CHƯƠNG 2

LỚP VÀ ĐỐI TƯỢNG

GVHD: TS. GVC LÊ THỊ THÚY NGA

BỘ MÔN: ĐIỀU KHIỂN HỌC

NỘI DUNG

2.1 Lớp và đối tượng

2.2 Tạo và sử dụng đối tượng

2.3 Các kỹ thuật trong xây dựng lớp

2.4 Kết tập và kế thừa

2.5 Các kiểu dữ liệu tập hợp trong C#

2.6 Ngoại lệ và xử lý ngoại lệ

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

- Ngoại lệ trong C# là những tình huống mà chương trình không thể thực hiện được lệnh theo yêu cầu. Ví dụ: khi thực hiện phép chia, nếu mẫu số bằng 0 thì phép chia không thể thực hiện được, lúc này chương trình không biết phải thực hiện việc gì tiếp theo.
- Vì vậy C# đã cung cấp các công cụ đặc biệt để thông báo ngoại lệ, bắt và xử lý ngoại lệ.

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

❖Exception Handling (Xử lý ngoại lệ) trong C# được xây dựng dựa trên 4 từ khóa là: **try**, **catch**, **finally**, và **throw**:

- Câu lệnh **try** dùng để thực thi đoạn mã lệnh bắt ngoại lệ (lỗi) nếu có.
- Câu lệnh **catch** dùng để thực thi khi có ngoại lệ.
- Câu lệnh **finally** dùng để làm sạch tài nguyên sau khi có ngoại lệ xảy ra, luôn được thi hành dù có phát sinh ngoại lệ hay không.
- Câu lệnh **throw** dùng để ném đối tượng nếu đối tượng đó trực tiếp hoặc gián tiếp được kế thừa từ lớp **System.Exception** trong C#.

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

```
try
  {
    //khối lệnh cần thực thi để bắt ngoại lệ
  }
 catch (Exception type)
  {
  // khối lệnh cần thực thi khi có ngoại lệ
  }
finally
  {
    // khối lệnh luôn được thực thi dù có
   phát sinh ngoại lệ hay không.
  }
```

- Khối lệnh nào muốn giám sát để bắt ngoại lệ thì đưa vào khối **try**, nếu ngoại lệ xảy ra do lệnh trong khối đó thì sẽ bắt được - chương trình sẽ không kết thúc mà lập tức chuyển sang khối **catch**, tại đó bạn có ngay đối tượng lớp Exception - bạn cần xử lý theo logic ứng dụng của bạn điều hướng chương trình một cách thích hợp ở đây.
- Đối tượng lớp Exception có một số thuộc tính, tiện dụng cho bạn gỡ rối đó là:
  - **Message** chuỗi chứa nội dung thông báo lỗi
  - **StackTrace** chuỗi chứa các bước thực thi chương trình cho đến khi bị lỗi (có chứa các phương thức, hàm khi thực thi gây lỗi, vị trí file lỗi ...)
  - **Source** chứa tên ứng dụng hoặc đối tượng bị lỗi

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

Ví dụ 1: Truy xuất phần tử trong mảng số nguyên

```
try
{ // khối này được giám sát để bắt lỗi - khi nó phát sinh
    int[] a = new int[] {1,2,3};
   int i = a[10]; // dòng này phát sinh lỗi
   Console.WriteLine(i); // dòng này không được thực thi vì lỗi trên
}
```

```
catch (Exception b)
 { // khối này thực thi khi bắt được lỗi
    Console.WriteLine("Có lỗi rồi");
    Console.WriteLine(b.Message);
}
```

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

Khi có ngoại lệ chia một số cho 0, thì khối **catch** với ngoại lệ kiểu **DivideByZeroException** được thi hành

Bắt ngoại lệ khi chia cho số 0

Ví dụ 2: Phép chia 2 số nguyên

```
try
    {
        c = (float)(a/b);
    }
 catch (DivideByZeroException e)
    {
        Console.WriteLine("Bat Exception: {0}", e);
        Console.WriteLine(e.Message);
    }
 finally
     {
        Console.WriteLine("Ket qua: {0}",c);
     }
```

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

Ví dụ 3: Ném ngoại lệ bằng lệnh **throw**

```
public static double Thuong(double x, double y)
{
      if (y == 0)
       {
      Exception e = new Exception("Số chia không được bằng 0");
      throw e; // phát sinh ngoại lệ do số chia bằng 0.
       }
    return x / y;
}
static void Main(string[] args)
{
     double z = Thuong(100,0);
}
```

Nếu code của bạn muốn phát sinh ngoại lệ, khi muốn hệ thống biết có một lỗi nào đó cần xử lý, nếu không xử lý (bắt lại) thì chương trình kết thúc ở điểm phát sinh ngoại lệ. Để phát sinh ngoại lệ cần tạo ra một đối tượng lớp **Exception** hoặc các lớp kế thừa từ nó, sau đó phát sinh bằng lệnh **throw**

2.6 NGOẠI LỆ (EXCEPTION) VÀ XỬ LÝ NGOẠI LỆ TRONG C#

❖ **Các lớp ngoại lệ cơ bản đã được C# định nghĩa:**

Lớp ngoại lệ	Mô tả
System.IO.IOException	Xử lý lỗi I/O
System.IndexOutOfRangeException	Xử lý các lỗi phát sinh khi một phương pháp đề cập đến một chỉ số mảng nằm ngoài phạm vi.
System.ArrayTypeMismatchException	Xử lý các lỗi phát sinh khi loại chưa phù hợp với các kiểu mảng.
System.NullReferenceException	Xử lý các lỗi phát sinh từ một đối tượng null.
System.DivideByZeroException	Xử lý các lỗi phát sinh từ việc chia cho số không.
System.InvalidCastException	Xử lý các lỗi phát sinh trong ép kiểu.
System.OutOfMemoryException	Xử lý các lỗi được tạo ra từ bộ nhớ không đủ.
System.StackOverflowException	Xử lý các lỗi phát sinh từ tràn stack.

❖Các lớp ngoại lệ có thể được người dùng tự định nghĩa:

```
using System;
namespace UserDefinedException
{
   class TestTemperature
   {
      static void Main(string[] args)
      {
         Temperature temp = new Temperature(); //TAO DOI TUONG NHIET DO
         try
         {
            temp.showTemp();  //IN NHIET DO
         }
         catch(TempIsZeroException e) //NGOAI LE KHI NHIET DO BANG 0
         {
            Console.WriteLine("Ngoai le nhiet do bang 0: {0}", e.Message);
         }
         Console.ReadKey();
      }
   }
}
```

```
public class TempIsZeroException: Exception  //TAO LOP NGOAI LE BOI NGUOI DUNG
{
   public TempIsZeroException(string message): base(message)
   {
   }
}

public class Temperature
{
   int temperature = 0; //CHO NHIET DO BANG 0
   public void showTemp()  //IN NHIET DO
   {
      if(temperature == 0)
      {
         throw (new TempIsZeroException("Khong tim thay nhiet do")); //NEM NGOAI LE
      }
      else
      {
         Console.WriteLine("Nhiet do: {0}", temperature);
      }
   }
}
```