# Lập trình giao diện (GUI)

Giới thiệu

* Các tầng

User

Presentation (học phần giải quyết phần này)

Business

Data

Data Sources

* Kiến thức
  + Phân biệt được các tầng và chức năng mỗi tầng
  + Hiểu được các nguyên lý thiết kế giao diện Windows Form
  + Hiểu và giải thích được sự khác biệt của các loại giao diện. Từ đó đưa ra lựa chọn phù hợp cho phần mềm
  + Có kiến thức sử dụng ngôn ngữ C# trong môi trường Visual Studio.Net dùng trong lập trình xây dựng các ứng dụng ở mức cơ bản

## Tổng quan về lập trình giao diện

#### Mô hình đa tầng, đa lớp

* Trong lập trình ứng dụng, một phần mềm có thể được chia thành nhiều lớp ứng dụng gọi là các tầng (tier)
* Thường sử dụng mô hình 3 tầng
  + Presentation: gồm các thành phần xử lý giao diện (GUI)
  + Busuness: gồm các thành phần xử lý nghiệp vụ như
    - Business logic layer(BLL)
    - Data access layer(DAL)
    - Data transfer object(DTO)
  + Data tier

## Giới thiệu công nghệ .Net của Microsoft

* .Net Framwork bao gồm 2 thành phần chính
  + Lớp thư viện Framework
  + Bộ thực thi ngôn ngữ chung
* Các loại ngôn ngữ lập trình trên .Net
  + C#
  + Visual Basic
  + Visual C++
  + F#
  + Phython

### Vào ra

* Console.WriteLine("");
  + In ra màn hình
* Console.ReadLine();
  + Giữ màn hình

#### Lệnh nhập

* Lệnh nhập mặc định coi giá trị là kiểu chuỗi

Console.ReadLine();

* Nhập kiểu int

int.Parse(Console.ReadLine());

* Tương tự với các kdl khác

#### Lệnh xuất

* Lệnh xuất

Console.Write()

* Hoặc

Console.WriteLine()

* In giữ chỗ

Console.Write(“{0}, {1}, …, {n}”, var\_1, var\_2, var\_n)

#### Giữ màn hình

* Để lệnh hiển thị với người dùng, cần giữ màn hình

Console.ReadKey();

* Hoặc

Console.ReadLine();

# Thao tác với console

### Lý thuyết

#### Lý thuyết cơ bản

* namespace: định nghĩa một phạm vi (scope). Phạm vi namespace này cho phép chúng ta tổ chức chương trình lập trình và tạo ra các kiểu có tên duy nhất
* Trong namespace có thể khai báo
  + namespace khác
  + class
  + interface
  + struct
  + enum
  + delegate
* Đặc điểm của namespace
  + dùng để tổ chức các dự án lớn
  + các namspace được cách bằng toán tử “.”
  + có thể định nghĩa 1 namespace trong hai hay nhiều khai báo
  + namespace “global” là namespace gốc cho các namespace khác

A screen shot of a computer code

Description automatically generated with low confidence

* Để truy cập các thành phần bên trong namespace :
  + <namespaceName1>.<namespaceName2>…<class>
* Thuận tiện cho toàn chương trình
  + using <namespaceName1>.<namespaceName2>…;
* Từ khóa trong C#
  + namespace, using, class,…
* Kiểu dữ liệu. C# hỗ trợ 2 kiểu dữ liệu
  + Kdl giá trị (value): gồm các kdl xây dựng sẵn như kiểu số, ký tự, luận lý, liệt kê, cấu trúc

|  |  |
| --- | --- |
| Type | Examples |
| object | object o = new Stack(); |
| string | string s = “Hello”; |
| sbyte | sbyte val = 12; |
| short | short val = 12; |
| int | int val = 12; |
| long | long val1 = 12;  long val2 = 12L; |
| byte | byte val1 = 12;  byte val2 = 12U; |
| ushort | ushort val1 = 12;  ushort val2 = 34U; |
| uint | uint val1 = 12;  uint val2 = 12U; |
| ulong | ulong val1 = 12;  ulong val2 = 12U;  ulong val3 = 12L;  ulong val4 = 12UL; |
| float | float value = 1.23F; |
| double | double val1 = 1.23;  double val2 = 1.23F; |
| bool | bool value = true; |
| char | char value = ‘h’; |
| decimal | decimal value = 1.23M; |

* + Kiểu tham chiếu (reference): các kiểu dữ liệu do người dùng định nghĩa: class, interface, delegate, array
* Các ký tự đặc biệt

|  |  |  |  |
| --- | --- | --- | --- |
| Cách sử dụng | Ý nghĩa | Cách sử dụng | Ý nghĩa |
| \\ | \ | \f | form feed |
| \' | ' | \n | dòng mới |
| \'' | '' | \r | carriage return |
| \? | ? | \t | tab ngang |
| \a | tiếng beep | \v | tab dọc |
| \b | backspace |  |  |

* Kiểu dữ liệu liệt kê (Enum)
  + Sử dụng một biến có thể lấy một giá trị trong một bộ giá trị cố định
  + Đặt tên chung cho một tập các giá trị nguyên (tương tự như tập các hằng), làm cho chương trình rõ ràng, dễ hiểu hơn
  + Cú pháp

enum <TenKDL>

{

<Giatri1> [= hằng],

<Giatri1> [= hằng],

<Giatri1> [= hằng],

…..

<Giatri1> [= hằng]

}

* + Sử dụng

<TenKDL> <TenBien>;

<TenBien> = <TenKieuDuLieu>.<GiaTri>;

* Kiểu dữ liệu struct
  + Là một kiểu dữ liệu tự định nghĩa có thể chứa các thành viên dữ liệu (member) và các phương thức (method)
  + Cú pháp khai báo (không có ;)

struct <TenCauTruc>

{

<Khai báo các thành phần>

}

* + Một số đặc điểm
    - Không thể kế thừa
    - Không thể khai báo một phương thức khởi tạo không tham số cho một cấu trúc
    - Các thuộc tính của cấu trúc đều phải được khởi tạo giá trị
    - Khi khai báo các trường của cấu trúc, không thể khởi gán giá trị
    - Trường hợp gọi phương thức khởi tạo của cấu trúc với từ khóa new, các thuộc tính chưa khởi trị sẽ được tự động gán giá trị 0 hoặc null
* Biến, hằng
  + Phải gán giá trị trước khi sử dụng
  + Mức độ truy cập
    - Mặc định (private)

|  |  |
| --- | --- |
| Mức độ truy cập (Modifier) | Mô tả |
| private | Truy cập bị hạn chế trong phạm vi định nghĩa Class. Đây là loại phạm vi truy cập mặc định nếu không được chỉ định |
| protected | Truy cập bị giới hạn trong phạm vi định nghĩa của Class và bất kỳ các class con kế thừa |
| internal | Truy cập bị giới hạn trong phạm vi assembly của dự án hiện tại |
| protected internal | Truy cập bị giới hạn trong phạm vi Assambly hiện tại và trong class định nghĩa hoặc class con |
| public | Không có giới hạn truy cập |

* + Tầm vực của biến
    - Biến khai báo bên trong phương thức thì có phạm vi trong phương thức đó, gọi là biến cục bộ, không thể truy xuất các phương thức
    - Biến khai báo bên trong thân của một lớp thì có phạm vi là lớp đó
    - Trong một phạm vi hoạt động, không thể có hai biến cùng tên
  + Hằng
    - Hằng tương tự biến nhưng giá trị của hằng không thay đổi khi chương trình thực thi
    - Hằng phải được khởi tạo khi khai báo và chỉ khai báo một lần duy nhất trong chương trình và không được thay đổi giá trị
    - Cú pháp khai báo hằng

<const> <type> <CONSTNAME> = <value>

* Toán tử
  + Toán tử gán: =
  + Toán tử số học: +, -, \*, /, %
  + Toán tử tăng, giảm: ++, --, +=, -=, \*=, /=, %=
  + Toán tử quan hệ: ==, !=, >, >=, <, <=
  + Toán tử logic: &&(and), ||(or), !(not)
* Cấu trúc lựa chọn
  + Cấu trúc if…else
  + Cấu trúc if…else if
  + switch..case

switch(bien\_kiemtra){

case <gia\_tri1>:

//code

break;

…

default:

//code

break;

}

* Biểu thức điểu kiện
  + dieukien ? giatri1 : giatri2;
* Cấu trúc lặp
  + Lặp for
  + Lặp While (lặp không biết trước)
  + do…while
  + foreach (kdl bien\_daidien in ten\_th){

khối lệnh

}

#### Xử lý ngoại lệ

##### try…catch…finally

* Ngoại lệ là một lỗi phát sinh không mong muốn trong quá trình thực thi chương trình, còn gọi là lỗi
* Ngoại lệ xảy ra trong code hoặc một lời gọi hàm từ code
* Có nhiều nguyên nhân
  + thiếu bộ nhớ
  + thiếu tài nguyên
  + thao tác của người sử dụng
  + …
* Cú pháp xử lý ngoại lệ
* try…catch…finally

|  |
| --- |
| try{//lệnh thực hiện}catch([Exception]){//lệnh xử lý lỗi}finally{//lệnh kết thúc xử lý} |

* Ví dụ, chương trình chia 2 số nguyên với try…catch

|  |
| --- |
| try{int so1, so2;double kq;so1 = int.Parse(txtSo1.Text);so2 = int.Parse(txtSo2.Text);kq = so1 / so2;lbKetQua.Text = kq.ToString();}catch (FormatException){MessageBox.Show("Bạn phải nhập 2 số");}catch (DivideByZeroException ex){MessageBox.Show(ex.Message, "Lỗi chia cho 0");} |

##### throw

* Lệnh throw được dùng để ném ra một tín hiệu về sự bất bình thường làm phát sinh một ngoại lệ
* Khi ngoại lệ phát sinh, việc thực thi trong khi CLR sẽ được dừng và tìm kiếm một xử lý ngoại lệ
* Nếu không tìm được một trình xử lý ngoại lệ nào trong chương trình thì chương trình sẽ kết thúc

|  |
| --- |
| //giả sử tạo ra 1 exception nếu số nhập không nằm trong phạm //vi [0, 10]try{int so1, so2;double kq;so1 = int.Parse(txtSo1.Text);so2 = int.Parse(txtSo2.Text);if (int.Parse(txtSo1.Text) > 10 || int.Parse(txtSo1.Text) < 0|| int.Parse(txtSo2.Text) > 10 || int.Parse(txtSo2.Text) < 0)throw new Exception("Nằm ngoài phạm vi");kq = so1 + so2;lbKetQua.Text = kq.ToString();}catch (Exception ex){MessageBox.Show(ex.Message);} |

##### checked và unchecked

* Xử lý lỗi “tràn số”
* Lệnh checked: Kiểm tra và làm phát sinh ngoại lệ OverflowException khi có lỗi tràn số
* Lệnh unchecked: hủy bỏ kiểm tra, không làm phát sinh lỗi Overflow Exception khi có lỗi tràn số

##### Phát biểu using

* Phát biểu using: được sử dụng trong code nhằm tạo các đối tượng tự hủy một cách an toàn. Ví dụ:

|  |
| --- |
| using (StreamReader reader = new StreamReader("infor.txt")){string row;while((row = reader.ReadLine()) != null)Console.WriteLine(row);} |

##### Một số ngoại lệ hay gặp

|  |  |
| --- | --- |
| Tên lớp ngoại lệ | Ý nghĩa |
| MethodAccessException | Lỗi truy cập đến các thành phần (phương thức), không được phép truy cập |
| ArrayTypeMismatchException | Kiểu mảng không phù hợp |
| ArithmeticException | Lỗi liên quan đến các phép toán |
| DiveByZeroException | Lỗi chia cho 0 |
| FormatException | Lỗi sai định dạng một kiểu dữ liệu |
| IndexOutOfRangeException | Lỗi truy xuất ngoài chỉ số của mảng |
| InvalidCastException | Phép gán không hợp lệ |
| NullReferenceException | Tham chiếu đến một đối tượng null |
| OutOfMemoryException | Tràn bộ nhớ |
| OverflowException | Tràn phép toán |

### Bài tập

Viết chương trình nhập vào giá trị số nguyên dương h. Sau đó, tính và xuất ra tổng các số từ 1 đến n.

a, console

b, window form

* Gợi ý: Với phần Winform

Giả sử giao diện gồm: 1 ô text box, 1 button và 1 label hiển thị thông báo

A screenshot of a computer

Description automatically generated with low confidence

* Cần đọc dữ liệu từ textbox sau đó tính tổng.
* Ý tưởng ban đầu

|  |
| --- |
| int n;do{n = int.Parse(txtSo.Text);if (n <= 0){MessageBox.Show("Bạn phải nhập số nguyên dương", "Lỗi");txtSo.Clear();}} while (n <= 0); |

* Giả sử khi dữ liệu nhâp nhỏ hơn 0, ô text box sẽ bị clear. Lần lặp tiếp theo, nếu đọc dữ liệu sẽ trả ra ngoại lệ -> chương trình lỗi
* Do đó, có thể xử lý nhủ sau

|  |
| --- |
| int n;  n = int.Parse(txtSo.Text); if(n <= 0){MessageBox.Show("Bạn phải nhập số nguyên dương", "Lỗi");txtSo.Clear();return;} |

* Bắt lỗi với try…catch

|  |
| --- |
| int n;try{n = int.Parse(txtSo.Text);if (n <= 0){MessageBox.Show("Bạn phải nhập số nguyên dương", "Lỗi");txtSo.Clear();return;}int tong = 0;for (int i = 1; i <= n; i++)tong += i;lblKQ.Text = "Tổng các số nguyên từ 1 đến " + n.ToString() + " = " + tong.ToString();}catch (Exception ex){MessageBox.Show(ex.Message, "Thông báo");txtSo.Clear();} |

# Hướng đối tượng trong C#

## Giới thiệu

***Mục tiêu***

Mục tiêu của việc thiết kế một phần mềm;

* Tính tái sử dụng (reusability)
* Tính mở rộng (extensibility)
* Tính mềm dẻo (flexibility)

***Cách tiếp cận***

Các cách tiếp cận:

* Thiết kế theo hàm/ thủ tục
* Thiết kế theo module
* Thiết kế theo hướng đối tượng

***Lập trình hướng thủ tục***

* Chương trình là một hệ thống các thủ tục, hàm
* Cần xác định các hàm, định nghĩa hàm, gọi hàm
* Nhược điểm
  + Giá trị biến toàn cục trong từng hàm
  + Khó kiểm soát các hàm khi sửa đổi chương trình
  + Cần quản lý danh sách các hàm

***Lập trình hướng đối tượng***

* Mô hình hóa đối tượng từ thế giới thực thành các đối tượng có thể lưu trữ và xử lý được trong chương trình
* Đối tượng trong thế giới thực: là các thực thể được quan sát trong quá trình thu thập thông tin
  + Các đặc điểm
  + Các hoạt động
* Trừu tượng hóa thành
  + Các thuộc tính
  + Các hành động
* Phần mềm
  + thuộc tính -> dữ liệu (các trường – field)
  + hành động -> hàm (phương thức)

### Lớp

* Khái niệm mang tính trừu tượng -> biểu diễn một tập các đối tượng
* Lớp cũng có thuộc tính và phương thức
  + Thuộc tính của lớp tương ứng với thuộc tính mô tả trạng thái của đối tượng, là thành phần dữ liệu mô tả đối tượng
  + Phương thức của lớp tương ứng với các hành động của đối tượng, dùng để mô tả và mô tả các hành vi của đối tượng
* Trong ngôn ngữ lập trình
  + Đối tượng là thể hiện của lớp, là biến kiểu lớp

***Đặc trưng của lớp***

* Tính đóng gói
* Tính kế thừa
* Tính đa hình

#### Đóng gói

* Ràng buộc dữ liệu và phương thức vào các lớp riêng biệt
* Chỉ cung cấp một phương thức để giao tiếp

#### Kế thừa

* Định nghĩa lớp đối tượng dựa trên các lớp khác
* Lớp mới chỉ cần bổ sung thêm các thành phần riêng

#### Tính đa hình

* Sử dụng một giao diện chung cho nhiều phương thức khác nhau
* Phương thức cụ thể sẽ được xác định vào lúc chạy chương trình

### Lớp (class)

* Trong ngôn ngữ lập trình, lớp (class) là một cấu trúc mở rộng, được định nghĩa để tạo nên một kiểu dữ liệu mới
* Một lớp có
  + các thành phần dữ liệu
  + thuộc tính
  + phương thức khởi tạo
  + phương thức tĩnh
  + …

***So sánh với struct***

|  |  |
| --- | --- |
| class | struct |
| *Có tính kế thừa* | *Không có tính kế thừa* |
| Mức độ mặc định các member: private | Mức độ mặc định các member: public |
|  |  |

#### Khai báo

|  |
| --- |
| [MucTruyCap] class TenLop [:LopCoSo]{Khai báo các trường dữ liệu (mức truy cập, các biến)Định nghĩa các phương thức, thuộc tính (mức truy cập)} |

|  |  |
| --- | --- |
| Mức truy cập | Phạm vi |
| public | Có thể truy xuất từ mọi nơi |
| internal | Truy xuất trong phạm vi lớp, các lớp thừa kế và trong cùng một khối assembly (file .dll, .exe) |
| protected | Truy xuất trong phạm vi lớp và các lớp thừa kế |
| private | Chỉ truy xuất trong nội bộ lớp |
| protected internal | Có thể truy xuất từ mọi nơi |

#### Nạp chồng

* Là các phương thức cùng tên, khác tham số

|  |
| --- |
| class HinhTron{private double bankinh;public HinhTron(double ban\_kinh){bankinh = ban\_kinh;}public double TinhDienTich(){return bankinh \* bankinh \* Math.PI;}public double TinhDienTich(double bk){return bk \* bk \* Math.PI;}} |

#### Phương thức khởi tạo (constructor)

* class không có phương thức khởi tạo
  + Trình biên dịch sẽ tự động sử dụng phương thức khởi tạo mặc định
  + Các thuộc tính không được khởi tạo trong phương thức khởi tạo sẽ được khởi tạo mặc định
    - Kiểu số: 0
    - Kiểu luận lý: false
    - Kiểu đối tượng: null

***Khởi tạo mặc định***

|  |
| --- |
| //ví dụ, khởi tạo mặc định không tham sốclass HinhTron{private double bankinh;public HinhTron(){bankinh = 1.0;}public double TinhDienTich(){return bankinh \* bankinh \* Math.PI;}} |

***Khởi tạo nạp chồng***

|  |
| --- |
| //ví dụ, nạp chồng 2 phương thức khởi tạoclass HinhTron{private double bankinh;public HinhTron(){bankinh = 1.0;}public HinhTron(double ban\_kinh){bankinh = ban\_kinh;} } |

***Khởi tạo sao chép***

* Phương thức khởi tạo sao chép: khởi gán giá trị cho đối tượng mới bằng cách sao chép dữ liệu của đối tượng cùng kiểu đã tồn tại

|  |
| --- |
| //ví dụ, nạp chồng 2 phương thức khởi tạoclass HinhTron{private double bankinh;public HinhTron(){bankinh = 1.0;}public HinhTron(HinhTron ht){bankinh = ht.bankinh;} } |

#### Thuộc tính

* Mục đích: tăng sức mạnh của tính đóng gói
* Các biến dữ liệu của lớp thường được che dấu ở mức độ truy cập private
* Khai báo properties -> truy cập, thay đổi giá trị của các trường dữ liệu mà không cần truy cập trực tiếp vào các trường đó
* Mỗi property có các mức độ truy cập:
  + get: chỉ cho phép đọc giá trị của trường dữ liệu
  + set: cho phép gán giá trị cho trường dữ liệu
  + get và set: cho phép đọc và ghi

***Lợi ích***

* tăng sức mạnh của tính đóng gói
* bảo vệ dữ liệu của đối tượng, không cho phép gán các giá trị bất hợp lệ
* ví dụ, không cho phép gán giá trị âm cho bán kính

***Ví dụ về get, set***

|  |
| --- |
| //set và getpublic class Hinhtron{private double bankinh;public double Bankinh{get{return bankinh;}set {bankinh = value;}}}//sử dụngHinhtron ht = new Hinhtron();ht.Bankinh = 10;int bk = ht.Bankinh; |

## Bài tập

### Bài tập class

***Bài 1: Phân số***

Tạo lớp phân số bao gồm:

Thành phần **dữ liệu**: Tử số, mẫu số

**Phương thức**:

* Phương thức khởi tạo chuẩn
* Phương thức khởi tạo 2 tham số
* Cộng 2 phân số
* Trừ 2 phân số

Tạo lớp xử lý:

* Phương thức main
* Xử lý lỗi

***Gợi ý***

* Phương thức khởi tạo, phương thức get, set

|  |
| --- |
| //khởi tạo chuẩn, không tham sốpublic PHANSO(){tu = 0;mau = 1;}//khởi tạo 2 tham sốpublic PHANSO(int t, int m){tu = t;mau = m;}//phương thức get, setpublic int TuSo{get {return tuSo;}set{tuSo = value;}}public int MauSo{get {return mauSo;}set{mauSo = value;}} |

* Hàm xử lý cộng, trừ

|  |
| --- |
| //cộng hai phân sốpublic PhanSo Cong(PhanSo p){PhanSo kq = new PhanSo();kq.tuSo = tuSo \* p.mauSo + p.tuSo \* mauSo;kq.mauSo = mauSo \* p.mauSo;return kq;}//phép trừ tương tự |

***Bài 2: Sinh viên***

Xây dựng lớp sinh viên, biết rằng mỗi sinh viên có:

* Thành phần dữ liệu: msv, họ, tên, điểm tb
* Nhập/xuất thông tin sv
* Kết quả đậu/rớt theo dtb
* Xây dựng project với **console** và **winform,** có các chức năng
  + Tạo một đối tượng sinh viên
  + Nhập thông tin cho một sinh viên
  + Xuất thông tin kết quả đậu/rớt của sinh viên
  + Đổi tên sinh viên thành tên mới nhập từ bàn phím