BÀI GIẢNG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

LỜI MỞ ĐẦU

Lập trình theo phương pháp hướng đối tượng xuất hiện từ những năm 1990 và được hầu hết các ngôn ngữ lập trình hiện nay hỗ trợ. Nó có nhiều ưu diểm vượt trội so với phương pháp lập trình cấu trúc cổ điển. Giáo trình này sẽ giới thiệu các đặc trưng của phương pháp lập trình hướng đối tượng như tính đóng gói, tính kế thừa và tính đa hình. Chúng tôi chọn ngôn ngữ **C#** để minh họa, vì đây là ngôn ngữ lập trình hướng đối tượng dễ học và phổ dụng trong hiện nay trong các lĩnh vực nghiên cứu cũng như trong ngành công nghệ phần mềm. Sau khi hoàn tất giáo trình này, sinh viên sẽ biết được cách mô hình hóa các lớp đối tượng trong thế giới thực thành các lớp đối tượng trong C# và cách phối hợp các đối tượng này để giải quyết vấn đề đang quan tâm.

Trước khi tìm hiểu chi tiết về phương pháp lập trình hướng đối tượng, sinh viên nên đọc trước phần ***Phụ lục A - Cơ bản về ngôn ngữ C#*** để làm quen với các kiểu dữ liệu, các cấu trúc điều khiển trong ngôn ngữ C#. Sau khi đã nắm bắt được phương pháp lập trình hướng đối tượng, sinh viên nên đọc thêm phần phụ lục ***B-Biệt lệ*** để có thể viết chương trình có tính dung thứ lỗi cao hơn.

MỤC LỤC

NỘI DUNG

I. Giới thiệu lập trình hướng đối tượng

I.1. Lập trình hướng thủ tục (Pascal, C, …)

Trong phương pháp lập trình thủ tục, chương trình là một hệ thống các thủ tục, hàm. Tức là, khi viết chương trình, ta phải xác định chương trình làm những công việc (thao tác) nào? Mỗi thao tác gồm những thao tác con nào? Từ đó mỗi thao tác sẽ tương ứng với một hàm. Như vậy, lập trình theo phương pháp thủ tục là xác định các hàm, định nghĩa các hàm và gọi các hàm này để giải quyết vấn đề được đặt ra.

Một trong những nhược điểm của phương pháp này là mọi hàm đều có thể truy cập biến toàn cục. Hoặc dữ liệu có thể phải truyền qua rất nhiều hàm trước khi đến được hàm thực sự sử dụng hoặc thao tác trên nó. Điều này dẫn đến sự khó kiểm soát khi chương trình quá lớn và khi phát triển, sửa đổi chương trình.

Một khó khăn nữa đó là việc kiểm soát và ghi nhớ thông tin các hàm khi số lượng hàm quá nhiều.

I.2. Lập trình hướng đối tượng (Object-oriented programming)

Phương pháp này lấy đối tượng làm nền tảng để xây dựng chương trình. Đối tượng là sự gắn kết giữa dữ liệu liên quan và các hàm (còn gọi là phương thức) thao tác trên các dữ liệu này.

Đối tượng = Dữ liệu + Phương thức

Khi viết chương trình theo phương pháp hướng đối tượng ta thường phải trả lời các câu hỏi:

- Chương trình (đoạn chương trình) liên quan tới những lớp đối tượng nào?
- Mỗi đối tượng cần có những dữ liệu và thao tác nào?
- Các đối tượng quan hệ với nhau như thế nào trong chương trình?

Từ đó ta thiết kế các lớp đối tượng và tổ chức trao đổi thông tin giữa các đối tượng, ra lệnh để đối tượng thực hiện các nhiệm vụ thích hợp.

Đôi lúc, ta có thể tiếp cận theo cách ngược lại: xác định các hàm (thao tác) cần thiết để xử lý các tác vụ nào đó trước, sau đó sắp xếp, tổ chức lại dữ liệu và các hàm thành các lớp đối tượng tương ứng.

Ví dụ:

- Đối tượng tài khoản ngân hàng:
    - Dữ liệu: số tài khoản, mật khẩu, tên chủ sở hữu, số tiền hiện có...

    - Thao tác: đăng nhập, gửi tiền, rút tiền, chuyển khoản…
- Đối tượng chuỗi :
    - Dữ liệu: mảng các kí tự.
    - Thao tác: tính chiều dài, nối hai chuỗi...
- Đối tượng file:
    - Dữ liệu: tên file, đường dẫn, nội dung, các thuộc tính ….
    - Thao tác: tạo, xóa, sao chép, di chuyển, đổi tên, mở file, sửa nội dung…

Các ngôn ngữ lập trình hướng đối tượng đều có ba đặc điểm chung là tính đóng gói (encapsulation), tính kế thừa (inheritance) và tính đa hình (polymorphism).

I.2.1. Tính đóng gói

Tính đóng gói thể hiện qua việc ràng buộc dữ liệu và phương thức thao tác trên dữ liệu đó vào trong lớp để dễ kiểm soát, làm tăng tính trừu tượng của dữ liệu. Lớp đối tượng chỉ cung cấp một số phương thức để giao tiếp với môi trường bên ngoài, che dấu đi cài đặt thực sự bên trong của lớp.

I.2.2. Tính kế thừa

Tính kế thừa là quá trình định nghĩa một lớp đối tượng (gọi là lớp dẫn xuất) dựa trên lớp khác đã định nghĩa gọi là lớp cơ sở nhằm tận dụng các đoạn mã chương trình đã có. Lớp mới chỉ việc bổ sung các thành phần riêng của chính nó hoặc định nghĩa lại các hàm của lớp cơ sở không còn phù hợp với nó.

I.2.3. Tính đa hình

Tính đa hình là ý *tưởng "sử dụng một giao diện chung cho nhiều phương thức khác nhau"*, dựa trên cơ chế liên kết muộn. Tức là phương thức cụ thể sẽ được xác định vào lúc chạy chương trình, tùy thuộc vào đối tượng đang thực thi giao diện đó. Điều này làm giảm đáng kể độ phức tạp của chương trình.

I.2.4. Ưu điểm của phương pháp lập trình hướng đối tượng

- Tính đóng gói làm giới hạn phạm vi sử dụng của các biến, nhờ đó việc quản lý giá trị của biến dễ dàng hơn, việc sử dụng mã an toàn hơn.
- Phương pháp này làm cho tốc độ phát triển các chương trình nhanh hơn vì mã được tái sử dụng và cải tiến dễ dàng, uyển chuyển. Việc phân module chức năng cũng dễ dàng, thuận tiên hơn.
- Phương pháp này tiến hành tiến trình phân tích, thiết kế chương trình thông qua việc xây dựng các đối tượng có sự tương hợp với các đối tượng thực tế. Điều này làm cho việc sửa đổi dễ dàng hơn khi cần thay đổi chương trình.
- …

II. Lớp và đối tượng

Trong OOP (Object Otiented Programming), chương trình là một hệ thống các lớp, đối tượng. Xây dựng một chương trình là định nghĩa các lớp đối tượng, sau đó tổ chức sắp xếp để các đối tượng phối hợp nhau thực thi nhiệm vụ.

II.1. Định nghĩa lớp

Một lớp là một kiểu cấu trúc mở rộng, đó là một kiểu mẫu chung cho các đối tượng thuộc cùng một loại. Như vậy, thành phần của lớp gồm cấu trúc dữ liệu mô tả các đối tượng trong lớp và các phương thức (còn gọi là hàm, hành vi, thao tác) mà mỗi biến đối tượng của lớp đều có. Các phương thức này thao tác trên các thành phần dữ liệu được khai báo trong lớp.

Việc định nghĩa lớp thể hiện tính đóng gói của phương pháp lập trình hướng đối tượng.

Cú pháp định nghĩa lớp:

[MứcĐộTruyCập] class TênLớp [:LớpCơSở]

{

- Khai báo các thành phần dữ liệu (khai báo biến)

- Định nghĩa các phương thức, thuộc tính của lớp

}

Chú ý:

- *Dữ liệu và phương thức của lớp được gọi chung là thành phần của lớp.*
- *Các thành phần dữ liệu được xem như biến toàn cục đối với các phương thức của lớp, tức là các phương thức của lớp có quyền truy cập đến các thành phần dữ liệu này mà không cần phải khai báo lại trong từng phương thức.*

Mức độ truy cập

Thông thường, mức độ truy cập (***access-modifiers***) của một lớp là ***public***. Ngoài ra ta có thể quy định mức độ truy cập riêng cho từng thành phần của lớp. Mức độ truy cập của một thành phần cho biết loại phương thức nào được phép truy cập đến nó, hay nói cách khác nó mô tả phạm vi mà thành phần đó được nhìn thấy.

Bảng sau liệt kê các kiểu mức độ truy cập của các thành phần trong một lớp:

Mức độ truy cập	**Ý nghĩa**
public	Thành viên được đánh dấu public được nhìn thấy bởi bất kỳ phương thức nào của lớp khác.
private	Chỉ có các phương thức của lớp A mới được phép truy cập đến thành phần được đánh dấu private trong các lớp A.
protected	Chỉ có các phương thức của lớp A hoặc của lớp dẫn xuất từ A mới được phép truy cập đến thành phần được đánh dấu protected trong lớp A.
internal	Các thành viên internal trong lớp A được truy xuất trong các phương thức của bất kỳ lớp trong khối kết hợp (assembly, name space) của A
protected internal	Tương đương với protected or internal

Chú ý:

- *Mặc định, khi không chỉ cụ thể mức độ truy cập thì thành viên của lớp được xem là có mức độ truy cập private.*

- *Mức độ truy cập internal cho phép các phương thức của các lớp trong cùng một khối kết hợp (assembly) với lớp đang định nghĩa có thể truy cập. Các lớp thuộc cùng một project có thể xem là cùng một khối kết hợp.*

II.2. Tạo đối tượng

Lớp mô tả cấu trúc chung của một nhóm đối tượng nào đó, ngược lại, một đối tượng là một trường hợp cụ thể của một lớp (còn gọi là một thể hiện của một lớp).

Vì đối tượng là một kiểu tham chiếu nên dữ liệu thực sự được tạo trên vùng nhớ Heap và ta phải dùng toán tử ***new*** để cấp phát cho đối tượng. Kể từ lúc đối tượng được cấp phát bộ nhớ, ta có thể gán các giá trị cho các biến thành viên, gọi thi hành các phương thức của đối tượng này.

Thường thì ta chỉ việc khai báo và cấp phát đối tượng, việc hủy vùng nhớ mà đối tượng chiếm giữ khi đối tượng đó mất hiệu lực sẽ do bộ dọn rác của trình biên dịch đảm nhiệm.

Cú pháp khai báo đối tượng và cấp phát vùng nhớ cho đối tượng:

TênLớp TênBiếnĐốiTượng;

TênBiếnĐốiTượng = new TênLớp(DanhSáchĐốiSố);

hoặc

TênLớp TênBiếnĐốiTượng = new TênLớp(DanhSáchĐốiSố);

Chú ý:

- *Sau khi khai báo biến đối tượng thì biến đó chỉ là một con trỏ.*
- *Sau khi cấp phát bằng từ khóa new thì biến trỏ tới một đối tượng thực sự.*

Ví dụ:

Chương trình nhập chiều dài, chiều rộng của hình chữ nhật và xuất ra diện tích, chu vi của hình chữ nhật.

```
using System;
namespace LopDoiTuongHCN  {
     class HCN      {
          protected float Dai, Rong;
          public float ChuVi() {
               return (Dai + Rong )*2;
          }
          public float DienTich()  {
               return Dai* Rong;
          }
          public void Nhap()  {
               Console.WriteLine("Nhap chieu dai: ");
               Dai = float.Parse(Console.ReadLine());
```

```
            Console.WriteLine("Nhap chieu rong: ");
            Rong = float.Parse(Console.ReadLine());
        }
        public void Xuat()  {
            Console.WriteLine("Hinh chu nhat: Dai =
            {0}, Rong = {1}", Dai, Rong);
        }
    }

    class Application   {
        static void Main(string[] args)         {
            HCN h;
            h = new HCN();
            h.Nhap();
            h.Xuat();
            Console.WriteLine("Chu vi hinh chu nhat:
            {0}", h.ChuVi());
            Console.WriteLine("Dien tich hinh chu nhat:
            {0}", h.DienTich());
            Console.ReadLine();
        }
    }
}
```

Trong ví dụ trên, ta định nghĩa một lớp các hình chữ nhật (HCN), mỗi đối tượng thuộc lớp này có thành phần dữ liệu là chiều dài và chiều rộng và có các phương thức như: *nhap(), xuat(), DienTich(), ChuVi()*. Sau đó, trong hàm *Main()* ta khai báo một đối tượng hình chữ nhật tên là *h*, cấp phát vùng nhớ cho đối tượng này và gọi thực hiện các phương thức của nó.

Chú ý:

Nếu ta bỏ đi từ khóa *public* đứng trước mỗi phương thức của lớp HCN thì hàm *Main()* sẽ không thể truy cập đến các phương thức của đối tượng *h* và trình biên địch sẽ báo lỗi vì khi đó các phương thức này có mức độ truy cập là *private*.

Bài tập 1: *xây dựng lớp hình chữ nhật với thành phần dữ liệu là tọa độ góc trên bên trái (x1, y1), tọa độ góc dưới bên phải (x2, y2) và các phương thức tính chiều dài, chiều rộng, diện tích, chu vi của hình chữ nhật và phương thức vẽ hình chữ nhật bằng các ký tự '*' ra màn hình.*

Bài tập 2: *viết chương trình xây dựng lớp phân số và các thao tác trên phân số như +, -, *, /, tìm ước số chung lớn nhất của tử và mẫu, rút gọn, cộng phân số với một số nguyên.*

Gợi ý:

```
class PhanSo
{
    int Tu, Mau;   // private members
    public void NhapPhanSo()
```

```
        {
            // Đoạn mã nhập tử số và mẫu số.
        }
        public void GanGiaTri(int TuSo, int MauSo)
        {
            // Đoạn mã gán giá trị cho tử số và mẫu số.
        }
        public void XuatPhanSo()
        {
            // Đoạn mã xuất tử số và mẫu số ở dạng
        (a/b)

        }
        public PhanSo Cong(PhanSo PS2)
//cộng phân số hiện hành với phân số PS2 và trả về một
phân số
        {
            PhanSo KetQua = new PhanSo();
            KetQua.Tu = Tu * PS2.Mau + Mau* PS2.Tu;
            KetQua.Mau = Mau * PS2.Mau;
            return KetQua;
        }
        public PhanSo Tru(PhanSo PS2)
        {
        // Đoạn mã tru phân số hiện hành với phân số
        PS2 và trả về một phân số

        }
```

… các phương thức khác

II.3. Phương thức tạo lập (constructor) của một đối tượng

Phương thức tạo lập của một đối tượng có các tính chất sau:

- Được gọi đến một cách tự động khi một đối tượng của lớp được tạo ra. Dùng để khởi động các giá trị đầu cho các thành phần dữ liệu của đối tượng thuộc lớp.
- Tên phương thức giống với tên lớp và có mức độ truy cập là public.
- Không có giá trị trả về.
- Trước khi phương thức tạo lập chạy, đối tượng chưa thực sự tồn tại trong bộ nhớ, sau khi tạo lập hoàn thành, bộ nhớ lưu trữ một thể hiện hợp lệ của lớp.
- Khi ta không định nghĩa một phương thức tạo lập nào cho lớp, trình biên dịch sẽ tự động tạo một phương thức tạo lập mặc định cho lớp đó và khởi tạo các biến bằng các giá trị mặc định.

Thông thường ta nên định nghĩa một phương thức tạo lập cho lớp và cung cấp tham số cho phương thức tạo lập để khởi tạo các biến cho đối tượng của lớp.

Chú ý rằng, nếu lớp có phương thức tạo lập có tham số thì khi khởi tạo đối tượng (bằng toán tử new) ta phải truyền tham số cho phương thức tạo lập theo cú pháp:

TênBiếnĐốiTượng = new TênLớp(DanhSáchĐốiSố);

Ví dụ:

Ví dụ sau xây dựng một lớp ***Time*** trong đó có một phương thức tạo lập nhận tham số có kiểu ***DateTime*** (kiểu xây dựng sẵn của trình biên dịch) làm tham số khởi gán cho các thành phần dữ liệu của đối tượng thuộc lớp ***Time***.

```
using System;
public class Time
{
      // private member variables
      int Year;
      int Month;
      int Date;
      int Hour;
      int Minute;
      int Second = 30;

      public void DisplayCurrentTime( )
      {
            Console.WriteLine("Current time is: {0}/{1}/{2}
            {3}:{4}:{5}",Month, Date, Year, Hour, Minute,
            Second);
      }

      public Time(System.DateTime dt)// constructor
      {
            Console.WriteLine("Ham constructor tu dong duoc
goi!");
            Year = dt.Year;
            Month = dt.Month;
            Date = dt.Day;
            Hour = dt.Hour;
            Minute = dt.Minute;
      }
}

class DateTimeConstrcutorApp
{
      static void Main( )
      {
            DateTime currentTime = DateTime.Now;
            Time t = new Time(currentTime);
            t.DisplayCurrentTime( );
            Console.ReadLine();
      }
```

}

Kết quả của chương trình:

Hãy nhấn **F11** chạy debug để hiểu rõ hơn quá trình khởi tạo đối tượng *t,* gọi thực hiện hàm constructor của *t*.

Chú ý rằng, ta cố tình gán giá trị mặc định là 30 cho biến *Second* để biến *Second* của mọi đối tượng thuộc lớp *Time* khi mới được tạo ra đều có giá trị là 30.

II.4. Phương thức tạo lập sao chép (copy constructor)

Phương thức tạo lập sao chép khởi gán giá trị cho đối tượng mới bằng cách sao chép dữ liệu của đối tượng đã tồn tại (cùng kiểu). Ví dụ, ta muốn truyền một đối tượng *Time t1* để khởi gán cho đối tượng *Time t2* mới với mục đích làm cho *t2* có giá trị giống t1, ta sẽ xây dựng phương thức tạo lập sao chép của lớp *Time* như sau:

```
public Time(Time existingTimeObject) {
     Year      = existingTimeObject.Year;
     Month     = existingTimeObject.Month;
     Date      = existingTimeObject.Date;
     Hour      = existingTimeObject.Hour;
     Minute    = existingTimeObject.Minute;
     Second    = existingTimeObject.Second;
}
```

Khi đó cú pháp khai báo *t2* là:

Time t2 = new Time(t1).

Khi đó hàm copy constructor được gọi và gán giá trị của *t1* cho *t2*.

Bài tập 1: *Xây dựng lớp HocSinh (họ tên, điểm toán, điểm văn) với các phương thức: khởi tạo, xuất, tính điểm trung bình.*

Bài tập 2: *Xây dựng lại lớp PhanSo phần trước với phương thức khởi tạo gồm 2 tham số.*

Bài tập 3: *Xây dựng lớp ngăn xếp Stack lưu trữ dữ liệu số nguyên bằng mảng với các thao tác cơ bản như: Push, Pop, kiểm tra tràn stack, kiểm tra stack rỗng…Dữ liệu của một đối tượng thuộc lớp Stack gồm: Data (mảng số nguyên), Size (kích thước của mảng Data), Top (chỉ số của phần tử nằm trên đỉnh Stack).*

Bài tập 4: *Xây dựng lớp hàng đợi Queue lưu trữ dữ liệu số nguyên bằng mảng với các thao tác trên hàng đợi.*