WriteLine(ketQua);

// 9. Thay chuỗi "Dai hoc" trong dhoc thành chuỗi "University of"
ketQua = dhoc.Replace("Dai hoc", "University of");
Console.WriteLine(ketQua);
```

Bước 9. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 10. Thay đoạn mã trong bước 8 bởi đoạn mã sau để xóa các ký tự trong một chuỗi

```
// 10. Xóa 7 ký tự từ chuỗi dhoc, bắt đầu từ vị trí 2
ketQua = dhoc.Remove(2, 7);
Console.WriteLine(ketQua);
```

```
// 11. Xóa những ký tự từ vị trí thứ 10 trong chuỗi khoa
ketQua = khoa.Remove(10);
Console.WriteLine(ketQua);

// 12. Xóa 3 ký tự cuối của chuỗi khoa
ketQua = khoa.Remove(khoa.Length - 3);
Console.WriteLine(ketQua);
```

Bước 11. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 12. Thay đoạn mã trong bước 10 bởi đoạn mã sau để thực hiện việc canh lề văn bản

```
// 13. Chèn thêm các ký tự trắng vào bên trái chuỗi khoa
// cho đủ 40 ký tự (dùng khi muốn canh lề bên phải)
ketQua = khoa.PadLeft(40);
Console.WriteLine(ketQua);

// 14. Chèn thêm các dấu chấm vào bên trái chuỗi khoa
// cho đủ 40 ký tự
ketQua = khoa.PadLeft(40, '.');
Console.WriteLine(ketQua);

// 15. Chèn thêm các ký tự trắng vào bên phải chuỗi khoa
// cho đủ 40 ký tự (dùng khi muốn canh lề bên trái)
ketQua = khoa.PadRight(40);
Console.WriteLine(ketQua);

// 16. Chèn thêm các dấu + vào bên phải chuỗi khoa
// cho đủ 40 ký tự
ketQua = khoa.PadRight(40, '+');
Console.WriteLine(ketQua);

// 17. Xuất chuỗi khoa nằm giữa màn hình
ketQua = khoa.PadLeft((80 - khoa.Length) / 2);
Console.WriteLine(ketQua);
```

Bước 13. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 14. Thay đoạn mã trong bước 12 bởi đoạn mã sau để thực hiện một số phép kiểm tra

```
// 18. Kiểm tra chuỗi con có nằm trong chuỗi dhoc
bool kiemTra = dhoc.Contains(con);
if (kiemTra)
    Console.WriteLine("Chuoi {0} co chua chuoi {1}", dhoc, con);

// 19. Kiểm tra có phải chuỗi khoa bắt đầu với chuỗi "Cong"
kiemTra = khoa.StartsWith("Cong");
if (!kiemTra)
    Console.WriteLine("Chuoi {0} khong bat dau voi chuoi Cong", dhoc);
```

Bước 15. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 16. Để canh lề văn bản, ta sử dụng hàm PadLeft, PadRight hoặc sử dụng hàm string.Format. Thay đoạn mã trong bước 14 bởi đoạn mã sau:

```
// 24. Xuất dữ liệu có định dạng & canh lề
// {0,5} nghĩa là thêm các ký tự trắng vào bên trái
// của đối số thứ nhất cho đủ 5 ký tự => canh lề phải
// {1,-30} nghĩa là thêm các ký tự trắng vào bên phải
// của đối số thứ 2 cho đủ 30 ký tự => canh lề trái
string dinhDang = "{0,5}  {1,-30}{2,-10}{3,5}";

ketQua = string.Format(dinhDang, "STT", "Ho va Ten", "Lop", "Diem");
Console.WriteLine(ketQua);

ketQua = string.Format(dinhDang, 1, "Nguyen Thanh Phong", "CTK36", 5.5f);
Console.WriteLine(ketQua);

ketQua = string.Format(dinhDang, 12, "Trinh Hoa Binh", "CTK36CD", 7.25f);
Console.WriteLine(ketQua);

ketQua = string.Format(dinhDang, 100, "Phan Quoc Khanh", "CTK36LT", 6);
Console.WriteLine(ketQua);
```

Bước 17. Chạy chương trình và cho biết kết quả xuất ra màn hình.

Bước 18. Tự tìm hiểu thêm cách sử dụng các hàm StartWith, EndWith, IndexOf, LastIndexOf

D. Bài tập bắt buộc

1. Viết các hàm tính tổng diện tích các mặt và thể tích của hình hộp chữ nhật.
2. Viết hàm tính giá trị sin(x), cos(x) theo công thức sau (cho sai số là 10^{-5})

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots$$

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

3. Viết hàm tính giá trị biểu thức sau với x và n cho trước:

$$x + \frac{1}{2} \times \frac{x^3}{3} + \frac{1 \times 3}{2 \times 4} \times \frac{x^5}{5} + \frac{1 \times 3 \times 5}{2 \times 4 \times 6} \times \frac{x^7}{7} + \frac{1 \times 3 \times 5 \times 7}{2 \times 4 \times 6 \times 8} \times \frac{x^9}{9} + \cdots; \ -1 \le x \le 1$$

4. Viết hàm kiểm tra một số nguyên dương N có phải là số hoàn chỉnh (hay số hoàn hảo).
5. Cài đặt các thao tác sau trên mảng một chiều:
    a. Xóa phần tử x trong danh sách.
    b. Xóa phần tử theo vị trí trong danh sách.

c. Xóa tất cả phần tử x trong danh sách.
d. Xóa tất cả số âm trong danh sách.
e. Tìm phần tử lớn nhất trong mảng.
f. Tìm vị trí đầu tiên của phần tử lớn nhất trong mảng.
g. Xóa tất cả phần tử lớn nhất trong mảng.
h. Tìm tất cả vị trí của phần tử lớn nhất trong mảng.
i. Thay thế phần tử x bằng phần tử y trong danh sách.
j. Chèn một phần tử vào trong danh sách tại vị trí bất kì.
k. Chèn một phần tử x vào trước phần tử y trong danh sách.
l. Chèn một phần tử x vào sau phần tử y trong danh sách.
m. Đảo ngược danh sách.
n. Đếm số phần tử (không tính trùng nhau) trong dánh sách.
o. Xóa tất cả phần tử trùng nhau trong danh sách.
p. Tính trung bình cộng các phần tử trong mảng
q. Tính tổng các phần tử chỉ xuất hiện một lần trong mảng

6. Tạo thực đơn (menu) cho chương trình ở bài 5.
7. Tìm ước chung lớn nhất của 2 số nguyên a, b. Viết cả dạng lặp và đệ quy.
8. Viết hàm tính các tổng sau bằng 2 cách: lặp và đệ quy

$$A = 1 + 2 + 3 + 4 + \ldots + (n-1) + n$$
$$B = 1^2 + 2^2 + 3^2 + \ldots + (n-1)^2 + n^2$$
$$C = 1^3 + 2^3 + 3^3 + 4^3 + \ldots + (n-1)^3 + n^3$$
$$D = 1.2 + 2.3 + 3.4 + \ldots + (n-2)(n-1) + (n-1)n$$
$$E = 1.2.3 + 2.3.4 + 3.4.5 + \ldots + (n-3)(n-2)(n-1) + (n-2)(n-1)n$$
$$F = \frac{1}{1.2} + \frac{1}{2.3} + \ldots + \frac{1}{(n-1)n}$$
$$G = \frac{1}{1.2.3} + \frac{1}{2.3.4} + \frac{1}{3.4.5} + \ldots + \frac{1}{(n-2)(n-1)n}$$
$$H = 2 + 4 + 6 + \ldots + (2n-4) + (2n-2) + 2n$$
$$I = 1 + 3 + 5 + \ldots + (2n-3) + (2n-1) \text{ với } (n \geq 2)$$

9. Viết hàm tìm tất cả các số có 3 chữ số abc sao cho abc = $a^3 + b^3 + c^3$.
10. Trò chơi đoán số: Viết chương trình cho máy tự sinh ra một số ngẫu nhiên trong khoảng từ 0 đến 1000. Người dùng sẽ đoán số bằng cách nhập số đó vào máy. Nếu số đoán nhỏ hơn số máy tạo ra, chương trình sẽ xuất ra câu "Nhỏ hơn". Nếu số đoán lớn hơn số máy tạo ra, chương trình sẽ in ra câu "Lớn hơn". Nếu số đoán trùng với số máy tạo ra, người chơi thắng. Trong trường hợp quá k lần đoán nhưng người chơi vẫn sai thì chương trình dừng và thông báo người chơi thua.

E. Bài tập làm thêm

1. Viết hàm tính giá trị hàm Ackerman theo định nghĩa sau

   Cho hàm A(x, y), với miền xác định R^2 và miền giá trị là R

   - A (0, y) = 1, nếu y ≥ 0
   - A (1, 0) = 2
   - A (x, 0) = x + 2, nếu x ≥ 0
   - A (x, y) = A (A (x - 1, y), y - 1), nếu x ≥ 1 và y ≥ 1

2. Viết hàm tính giá trị hàm e^x theo công thức sau

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

3. Số định danh sách chuẩn quốc tế (ISBN) là một con số có 10 chữ số. Trong đó, chữ số cuối cùng được xác định bởi 9 chữ số trước nó. Ngoài ra, $d_1 + 2*d_2 + 3*d_3 + \ldots + 10*d_{10}$ phải là một số chia hết cho 11, trong đó d_i là chữ số ở vị trí thứ i tính từ trái sang phải. Viết hàm cho phép nhập vào 9 chữ số đầu và in ra chữ số cuối cùng.
4. Cho hai số x và n. Viết hàm tính giá trị biểu thức x^n / n!
5. Viết hàm chuyển một số sang chuỗi tương ứng. Ví dụ: từ số 12345 sang chuỗi "12345".
6. Viết hàm đảo ngược các chữ số của một số nguyên dương. Ví dụ: 12345 → 54321.
7. Hàm tìm tất cả các số từ 1 đến 100 sao cho tích các ước số nguyên tố của nó đúng bằng chính nó. Chẳng hạn: 14 = 2x7, 30 = 2x3x5, …
8. Viết hàm tính số PI theo công thức sau:

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \cdots\right)$$

9. Một máy đổi tiền chứa các đồng tiền mệnh giá 500, 1000, 2000, 5000, 10000, 20000. Hãy viết chương trình in ra tất cả các cách đổi một số tiền X sang các đồng tiền mệnh giá đã cho. Giả sử số lượng tờ tiền mỗi loại trong máy là không giới hạn.
10. Trò chơi đoán từ: Máy sẽ tạo ra một từ W gồm n chữ cái. Người chơi tiến hành đoán bằng cách nhập từ đoán G từ bàn phím. Chương trình sẽ kiểm tra từng ký tự trong G và so sánh với các ký tự trong W. Sau đó, chương trình xuất ra số ký tự đoán đúng và nằm cùng vị trí trong cả W, G kèm theo số ký tự có trong cả W và G nhưng nằm sai vị trí. Người chơi được phép đoán k lần. Nếu sau k lần đoán, người chơi vẫn chưa thể đoán ra từ W thì chương trình kết thúc và hiển thị thông báo người chơi thua. Ngược lại, thông báo người chơi thắng và bắt đầu màn chơi mới.