|  |
| --- |
| **TRƯỜNG CAO ĐẲNG VIỄN ĐÔNG**          **LẬP TRÌNH ỨNG DỤNG C#**      ***Đề tài*: XÂY DỰNG CHƯƠNG TRÌNH QUẢN LÝ TRƯỜNG HỌC**      **Nhóm thực hiện: Nhóm 3**  **Học sinh thực hiện: Đỗ Minh Quý**  **Phan Trương Đăng Khoa**  **Trần Văn Thịnh**  **Lớp: 14THC**  **Giáo viên: Nguyễn Hữu Lợi**  **Học Kì: Học Kì I** |

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## I. Ngôn ngữ lập trình C#

**1. Giới thiệu**

Ngôn ngữ C# là một ngôn ngữ được dẫn xuất từ C và C++, nhưng nó được tạo từ nền tảng phát triển hơn. Microsoft bắt đầu với công việc trong C và C++ và thêm vào những đặc tính mới để làm cho ngôn ngữ này dễ sử dụng hơn. Nhiều trong số những đặc tính này khá giống với những đặc tính có trong ngôn ngữ Java. Không dừng lại ở đó, Microsoft đưa ra một số mục đích khi xây dựng ngôn ngữ này. Những mục đích này được được tóm tắt như sau:

* C# là ngôn ngữ đơn giản
* C# là ngôn ngữ hiện đại
* C# là ngôn ngữ hướng đối tượng
* C# là ngôn ngữ mạnh mẽ và mềm dẻo
* C# là ngôn ngữ có ít từ khóa
* C# là ngôn ngữ hướng module
* C# sẽ trở nên phổ biến

## 2. Các kiểu dữ liệu

Tương tự như C++ hay Java, C# chia thành hai tập hợp kiểu dữ liệu chính: Kiểu xây dựng sẵn (built- in) mà ngôn ngữ cung cấp cho người lập trình và kiểu được người dùng định nghĩa (user-defined) do người lập trình tạo ra

C# phân tập hợp kiểu dữ liệu này thành hai loại: Kiểu dữ liệu giá trị (value) và kiểu dữ liệu tham chiếu (reference). Việc phân chi này do sự khác nhau khi lưu kiểu dữ liệu giá trị và kiểu dữ liệu tham chiếu trong bộ nhớ. Đối với một kiểu dữ liệu giá trị thì sẽ được lưu giữ kích thước thật trong bộ nhớ đã cấp phát là stack. Trong khi đó kiểu dữ liệu tham chiếu như các đối tượng được cấp phát trên heap. Khi một đối tượng được cấp phát trên heap thì địa chỉ của nó được trả về, và địa chỉ này được gắn đến một tham chiếu.

## Bảng 1-1 Kiểu dữ liệu xây dựng sẵn

|  |  |  |  |
| --- | --- | --- | --- |
| **Kiểu C#** | **Số byte** | **Kiểu .NET** | **Mô tả** |
| byte | 1 | Byte | Số nguyên dương không dấu từ 0-255 |
| char | 2 | Char | Kí tự Unicode |
| bool | 1 | Boolean | Giá trị logic true/ false |
| sbyte | 1 | Sbyte | Số nguyên có dấu ( từ -128 đến 127) |
| short | 2 | Int16 | Số nguyên có dấu giá trị từ -32768 đến 32767 |
| ushort | 2 | Int16 | Số nguyên không dấu 0 – 65.535 |
| int | 4 | Int32 | Số nguyên có dấu –2.147.483.647 và  2.147.483.647 |
| uint | 4 | Uint32 | Số nguyên không dấu 0 – 4.294.967.295 |
| float | 4 | Single | Kiểu dấu chấm động, giá trị xấp xỉ từ 3,4E- 38 đến 3,4E+38, với 7 chữ số có nghĩa.. |
| double | 8 | Double | Kiểu dấu chấm động có độ chính xác gấp đôi, giá trị xấp xỉ từ 1,7E-308 đến 1,7E+308, với 15,16 chữ số có nghĩa |
| decimal | 8 | Decimal | Có độ chính xác đến 28 con số và giá trị thập phân, được dùng trong tính toán tài chính, kiểu này đòi hỏi phải có hậu tố “m” hay “M” theo sau giá trị. |
| long | 8 | Int64 | Kiểu số nguyên có dấu có giá trị trong khoảng :  -9.223.370.036.854.775.808 đến  9.223.372.036.854.775.807 |
| ulong | 8 | Uint64 | Số nguyên không dấu từ 0 đến  0xffffffffffffffff |

## Chọn một kiểu định sẵn

Tuỳ vào từng giá trị muốn lưu trữ mà ta chọn kiểu cho phù hợp. Nếu chọn kiểu quá lớn so với các giá trị cần lưu sẽ làm cho chương trình đòi hỏi nhiều bộ nhớ và chạy chậm. Trong khi nếu giá trị cần lưu lớn hơn kiểu thực lưu sẽ làm cho giá trị các biến bị sai và chương trình cho kết quả sai.

Kiểu char biểu diễn một ký tự Unicode. Ví dụ “\u0041” là ký tự “A” trên bảng Unicode.

Một số ký tự đặc biệt được biểu diễn bằng dấu “\” trước một ký tự khác

## Kiểu dữ liệu do người dùng định nghĩa

- Tất cả kiểu dữ liệu do người dùng định nghĩa ngoài trừ kiểu cấu trúc điều là kiểu dữ liệu tham chiếu

Một sô kiểu dữ liệu do người dùng định nghĩa gồm

* **object**: đây là kiểu dữ liệu cơ sở chứa tất cả các kiểu dữ liệu khác trong C#.
* **string**: kiểu dữ liệu chuỗi ký tự.
* **class**: kiểu dữ liệu class.
* **delegate**: kiểu dữ liệu chuyển giao.
* **interface**: kiểu dữ liệu giáo tiếp.  **array**: kiểu dữ liệu mảng.

## 3. Các toán tử

Toán tử được kí hiệu bằng một biểu tượng dùng để thực hiện một hành động. Các kiểu dữ liệu cơ bản của C# như kiểu nguyên hỗ trợ rất nhiều các toán tử như toán tử gán, toán tử toán học, logic....

Các phép toán +, -, \*, / là một ví dụ về toán tử. Áp dụng các toán tử này lên các biến kiểu số ta có kết quả như việc thực hiện các phép toán thông thường.

int a = 10; int b = 20;

int c = a + b; // c = 10 + 20 = 30

C# cung cấp cấp nhiều loại toán tử khác nhau để thao tác trên các kiểu biến dữ liệu, được liệt kê trong bảng sau theo từng nhóm ngữ nghĩa.

## Bảng 1.2 Các nhóm toán tử trong C#

|  |  |  |
| --- | --- | --- |
| **Nhóm toán tử** | **Toán tử** | **Ý nghĩa** |
| Toán học | + - \* / % | Cộng, trừ, nhân, chia, chia lấy phần dư |
| Logic | & | ^ ! ~ && true false | Phép toàn logic và thao tác trên bit |
| Ghép chuỗi | + | Ghép nối hai chuỗi |
| Tăng, giảm | ++, -- | Tăng/giảm toán hạng lên/xuống 1.  Đứng trước hoặc sau toán hạng |
| Dịch bit | << >> | Dịch trái, dịch phải |
| Quan hệ | == != < > <= >= | Bằng, khác, nhỏ hơn, lớn hơn, nhỏ hơn hoặc bằng, lớn hơn hoạc bằng |
| Gắn | = += -= \*= /= %= &= != ^ =  <<= >>= | Phép gắn |
| Chỉ số | [] | Cách truy xuất phần tử của mảng |
| Ép kiểu | () |  |
| Indirecti0on và  Address | \* -> [] & | Dùng cho con trỏ |

### 3.1 Toán tử gán (=)

Toán tử gán (=) Toán tử này cho phép thay đổi các giá trị của biến bên phải toán tử bằng giá trị bên trái toán tử.

### 3.2 Nhóm toán tử toán học

C# dùng các toàn tử số học với ý nghĩa theo đúng tên của chúng như: + (cộng), – (trừ) \* (nhân) và / (chia). Tùy theo kiểu của hai toán hạng mà toán tử trả về kiểu tương ứng. Ngoài ra, còn có toán tử % (lấy phần dư) được sử dụng trong các kiểu số nguyên.

### 3.3 Các toán tử tăng và giảm

C# cũng kế thừa từ C++ và Java các toán tử: +=,-=, \*=, /= , %= nhằm làm đơn giản hoá. Nó còn kế thừa các toán tử tiền tố và hậu tố (như biến++, hay ++biến) để giảm bớt sự cồng kềnh trong các toán tử cổ điển.

### 3.4 Các toán tử quan hệ

Các toán tử quan hệ được dùng để so sánh hai giá trị với nhau và kết quả trả về có kiểu Boolean. Toán tử quan hệ gồm có: == (so sánh bằng), != (so sánh khác), > (so sánh lớn hơn), >= (lớn hơn hay bằng), < (so sánh nhỏ hơn), <= (nhỏ hơn hay bằng).

### 3.5 Các toán tử logic

Các toán tử logic gồm có: && (và), || (hoặc), ! (phủ định). Các toán tử này được dùng trong các biểu thức điều kiện để kết hợp các toán tử quan hệ theo một ý nghĩa nhất định.

### 3.6 Thứ tự các toán tử

Đối với các biểu thức toán, thứ tự ưu tiên là thứ tự được qui định trong toán học. Còn thứ tự ưu tiên thực hiện của các nhóm toán tử được liệt kê theo bảng dưới đây

## Bảng 1-3 Thứ tự ưu tiên của các nhóm toán tử (chiều ưu tiên từ trên xuống)

|  |  |  |
| --- | --- | --- |
| **Nhóm toán tử** | **Toán tử** | **Ý nghĩa** |
| Primary(chính) | {x} x.y f(x) a[x] x++ x-- |  |
| Unary | + - ! ~ ++x –x (T)x |  |
| Nhân | \* / % | Nhân, chia, chia lấy phần dư |
| Cộng | + - | công, trừ |
| Dịch bit | << >> | Dịch trái dịch phải |
| Bằng | == != | Bằng, khác |
| Logic trên bit AND | & | Và trên bit |
| XOR | ^ | Xor trên bit |
| OR | | | Hoặc trên bit |
| Điều kiện AND | && | Và trên biểu thức điều kiện |
| Điều kiện OR | || | Hoặc trên biểu thức điều kiện |
| Điều kiện | ?: | Điều kiện tượng tự if |
| Asignment | = \*= /= %= += -= <<=  =>> &= ^= |= |  |

**3.7 Toán tử tam phân** Cú pháp:

<biểu thức điều kiện>? <biểu thức 1>: <biểu thức 2>;

Ý nghĩa:

* Nếu biểu thức điều kiện đúng thì thực hiện biểu thức 1.
* Nếu sai thì thực hiện biểu thức 2.

## 4. Cấu trúc rẽ nhánh

**a. Câu lệnh if… else …**

Cú pháp:

if ( biểu thức logic ) khối lệnh;

hoặc

if ( biểu thức logic )

|  |  |
| --- | --- |
| else | khối lệnh 1; |
|  | khối lệnh 2; |

*Ghi chú: Khối lệnh là một tập các câu lện trong cặp dấu “{…}”. Bất kỳ**nơi đâu có câu lệnh thì ở đó có thể viết bằng một khối lệnh.*

Biểu thức logic là biểu thức cho giá trị dúng hoặc sai (***true*** hoặc ***false***). Nếu “*biểu thức logic*” cho giá trị đúng thì “*khối lệnh*” hay “*khối lệnh 1*” sẽ được thực thi,ngược lạ**i** “*khối lệnh 2*” sẽ thực thi. Một điểm khác biệt với C++ là biểu thức trongcâu lệnh if phải là biểu thức logic, không thể là biểu thức số. **b. Câu lệnh switch case** Cú pháp:

switch ( biểu\_thức\_lựa\_chọn )

{ case biểu\_thức\_hằng :

khối lệnh; lệnh nhảy; [ default : khối lệnh; lệnh nhảy; ]

}

Biểu thức lựa chọn là biểu thức sinh ra trị nguyên hay chuỗi. Switch sẽ so sánh biểu\_thức\_lựa\_chọn với các biểu\_thức\_hằng để biết phải thực hiện với khối lệnh nào. Lệnh nhảy như break, goto…để thoát khỏi câu switch và bắt buộc phải có.

int nQuyen = 0;

switch ( sQuyenTruyCap )

{ case “Administrator”:

nQuyen = 1; break; case “Admin”:

goto case “Administrator”; default: nQuyen = 2; break;

}

**c. Câu lệnh for** Cú pháp:

for ( [khởi\_tạo\_biến\_đếm]; [biểu\_thức]; [gia\_tăng\_biến\_đếm] )

khối lệnh;

Ví dụ 3-4 Tính tổng các số nguyên từ a đến b

int a = 10; int b = 100; int nTong = 0; **d. Câu lệnh while** Cú pháp:

while ( biểu\_thức\_logic ) khối\_lệnh;

Khối\_lệnh sẽ được thực hiện cho đến khi nào biểu thức còn đúng. Nếu ngay từ đầu biểu thức sai, khối lệnh sẽ không được thực thi.

**e. Câu lệnh do …while** Cú pháp:

do khối\_lệnh while ( biếu\_thức\_logic )

Khác với while khối lệnh sẽ được thực hiện trước, sau đó biệu thức được kiểm tra. Nếu biểu thức đúng khối lệnh lại được thực hiện.

**II. Lập trình hướng đối tượng với C#**

## 1. Lớp và đối tượng

### 1.1 Đối tượng

Trong lập trinh hướng đối tượng, tất cả thực thể trong hệ thống đều được coi là các đối tượng cụ thể. Đối tượng là một thực thể hoạt động khi chạy chượng trình.

Một đối tượng là một thực thể đang tồn tại trong hệ thống và được xác định bằng ba yếu tố:

* **Định danh đối tượng:** xác định duy nhất cho mỗi đối tượng trong hệ thống, nhằm phân biết các đối tượng với nhau.
* **Trạng thái của đối tượng:** là sự tổ hợp của các giá trị của các thuộc tính mà đối tượng đang có.
* **Hoạt động của đối tượng:** là các hành động mà đối tượng có khả năng thực hiện được.

Trạng thái hiện tại của đối tượng quy định tính chất đặc trưng của đối tượng. Ví dụ, đối tượng xe có trạng thái là:

* Nhãn hiệu xe là Ford
* Màu xe là màu trắng
* Giá bán xe là 5000$

Mỗi đối tượng sẽ có một số hành động gọi là phương thức. Ví dụ xe có phương thức :

* Khởi động
* Dừng lại
* Chạy

### 1.2 Class (lớp)

**Định nghĩa lớp:** Trong lập trình hướng đối tượng, đối tượng là một thực thể cụ thể, tồn tại trong hệ thống. Trong khi đó, lớp là một khái niệm trừu tượng, dùng để chỉ một tập hợp các đối tượng cùng loại.

**Ví dụ:** Trong bài toán quản lí xe hơi của một của hàng kinh doanh xe, mỗi chiếc xe có mặt trong của hàng được coi là một đối tượng. Khi đó ta có khá niệm lớp “Xe hơi” chỉ tất cả các loại xe hơi đang có

## Khai báo một lớp

Định nghĩa một lớp mới với cú pháp như sau:

[attribute][bổ từ truy xuất] class định danh [:lớp cơ sở]

{ thân lớp

}

**Ví dụ** Khai báo một lớp

public class XeHoi

{

private string tenxe;

private string nhanhieu;

private string mauxe;

public string TenXe

{

get { retrurn tenxe;} set { tenxe= value;}

}

public string NhanHieu

{

get { retrurn nhanhieu;} set { nhanhieu = value;}

}

public string MauXe

{

get { retrurn mauxe;} set { mauxe = value;}

}

}

## Lớp và đối tượng

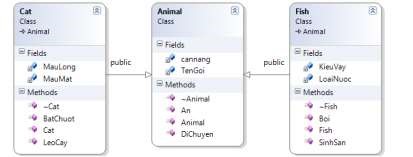
Lớp và đối tượng mặc dù có mối tượng quan, nhưng bản chất lại khác nhau:

* Lớp là sự trừu tượng hóa các đối tượng. Trong khi đó, đối tượng là thể hiện cụ thể của một lớp.
* Đối tượng là một thực thể cụ thể, có thực, tồn tại trong hệ thống. Trong khi đó lớp là một khai niệm trừu tượng, chỉ tồn tại ở dạng khái niệm để mô tả các đặc tính chung của một nhóm đối tượng.
* Tất cả các đối tượng thuộc cùng một lớp giống nhau về thuộc tính và phương thức.

### 2. Kế thừa lớp

Một tính năng then chốt của lập trình hướng đối tượng đó là tính kế thừa. Nhờ vào tính kế thừa, nó cho phép một lớp có thể dẫn xuất từ một lớp khác hay nói các khác một lớp có thể thừa có thể sử dụng lại các thuộc tính và phương thức của lớp bị kế thừa, chính vì thế chúng sẽ tự động tiếp nhận các thành viên của bố mẹ và bổ sung thêm các thành viên của riêng chúng. Tính kế thừa cho phép lớp mới có thể nhận được mọi dữ liệu thành viên (private, protected, public) và các hàm thành viên (trừ hàm tạo, hàm hủy, hàm bạn và hàm toán tử gán =)**.**

**Ví dụ:** Lớp động vật Animal có các thuộc tính thành viên: tên gọi, cân nặng. Các hàm thành viên: di chuyển, ăn. Ta xét hai lớp dẫn xuất của nó là lớp mèo Cat và lớp cá Fish. Lớp Cat có các thuộc tính thành viên riêng: màu lông, màu mắt. Các hàm thành viên riêng: Bắt chuột, Leo cây. Lớp Fish có các thuộc tính thành viên riêng: kiểu vẩy, loại nước (nước ngọt, nước mặn, nước lợ). Các hàm thành viên : bơi, sinh sản (cách thức sinh con như thế nào).



## Khai báo một lớp kế thừa

Trong C#, khi ta tạo một lớp kế thừa bằng cách công một thêm dấu “:” và sau tên của lớp kế thừa và theo sau đó là lớp cơ sở như sau:

public class <Tên lớp dẫn suất>: <Tên lớp cơ sở>

Ví dụ class Animal

{

private string cannang; private string tengoi; public void An

{

….

}

public void DiChuyen

{

….

}

} class Cat:Aminal

{

private string maulong; private string maumat; pulic string MauLong

{

…

}

pulic string MauMat

{

…

}

}

### 3. Giao diện (interface)

Giao diện giống như một khuôn mẫu, các class sử dụng giao diện đều phải thực hiện tất cả các phương thức của giao diện, điều này giúp đồng nhất về phương thức giúp các lớp khác nhau có thể làm việc được với nhau.

Cú pháp của việc định nghĩa một giao diện:

[attributes] [access-modifier] interface interface-name [:baselist] { interface-body

}

Ý nghĩa của từng thành phần như sau

* **attributes**: sẽ đề cập ở phần sau.
* **modifiers**: bổ từ phạm vi truy xuất của giao diện
* **identifier**: tên giao diện muốn tạo
* **base-list**: danh sách các giao diện mà giao diện này thừa kế,
* (nói rõ trong phần thừa kế)
* **interface-body**: thân giao diện luôn nằm giữa cặp dấu {}

***Lưu ý****: Tên giao diện thường bắt đầu bằng chữ I (in hoa) Ví dụ:*

interface IStorable

{ void Read( ); void Write(object);

}

Sử dụng giao diện vừa tạo ra

// lớp Document thừa kế IStorable,

// phải cài đặt tất cả các phương thức của IStorable public class Document : Istorable

{ public void Read( ) { // phải cài đặt...} public void Write(object obj) { // phải cài đặt...}

// ...

}

## Cài đặt nhiều giao diện

Lớp có thể cài đặt một hoặc nhiều giao diện. Chẳng hạn như ở lớp Document ngoài lưu trữ ra nó còn có thể được nén lại.

Ta cho lớp Document cài đặt thêm một giao diện thứ hai là Icompressible,Document phải cài đặt tất cả phương thức của Icompressible và IStorable: public class Document : IStorable, Icompressible

{

public void Comress( )

{

Console.WriteLine("Implementing the Compress Method");

} public void Decompress( )

{

Console.WriteLine("Implementing the Decompress Method");

}

}

## Mở rộng giao diện

Chúng ta có thể mở rộng (thừa kế) một giao diện đã tồn tại bằng cách thêm vào đó những phương thức hoặc thành viên mới. Chẳng hạn như ta có thể mở rộng ICompressable thành ILoggedCompressable với phương thức theo dõi những byte đã được lưu:

interface ILoggedCompressible : ICompressible

{ void LogSavedBytes( );

}

Lớp cài đặt phải cân nhắc chọn lựa giữa 2 lớp ICompressable hay ILoggedCompressable, điều này phụ thuộc vào nhu cầu của lớp đó. Nếu một lớp có sử dụng giao diện ILoggedCompressable thì nó phải thực hiện toàn bộ các phương thức của ILoggedCompressable (bao gồm ICompressable và phương thức mở rộng).

## Kết hợp các giao diện khác nhau

Tương tự, chúng ta có thể tạo một giao diện mới bằng việc kết hợp nhiều giao diện và ta có thể tùy chọn việc có thêm những phương thức hoặc những thuộc tính mới.

Ví dụ như ta tạo ra giao diện IStorableCompressable từ giao diện IStorable và ILoggedCompressable và thêm vào một phương thức mới dùng để lưu trữ kích thước tập tin trước khi nén.

interface IStorableCompressible: IStoreable,ILoggedCompressible

{ void LogOriginalSize( );

}

### 4. Không gian tên (Namespace)

Namespace trong ngôn ngữ C#, nhằm tránh sự xung đột giữa việc sử dụng các thư viện khác nhau từ các nhà cung cấp. Ngoài ra, namespace được xem như là tập hợp các lớp đối tượng, và cung cấp duy nhất các định dan cho các kiểu dữ liệu và được đặt trong một cấu trúc phân cấp. Việc sử dụng namespace trong khi lập trình là một thói quen tốt, bởi vì công việc này chính là cách lưu các mã nguồn để sử dụng về sau. Ngoài thư viện namespace do MS.NET và các hãng thứ ba cung cấp, ta có thể tạo riêng cho mình các namespace. C# đưa ra từ khóa using đề khai báo sử dụng namespace trong chương trình:

*using < Tên namespace >*

Để tạo một namespace dùng cú pháp sau:

*namespace <Tên namespace>*

*{*

*< Định nghĩa lớp A> < Định nghĩa lớp B > .....*

*}*

Ví dụ

namespace MyLib

{ using System; public class Tester

{ public static int Main()

{ for (int i =0; i < 10; i++)

{

Console.WriteLine( “i: {0}”, i);

} return 0;

}

}

}

Ví dụ trên tạo ra một namespace có tên là MyLib, bên trong namespace này chứa một lớp có tên là Tester. C# còn cho phép trong một namespace có thể tạo một namespace khác lồng bên trong và không giới hạn mức độ phân cấp này.

**CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG**

## I. Phân tích chức năng

Ứng dụng quản lí trường học có thể giúp công việc quản lí sinh viên, giảng viên trở lên dễ dàng, nhanh chóng, độ chính xác cao.

Chức năng ứng dụng:

* Quản lí tài khoản: tạo, thoát, sửa, xoá.
* Quản lí giảng viên: tạo, thoát, sửa, xoá.
* Quản lí sinh viên: tạo, sửa, xóa, thoát.

- Quản lí lớp: thêm mới, sửa, xóa, thoát.

* Quản lí kết quả: tạo, thoát, sửa xoá.

## II. Cơ sở dữ liệu

## 1. Các bảng

### 1.1. Bảng Giáo Viên

|  |  |  |
| --- | --- | --- |
|  | **Giáo Viên** |  |
| **Mã** | **Kiểu dữ liệu** | **Ghi chú** |
| MaGV | Varchar(10) | Khóa chính  Bắt buộc nhập |
| TenGV | Nvarchar(50) | Bắt buộc nhập |
| NgaySinh | Varchar(10) | Bắt buộc nhập |
| GioiTinh | Nvarchar(10) | Bắt buộc nhập |
| DienThoai | Varchar(10) | Bắt buộc nhập |
| MaGVQuanLi | Varchar(10) | Khoá phụ |

### 2.2 Bảng Học Viên

|  |  |  |
| --- | --- | --- |
|  | **Học Viên** |  |
| **Mã** | **Kiểu dữ liệu** | **Ghi chú** |
| MaHocVien | Varchar(10) | Khóa chính  Bắt buộc nhập |
| TenHocVien | Nvarchar(50) | Bắt buộc nhập |
| NgaySinh | Varchar(10) | Bắt buộc nhập |
| TinhTrang | Nvarchar(50) | Bắt buộc nhập |
| MaLop | Varchar(10) | Bắt buộc nhập |

### 2.3 Bảng tài khoản

|  |  |  |
| --- | --- | --- |
|  | **Tài khoản** |  |
| **Mã** | **Kiểu dữ liệu** | **Ghi chú** |
| TenDangNhap | Varchar(10) |  |
| MatKhau | Varchar(15) |  |
| Loai | Int |  |

### 2.4 Bảng Môn học

|  |  |  |
| --- | --- | --- |
|  | **Môn Học** | |
| **Mã** | **Kiểu dữ liệu** | **Ghi chú** |
| MaMonHoc | Varchar(10) | Khóa chính  Bắt buộc nhập |
| TenMonHoc | Nvarchar(50) | Bắt buộc nhập |
| SoChi | Int | Bắt buộc nhập |

**2.5 Bảng Kết Quả**

|  |  |  |
| --- | --- | --- |
|  | **Kết Quả** | |
| **Mã** | **Kiểu dữ liệu** | **Ghi chú** |
| MaKQHV | Varchar(10) | Khóa chính  Bắt buộc nhập |
| MaMonHoc | Varchar(10) | Bắt buộc nhập |
| LanThi | Varchar(1) | Bắt buộc nhập |
| DiemTB | Varchar(3) | Bắt buộc nhập |

## III. Các lớp

**1. Lớp Giáo Viên**

|  |  |  |
| --- | --- | --- |
|  | **Giảng Viên** |  |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Ghi chú** |
| maGv | string |  |
| tenGv | string |  |
| ngaySinh | string |  |
| gioiTinh | string |  |
| dienThoai | string |  |
| maGvQL | string |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| Them() | void | Nhập một GV vào DataBase |
| Sua() | void | Sửa một GV trong  DataBase |
| Xoa() | void | Xóa một GV trong DataBase |
| Thoat() | void | Thoát khỏi Quản Lí GV |

## 2. Lớp Học Viên

|  |  |  |
| --- | --- | --- |
|  | **Học Viên** |  |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Ghi chú** |
| maHv | string |  |
| tenHv | string |  |
| ngaySinh | string |  |
| tinhTrang | string |  |
| maLop | string |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| Tao() | void | Nhập một học viên vào DataBase |
| Sua() | void | Sửa một học viên trong DataBase |
| Xoa() | void | Xóa một học viên trong DataBase |
| Thoat() | void | Thoát khỏi Quản Lí Học Viên |

**3. Lớp Quản Lí KQ Sinh Viên**

|  |  |  |
| --- | --- | --- |
|  | **Kết Quả Học Viên** |  |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Ghi chú** |
| maKq | string |  |
| maMh | string |  |
| lanThi | string |  |
| diem | string |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| Tao() | void | Nhập một kq học viên vào DataBase |
| Sua() | void | Sửa một kq học viên trong DataBase |
| Xoa() | void | Xóa một kq học viên trong DataBase |
| Thoat() | void | Thoát khỏi Quản Lí KQ Học Viên |

**4. Lớp Tài Khoản**

|  |  |  |
| --- | --- | --- |
|  | **Tài Khoản** |  |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Ghi chú** |
| taiKhoan | string |  |
| tendangnhap | string |  |
| loai | int |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| Tao() | void | Nhập một tài khoản vào DataBase |
| Sua() | void | Sửa một tài khoản trong DataBase |
| Xoa() | void | Xóa một tài khoản trong DataBase |
| Thoat() | void | Thoát khỏi Quản Lí Tài Khoản |

**5. Lớp Quản Lí Môn Học**

|  |  |  |
| --- | --- | --- |
|  | **Quản Lí Môn Học** |  |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Ghi chú** |
| mamh | string |  |
| tenmh | string |  |
| sochi | int |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| Tao() | void | Nhập một môn học vào DataBase |
| Sua() | void | Sửa một môn học trong DataBase |
| Xoa() | void | Xóa một môn học trong DataBase |
| Thoat() | void | Thoát khỏi Quản Lí Môn học |

**6. Lớp Quản Lí Đang Nhập**

|  |  |  |
| --- | --- | --- |
|  | **Quản Lí Đăng Nhập** |  |
| **Tên thuộc tính** | **Kiểu dữ liệu** | **Ghi chú** |
| dangnhap | string |  |
| matkhau | string |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| DNhap() | void | Đăng nhập vào Database |
| Thoat() | void | Thoát khỏi đăng nhập |

**7. Lớp Quản Lí Form**

|  |  |  |
| --- | --- | --- |
|  | **Quản Lí Form** |  |
| **Tên phương thức** | **Kiểu dữ liệu trả về** | **Chức năng** |
| QLTK() | void | Đăng nhập vào quản lí tài khoản |
| QLGV() | void | Đăng nhập vào quản lí giáo viên |
| QLMH() | void | Đăng nhập vào quản lí môn học |
| QLSV() | void | Đăng nhập vào quản lí sinh viên |
| QLKQ() | void | Đăng nhập vào quản lí kết quả |

# CHƯƠNG 3: TRIỂN KHAI LẬP TRÌNH

1. **Lớp Giáo Viên**
   1. **Tạo**

private void btnTao\_Click(object sender, EventArgs e)

{

try

{

string maGv = tbMaGv.Text;

string tenGv = tbTenGv.Text;

string ngaySinh = tbNgaySinh.Text;

string gioiTinh = tbGioiTinh.Text;

string dienThoai = tbDienThoai.Text;

string maGvQL = tbMaGvQL.Text;

GiangVien giangVien = new GiangVien();

giangVien.MaGv = maGv;

giangVien.TenGv = tenGv;

giangVien.NgaySinh = ngaySinh;

giangVien.GioiTinh = gioiTinh;

giangVien.DienThoai = dienThoai;

giangVien.MaGvQuanli = maGvQL;

QuanLyGiangVienBUS quanLyGiangVienBUS = new QuanLyGiangVienBUS();

int kq = quanLyGiangVienBUS.ThemGiangVien(giangVien);

if (kq > 0)

{

MessageBox.Show("Thêm thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Taotaikhoan: " + ex.ToString());

}

}

**1.2 Sửa**

private void btnSua\_Click(object sender, EventArgs e)

{

try

{

GiangVien giangVien = getInForForm();

QuanLyGiangVienBUS quanLyGiangVienBUS = new QuanLyGiangVienBUS();

int kq = quanLyGiangVienBUS.SuaGiangVien(giangVien);

if (kq > 0)

{

MessageBox.Show("Sửa thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Sửa thất bại");

}

}

**1.3 Xoá**

private void btnXoa\_Click(object sender, EventArgs e)

{

try

{

QuanLyGiangVienBUS quanLyGiangVienBUS = new QuanLyGiangVienBUS();

int kq = quanLyGiangVienBUS.XoaGiangVien(rowSelected.MaGv);

if (kq > 0)

{

MessageBox.Show("Xoá thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Xoá thất bại");

}

}

**2 Lớp Quản Lí Kết Quả Học Viên**

## 2.1 Tạo

private void btnTao\_Click(object sender, EventArgs e)

{

try

{

string maKq = tbMaKQSV.Text;

string maMh = tbMaMH.Text;

string lanThi = tbLanthi.Text;

string diem = tbDiem.Text;

KetQuaSinhVien ketQuaSinhVien = new KetQuaSinhVien();

ketQuaSinhVien.MaKqSv = maKq;

ketQuaSinhVien.MaMH = maMh;

ketQuaSinhVien.LanThi = lanThi;

ketQuaSinhVien.DiemTb = diem;

QuanLyKetQuaSinhVienBUS quanLyKetQuaSinhVienBUS = new QuanLyKetQuaSinhVienBUS();

int kq = quanLyKetQuaSinhVienBUS.ThemKetQuaSinhVien(ketQuaSinhVien);

if (kq > 0)

{

MessageBox.Show("Thêm thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Taotaikhoan: " + ex.ToString());

}

}

### 2.2 Sửa

private void btnSua\_Click(object sender, EventArgs e)

{

try

{

KetQuaSinhVien ketQuaSinhVien = getInForForm();

QuanLyKetQuaSinhVienBUS quanLyKetQuaSinhVienBUS = new QuanLyKetQuaSinhVienBUS();

int kq = quanLyKetQuaSinhVienBUS.SuaKetQuaSinhVien(ketQuaSinhVien);

if (kq > 0)

{

MessageBox.Show("Sửa thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Sửa thất bại");

}

### }

### 2.3 Xóa

private void btnXoa\_Click(object sender, EventArgs e)

{

try

{

QuanLyKetQuaSinhVienBUS quanLyKetQuaSinhVienBUS = new QuanLyKetQuaSinhVienBUS();

int kq = quanLyKetQuaSinhVienBUS.XoaKetQuaSinhVien(rowSelected.MaKqSv);

if (kq > 0)

{

MessageBox.Show("Xoá thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Xoá thất bại");

}

}

## 3 Lớp Sinh viên

**3.1 Tạo**

private void btnTao\_Click(object sender, EventArgs e)

{

try

{

string maHv = tbMahv.Text;

string tenHv = tbTenhv.Text;

string ngaySinh = tbNgaysinh.Text;

string tinhTrang = tbTinhtrang.Text;

string maLop = tbMalop.Text;

SinhVien sinhVien = new SinhVien();

sinhVien.MaHV = maHv;

sinhVien.TenHV = tenHv;

sinhVien.NgaySinh = ngaySinh;

sinhVien.TinhTrang = tinhTrang;

sinhVien.MaLop = maLop;

QuanLySinhVienBUS quanLySinhVienBUS = new QuanLySinhVienBUS();

int kq = quanLySinhVienBUS.ThemSinhVien(sinhVien);

if (kq > 0)

{

MessageBox.Show("Thêm thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Taotaikhoan: " + ex.ToString());

}

}

### 3.2 Sửa

private void btnSua\_Click(object sender, EventArgs e)

{

try

{

SinhVien sinhVien = getInForForm();

QuanLySinhVienBUS quanLySinhVienBUS = new QuanLySinhVienBUS();

int kq = quanLySinhVienBUS.SuaSinhVien(sinhVien);

if (kq > 0)

{

MessageBox.Show("Sửa thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Sửa thất bại");

}

}

### 3.3 Xóa

private void btnXoa\_Click(object sender, EventArgs e)

{

try

{

QuanLySinhVienBUS quanLySinhVienBUS = new QuanLySinhVienBUS();

int kq = quanLySinhVienBUS.XoaSinhVien(rowSelected.MaHV);

if (kq > 0)

{

MessageBox.Show("Xoá thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Xoá thất bại");

}

}

## 4 Lớp Môn Học

**4.1 Tạo**

private void btnTao\_Click(object sender, EventArgs e)

{

try

{

string mamh = tbMaMH.Text;

string tenmh = tbTenMH.Text;

int sochi = Convert.ToInt32(nudSoChi.TabIndex.ToString());

MonHoc monHoc = new MonHoc();

monHoc.MaMH = mamh;

monHoc.TenMH = tenmh;

monHoc.SoChi = sochi;

QuanLyMonHocBUS quanLyMonHocBUS = new QuanLyMonHocBUS();

int kq = quanLyMonHocBUS.ThemMonHoc(monHoc);

if (kq > 0)

{

MessageBox.Show("Thêm thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Taotaikhoan: " + ex.ToString());

}

}

}

}

## 5 Lớp Tài Khoản

**5.1 Tạo**

private void btnTao\_Click(object sender, EventArgs e)

{

try

{

string tenDangnhap = tbTendangnhap.Text;

string matKhau = tbMatkhau.Text;

int loai = Convert.ToInt32(cbLoai.SelectedItem.ToString());

TaiKhoan taiKhoan = new TaiKhoan();

taiKhoan.TenDangNhap = tenDangnhap;

taiKhoan.MatKhau = matKhau;

taiKhoan.Loai = loai;

QuanLyTaiKhoanDAO quanLyTaiKhoanAcess = new QuanLyTaiKhoanDAO();

int kq = quanLyTaiKhoanAcess.ThemTaiKhoan(taiKhoan);

if (kq > 0)

{

MessageBox.Show("Thêm thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Taotaikhoan: " + ex.ToString());

}

}

### 5.2 Sửa

private void btnSua\_Click(object sender, EventArgs e)

{

try

{

TaiKhoan taiKhoan = getInForForm();

QuanLyTaiKhoanBUS quanLyTaiKhoanBUS = new QuanLyTaiKhoanBUS();

int kq = quanLyTaiKhoanBUS.SuaTaiKhoan(taiKhoan);

if (kq > 0)

{

MessageBox.Show("Sửa thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Sửa thất bại");

}

}

### 5.3 Xóa

private void btnXoa\_Click(object sender, EventArgs e)

{

try

{

QuanLyTaiKhoanBUS quanLyTaiKhoanBUS = new QuanLyTaiKhoanBUS();

int kq = quanLyTaiKhoanBUS.XoaTaiKhoan(rowSelected.TenDangNhap);

if (kq > 0)

{

MessageBox.Show("Xoá thành công");

loaddata();

ClearDataForm();

}

}

catch (Exception ex)

{

MessageBox.Show("Xoá thất bại");

}

}

## 6 Lớp Quản Lí Form

### 6.1 QLTK

private void btnQLTK\_Click(object sender, EventArgs e)

{

QuanLyTaiKhoan quanLyTaiKhoan = new QuanLyTaiKhoan();

quanLyTaiKhoan.Show();

}

### 6.2 QLGV

private void btnQLGV\_Click(object sender, EventArgs e)

{

QuanLyGiangVien quanLyGiangVien = new QuanLyGiangVien();

quanLyGiangVien.Show();

### }

### 6.3 QLSV

private void btnQLSV\_Click(object sender, EventArgs e)

{

QuanLySinhVien quanLySinhVien = new QuanLySinhVien();

quanLySinhVien.Show();

### }

### 6.4 QLMH

private void btnQLMH\_Click(object sender, EventArgs e)

{

QuanLyMonHoc quanLyMonHoc = new QuanLyMonHoc();

quanLyMonHoc.Show();

### }

### 6.5 QLKQ

private void btnQLKQ\_Click(object sender, EventArgs e)

{

QuanLyKetQuaSinhVien quanLyKetQuaSinhVien = new QuanLyKetQuaSinhVien();

quanLyKetQuaSinhVien.Show();

### }

**7. GiangVien.cs**

public class GiangVien

{

private string \_maGv;

private string \_tenGv;

private string \_ngaySinh;

private string \_gioiTinh;

private string \_dienThoai;

private string \_maGvQuanli;

public string MaGv { get => \_maGv; set => \_maGv = value; }

public string TenGv { get => \_tenGv; set => \_tenGv = value; }

public string NgaySinh { get => \_ngaySinh; set => \_ngaySinh = value; }

public string GioiTinh { get => \_gioiTinh; set => \_gioiTinh = value; }

public string DienThoai { get => \_dienThoai; set => \_dienThoai = value; }

public string MaGvQuanli { get => \_maGvQuanli; set => \_maGvQuanli = value; }

}

**8. KetQuaSinhVien.cs**

public class KetQuaSinhVien

{

private string \_maKqSv;

private string \_maMH;

private string \_lanThi;

private string \_diemTb;

public string MaKqSv { get => \_maKqSv; set => \_maKqSv = value; }

public string MaMH { get => \_maMH; set => \_maMH = value; }

public string LanThi { get => \_lanThi; set => \_lanThi = value; }

public string DiemTb { get => \_diemTb; set => \_diemTb = value; }

}

**9. MonHoc.cs**

public class MonHoc

{

private string \_maMH;

private string \_tenMH;

private int \_soChi;

public string MaMH { get => \_maMH; set => \_maMH = value; }

public string TenMH { get => \_tenMH; set => \_tenMH = value; }

public int SoChi { get => \_soChi; set => \_soChi = value; }

}

**10. SinhVien.cs**

public class SinhVien

{

private string \_maHV;

private string \_tenHV;

private string \_ngaySinh;

private string \_tinhTrang;

private string \_maLop;

public string MaHV { get => \_maHV; set => \_maHV = value; }

public string TenHV { get => \_tenHV; set => \_tenHV = value; }

public string NgaySinh { get => \_ngaySinh; set => \_ngaySinh = value; }

public string TinhTrang { get => \_tinhTrang; set => \_tinhTrang = value; }

public string MaLop { get => \_maLop; set => \_maLop = value; }

}

**11. TaiKhoan.cs**

public class TaiKhoan

{

private string \_tendangnhap;

private string \_matkhau;

private int \_loai;

public string TenDangNhap { get => \_tendangnhap; set => \_tendangnhap = value; }

public string MatKhau { get => \_matkhau; set => \_matkhau = value; }

public int Loai { get => \_loai; set => \_loai = value; }

**12. Provider.cs**

public class Provider

{

static string ConnectionString = @"Data Source=ADMIN\MISASME2020;Initial Catalog=QLTruongHocDB;Integrated Security=True";

private SqlConnection connection;

private SqlConnection GetConnection()

{

return connection;

}

private void SetConnection(SqlConnection value)

{

connection = value;

}

public void Connect()

{

try

{

if (GetConnection() == null)

SetConnection(new SqlConnection(ConnectionString));

if (GetConnection().State != ConnectionState.Closed)

GetConnection().Close();

GetConnection().Open();

}

catch (SqlException ex)

{

throw ex;

}

}

public void DisConect()

{

if (GetConnection() != null && GetConnection().State == ConnectionState.Open)

GetConnection().Close();

}

public int ExecuteNonQuery(CommandType cmdType, string strSql, params SqlParameter[] parameters)

{

int nRow = 0;

try

{

Connect();

SqlCommand command = GetConnection().CreateCommand();

command.CommandType = cmdType;

command.CommandText = strSql;

if (parameters != null && parameters.Length > 0)

command.Parameters.AddRange(parameters);

nRow = command.ExecuteNonQuery();

}

catch (SqlException ex)

{

throw;

}

finally

{

DisConect();

}

return nRow;

}

public DataTable Select(CommandType cmdType, string strSql, params SqlParameter[] parameters)

{

DataTable dt = new DataTable();

try

{

Connect();

SqlCommand command = GetConnection().CreateCommand();

command.CommandType = cmdType;

command.CommandText = strSql;

if (parameters != null && parameters.Length > 0)

command.Parameters.AddRange(parameters);

SqlDataAdapter adapter = new SqlDataAdapter(command);

adapter.Fill(dt);

return dt;

}

catch (SqlException ex)

{

throw ex;

}

finally

{

DisConect();

}

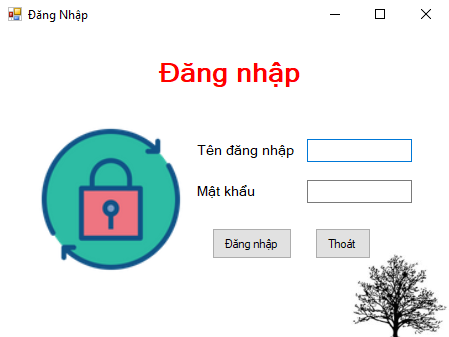
return dt;

}

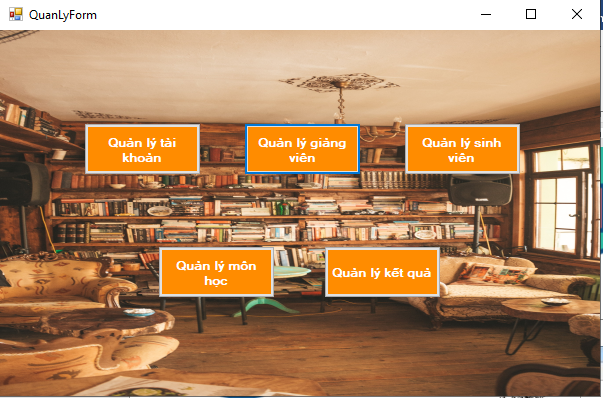
}

# CHƯƠNG 4: MINH HỌA ỨNG DỤNG

## I. Đăng Nhập

****

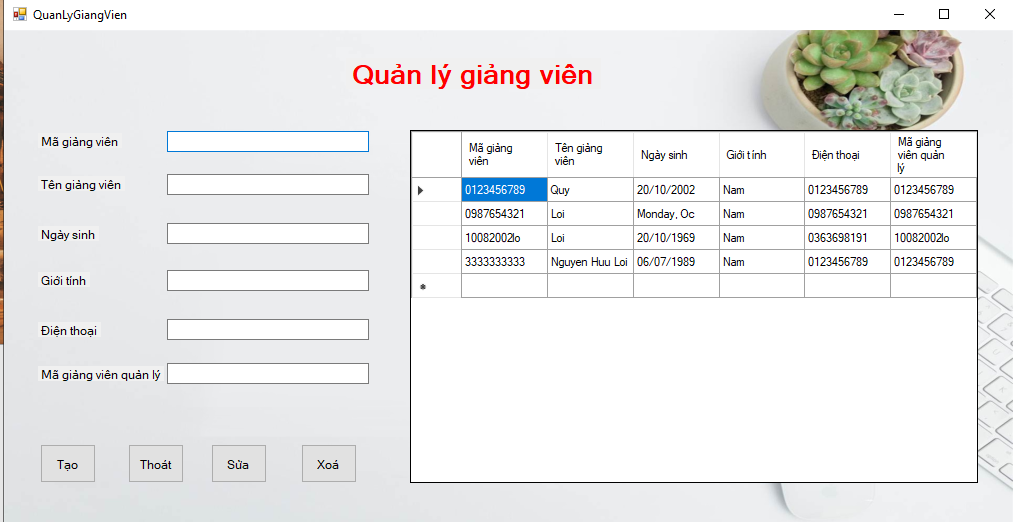
## II. Form Quản Lí



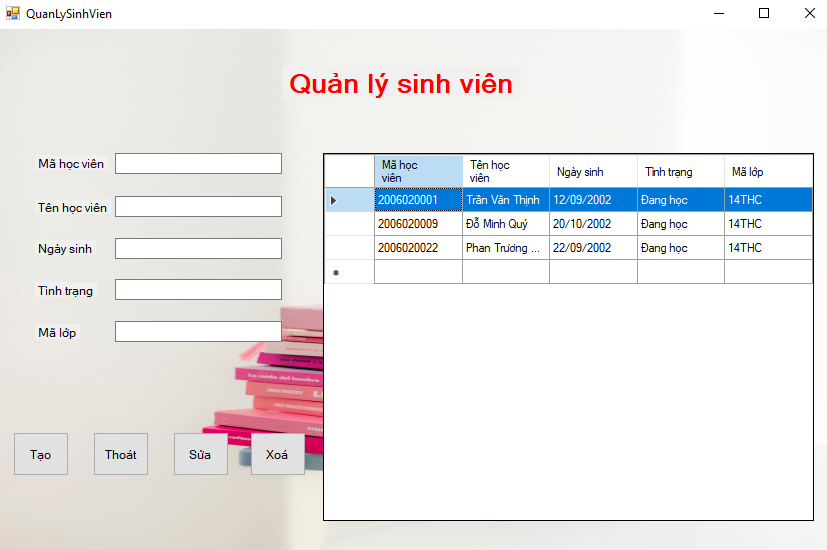
## III. Form Quản lý tài khoản

## 

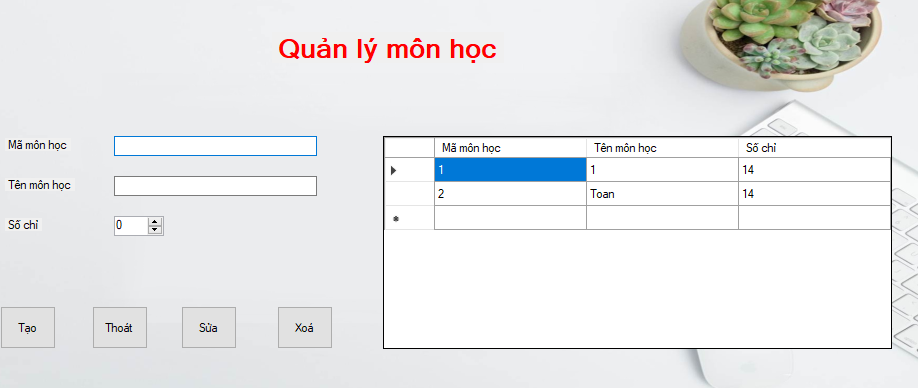
## IV. Form Quản lí GV

**

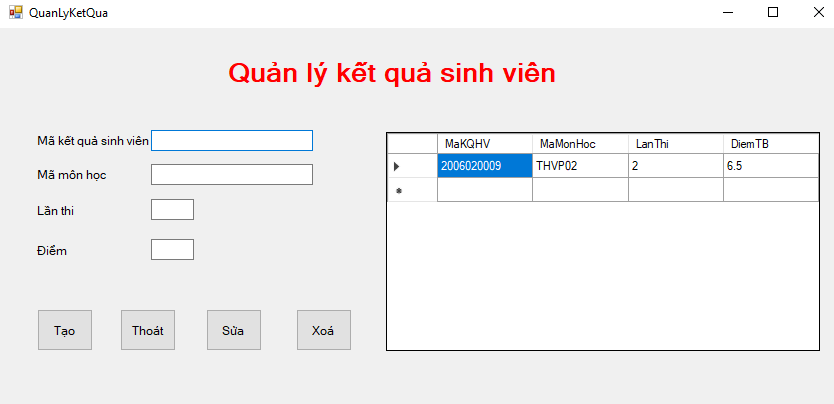
## V. Form Quản lý sinh viên

**

## VI.Form Quản lý môn học

**

**VII.Form Quản Lí Kết quả**

****

# TÀI LIỆU THAM KHẢO

1.Diễn đàn hỏi đáp lập trình http://stackoverflow.com/

# NHẬN XÉT CỦA GIÁO VIÊN

*TPHCM, ngày…tháng…năm 2022*

Giáo Viên