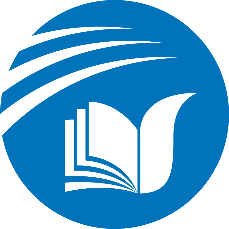
**ĐẠI HỌC THÁI NGUYÊN**

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

------- 🙞🕮🙜 -------



**BÁO CÁO THỰC TẬP CƠ SỞ**

**ĐỀ TÀI:**

“TÌM HIỂU NGÔN NGỮ LẬP TRÌNH C# VÀ XÂY DỰNG GAME DÒ MÌN.”

Sinh viên: Ngô Quang Việt

Lớp: Kỹ thuật phần mềm K15A

Giảng viên: Tô Hữu Nguyên

# LỜI MỞ ĐẦU

Để hoàn thành bài thực tập cơ sở này, Em xin cảm ơn Thầy ThS Tô Hữu Nguyên đã tận tình hướng dẫn và giúp đỡ tác giả trong thời gian thực hiện bài báo cáo thực tập cơ sở.

Với lòng biết ơn sâu sắc nhất, Em chân thành cảm ơn quý Thầy, Cô trong Khoa Công Nghệ Thông Tin, Trường Đại Học Công Nghệ Thông Tin và Truyền thông đã tận tình và tâm huyết truyền đạt kiến thức quý báu trong những năm tác gải học tập. Với vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình nghiên cứu học tập mà còn là hành trang quý báu để tác giả bước vào đời một cách vững chắc và tự tin.

Xin chân thành cảm ơn đến các bạn bè đã giúp đỡ tài liệu, trao đổi học thuật mới để thực hiện bài thực tập cơ sở này.

Sau cùng, xin kính chúc quý Thầy Cô thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau. Đồng kính chúc các thầy cô và các bạn luôn dồi dào sức khỏe, đạt được nhiều thành công tốt đẹp trong công việc.

Em xin chân thành cảm ơn!

Người thực hiện

Ngô Quang Việt

# CHƯƠNG I: CƠ SỞ LÍ THUYẾT

## Cơ bản ngôn ngữ C#

### Giới thiệu C#

Ngôn ngữ C# khá đơn giản, chỉ khoảng hơn 80 từ khóa và hơn mười mấy kiểu dữ liệu được dựng sẵn. Tuy nhiên, ngôn ngữ C# có ý nghĩa to lớn khi nó thực thi những khái niệm lập trình hiện đại. C# bao gồm tất cả những hỗ trợ cho cấu trúc, thành phần component, lập trình hướng đối tượng. Những tính chất đó hiện diện trong một ngôn ngữ lập trình hiện đại. Hơn nữa ngôn ngữ C# được xây dựng trên nền tảng hai ngôn ngữ mạnh nhất là C++ và Java.

Tóm lại, C# có các đặc trong sau đây:

* 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 hướng module
* C# sẽ trở nên phổ biến
  + - 1. ***C# là ngôn ngữ đơn giản***
* C# loại bỏ được một vài sự phức tạp và rối rắm của các ngôn ngữ C++ và Java.
* C# khá giống C / C++ về diện mạo, cú pháp, biểu thức, toán tử.
* Các chức năng của C# được lấy trực tiếp từ ngôn ngữ C / C++ nhưng được cải tiến để làm cho ngôn ngữ đơn giản hơn.
  + - 1. ***C# là ngôn ngữ hiện đại***

C# có được những đặc tính của ngôn ngữ hiện đại như:

* Xử lý ngoại lệ
* Thu gom bộ nhớ tự động
* Có những kiểu dữ liệu mở rộng
* Bảo mật mã nguồn
  + - 1. ***C# là ngôn ngữ hướng đối tượng***

C# hỗ trợ tất cả những đặc tính của ngôn ngữ hướng đối tượng là:

* Sự đóng gói
* Sự kế thừa
* Đa hình
  + - 1. ***C# là ngôn ngữ mạnh mẽ và mềm dẻo***
* Với ngôn ngữ C#, chúng ta chỉ bị giới hạn ở chính bản thân của chúng ta. Ngôn ngữ này không đặt ra những ràng buộc lên những việc có thể làm.
* C# được sử dụng cho nhiều dự án khác nhau như: tạo ra ứng dụng xử lý văn bản, ứng dụng đồ họa, xử lý bảng tính; thậm chí tạo ra những trình biên dịch cho các ngôn ngữ khác.
* C# là ngôn ngữ sử dụng giới hạn những từ khóa. Phần lớn các từ khóa dùng để mô tả thông tin, nhưng không vì thế mà C# kém phần mạnh mẽ. Chúng ta có thể tìm thấy rằng ngôn ngữ này có thể được sử dụng để làm bất cứ nhiệm vụ nào.
  + - 1. ***C# là ngôn ngữ hướng module***
* Mã nguồn của C# được viết trong Class (lớp). Những Class này chứa các Method (phương thức) thành viên của nó.
* Class (lớp) và các Method (phương thức) thành viên của nó có thể được sử dụng lại trong những ứng dụng hay chương trình khác.
  + - 1. ***C# sẽ trở nên phổ biến***

C# mang đến sức mạnh của C++ cùng với sự dễ dàng của ngôn ngữ Visual Basic.

### Biến, Hằng, Toán tử

* + - 1. ***Biến***

1. Khái niệm:

* Biến là một vùng lưu trữ ứng với một kiểu dữ liệu.
* Biến có thể được gán giá trị và cũng có thể thay đổi giá trị trong khi thực hiện các lệnh của chương trình.

1. Khai báo biến: Sau khi khai báo biến phải gán giá trị cho biến

<Kiểu\_Dữ\_Liệu> <tên\_biến> [=<giá\_trị>];

1. Ví dụ: khởi tạo gán giá trị một biến

class Bien{

static void Main(){

// khai bao khoi tao bien

int bien = 9;

System.Console.WriteLine(“Sau khi khoi tao bien ={0}”,bien);

// Gan gia tri cho bien

bien = 5;

// Xuat ra man hinh

System.Console.WriteLine(“Sau khi khoi tao bien ={0}”,bien);

}

}

* + - 1. ***Hằng***

1. Khái niệm:

* Hằng cũng là một biến nhưng giá trị hằng không thay đổi trong khi thực hiện các lệnh của chương trình.
* Hằng được phân ra làm 3 loại:
  + Giá trị hằng
  + Biểu tượng hằng
  + Kiểu liệt kê

1. Giá trị hằng:

Ví dụ: x = 100; // 100 được gọi là giá trị hằng

1. Biểu tượng hằng:

* Gán một tên hằng một giá trị hằng.
* Khai báo: <const><Kiểu\_Dữ\_Liệu><tên\_hằng>=<giá\_trị>;
* Ví dụ: Nhập vào bán kính, in ra ch vi diện tích hình tròn.

class **HinhTron**{

static void **main**(){

// Khai bao bieu tuong hang

const double PI = 3.14;

// Khai bao bien

int bankinh;

double chuvi, dientich;

string chuoi;

// Nhap gia tri cho bien chuoi

System.Console.Write("Nhap ban kinh hinh tron: ");

chuoi = System.Console.ReadLine();

// Doi chuoi thanh so va gan vao bien so

bankinh = System.Convert.ToInt32(chuoi);

// Gan gia tri cho bien

chuvi = 2 \* bankinh \* PI;

dientich = bankinh \* bankinh \* PI;

// Xuat ra man hinh

System.Console.WriteLine("Chu vi hinh tron = {0:0.00}", chuvi);

System.Console.WriteLine("Dien tich hinh tron = {0:0.00}", dientich);

}

}

1. Kiểu liệt kê: là tập hợp các tên hằng có giá trị số không thay đổi

Khai báo:

<enum><Tên\_Kiểu\_Liệt\_Kê>{

<tên\_hằng\_1> = <giá\_trị\_số\_1>,

<tên\_hằng\_1> = <giá\_trị\_số\_1>,

…,

};

Ví dụ:

enum NhietDoNuoc{

DoDong = 0, DoNguoi = 20, DoAm = 40, DoNong = 60, DoSoi = 100,

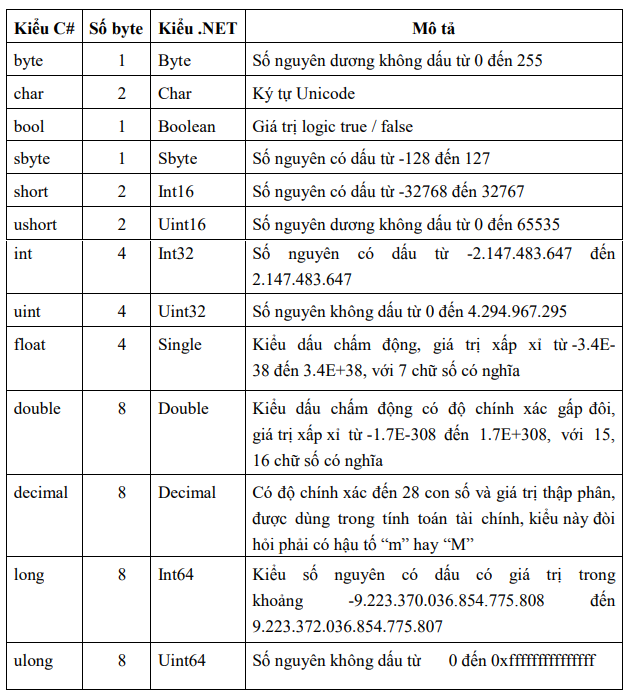
};

* + - 1. ***Toán tử***

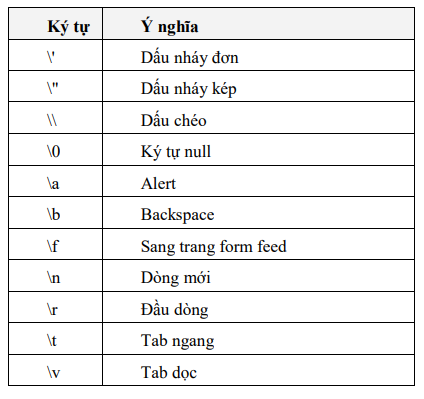
1. Toán tử toán học: +, - , \* , / , %
2. Toán tử tăng / giảm: += , -= , \*=, /=, %=
3. Toán tử tăng / giảm 1 đơn vị: ++, --
4. Toán tử gán: =
5. Toán tử quan hệ: == , != , > , >= , < , <=
6. Toán tử logic: ! , && , ||
7. Toán tử 3 ngôi: (Điều\_Kiện) ? (Biểu\_Thức\_1) : (Biểu\_Thức\_2) ;

### Kiểu dữ liệu

C# chia kiểu dữ liệu thành hai tập hợp kiểu dữ liệu chính:

* Kiểu xây dựng sẵn: do ngôn ngữ cung cấp cho người lập trình.
* Kiểu do người dùng định nghĩa: do người lập trình tạo ra.
  + - 1. ***Kiểu dữ liệu dựng sẵn***

Hình 1: Kiểu dữ liệu dựng sẵn



Hình 2: Kiểu dữ liệu đặc biệt

* + - 1. ***Chuyển đổi kiểu dữ liệu***

Ví dụ a:

short x = 10 ;

int y = x ; // chuyển đổi ngầm định

Ví dụ b:

short x ;

int y = 100 ;

x = (short) y ; // ép kiểu tường minh, trình biên dịch không báo lỗi

Ví dụ c:

short x ;

int y = 100 ;

x = y ; // không biên dịch, lỗi

### Cấu trúc điều khiển

* + - 1. ***Câu lệnh if…else***

1. Cú pháp:

if(Điều\_Kiện)

<Khối lệnh Điều\_Kiện đúng>

[else

<Khối lệnh Điều\_Kiện sai>]

1. Ví dụ: Dùng câu lệnh if…else

class Chan\_Le{

static void Main() {

// Khai bao va khoi tao bien

int bienDem = 9;

// Xuat ra man hinh

if (bienDem % 2 == 0)

Console.WriteLine("{0} la so chan", bienDem);

else

Console.WriteLine("{0} la so le", bienDem);

}

}

* + - 1. ***Câu lệnh if lồng nhau***

1. Cú pháp:

if (Điều\_Kiện\_1)

<Khối kệnh 1>

else if (Điều\_Kiện\_2)

<Khối kệnh 2.1>

else

<Khối kệnh 2.2>

1. Ví dụ:

class Thu\_Trong\_Tuan {

static void Main(){

// Khai bao va khoi tao bien

int thu = 5;

// 0: Chu nhat, 1: Thu hai, 2: Thu ba, 3: Thu tu,

// 4: Thu nam, 5: Thu sau, 6: Thu bay

if ((thu == 1) || (thu == 3) || (thu == 5))

Console.WriteLine("Day la ngay 2-4-6");

else if ((thu == 2) || (thu == 4) || (thu == 6))

Console.WriteLine("Day la ngay 3-5-7");

else

Console.WriteLine("Day la ngay chu nhat");

}

}

* + - 1. ***Câu lệnh switch***

1. Cú pháp:

switch (Biểu\_Thức) {

case <giá\_trị\_1> :

< Khối lệnh 1>

<Lệnh Nhảy>

case <giá\_trị\_2>:

< Khối lệnh 2>

<Lệnh Nhảy>

…. [default:

< Khối lệnh khác>]

}

1. Ví dụ:

class Thu{

static void Main(){

int thu = 5;

switch (thu){

case 0:

Console.WriteLine("Chu nhat");

break;

case 1:

Console.WriteLine("Thu hai");

break;

case 2:

Console.WriteLine("Thu ba");

break;

case 3:

Console.WriteLine("Thu tu");

break;

case 4:

Console.WriteLine("Thu nam");

break;

case 5:

Console.WriteLine("Thu sau");

break;

case 6:

Console.WriteLine("Thu bay");

break;

default:

Console.WriteLine("Khong phai la thu trong tuan");

break;

}

}

}

### Vòng lặp trong C#

Có thể có một tình huống khi chúng ta cần thực hiện một khối code một vài lần, điều này có thể được xem như một vòng lặp.

C# có 3 kỹ thuật lặp linh động. Bạn có thể sử dụng một trong ba vòng lặp sau:

* + - Vòng lặp while
    - Vòng lặp do…while
    - Vòng lặp for
      1. ***Vòng lặp for***

1. Cú pháp:

for ([Khởi\_tạo] ; [Điều\_kiện] ; [Bước\_lặp])

< Khối lệnh>

1. Ví dụ:

class UsingFor{

static void Main(){

for (int i = 1; i <= 30; i++){

if (i % 10 == 0)

Console.Write("{0} \n\r", i);

else

Console.Write("{0} ", i);

}

}

}

* + - 1. ***Vòng lặp while***

1. Cú pháp:

while (Điều\_Kiện)

< Khối lệnh>

1. Ví dụ:

class UsingWhile{

static void Main(){

int i = 1;

while (i <= 10){

Console.WriteLine("i = {0}", i);

i++; // tang bien dem,

}

}

}

* + - 1. ***Vòng lặp do…while***

1. Cú pháp:

do

< Khối lệnh>

while (Điều\_Kiện) ;

1. Ví dụ:

class UsingDoWhile{

static void Main(){

// Khai bao va khoi tao bien dem

int i = 1;

// Xuat ra man hinh

do{

Console.WriteLine("i = {0}", i);

i++; // tang bien dem

}while (i <= 10);

}

}Một số kiến thức cần nhớ

* + - 1. ***Namespace***
* .NET cung cấp một thư viện các class rất đồ sộ, trong đó Console là một class nhỏ trong thư viện các class này.
* Mỗi class có một tên riêng, vì vậy người lập trình không thể nào nhớ hết tên các class trong .NET. Để giải quyết vấn đề này là việc tạo ra một namespace, namespace sẽ hạn chế phạm vi của một tên, làm cho tên này chỉ có ý nghĩa trong vùng đã định nghĩa.
  + - 1. ***Từ khóa using***
* Để không phải viết namespace cho từng đối tượng, ta dùng từ khóa using.
* Ta có thể dùng dòng lệnh sau ở đầu chương trình: using System ;
  + - 1. ***Từ khóa static***

Từ khóa static chỉ ra rằng hàm Main() có thể được gọi mà không cần phải tạo đối tượng ChaoMung.

* + - 1. ***Từ khóa this***

Từ khóa this dùng để tham chiếu đến thể hiện hiện hành của đối tượng.

* + - 1. ***Chú thích***
* Một chương trình được viết tốt thì cần phải có chú thích các đoạn mã lệnh được viết.
* Mục đích chính là làm cho đoạn mã lệnh nguồn rõ ràng và dễ hiểu.
* Có 2 loại chú thích:
  + Chú thích một dòng: //
  + Chú thích nhiều dòng: /\* \*/
    - 1. ***Phân biệt chữ thường và chữ hoa***

C# là ngôn ngữ phân biệt chữ thường với chữ hoa.

* + - 1. ***Toán tử ‘.’***

Toán tử '. ' được sử dụng để truy cập đến phương thức hay dữ liệu trong một class và ngăn cách giữa tên class đến một namespace.

* + - 1. ***Câu lệnh***

Một chỉ dẫn lập trình đầy đủ được gọi là một câu lệnh.

Ví dụ: int bankinh = 5 ; *// một câu lệnh*

chuvi = 2 \* bankinh \* PI ; *// một câu lệnh khác*

* + - 1. ***Kiểu chuỗi ký tự***

Kiểu chuỗi ký tự là một mảng các ký tự.

1. Khai báo chuỗi hằng:

string <Tên\_chuỗi\_hằng> = <"Noi dung chuoi hang"> ;

Ví dụ: string tentuong = "Lap Trinh Ung Dung Kinh te" ;

1. Khai báo biến kiểu chuỗi:

string <Biến\_chuỗi> [= "Noi dung chuoi hang"] ;

Ví dụ: string hoten = "Nguyen Van A" ;

1. Nhập chuỗi:

<Biến\_chuỗi> = System.Console.ReadLine() ;

Ví dụ: hoten = System.Console.ReadLine() ;

1. Xuất chuỗi:

System.Console.WriteLine("Chuoi") ;

Ví dụ: System.Console.WriteLine("Do dai cua chuoi la:") ;

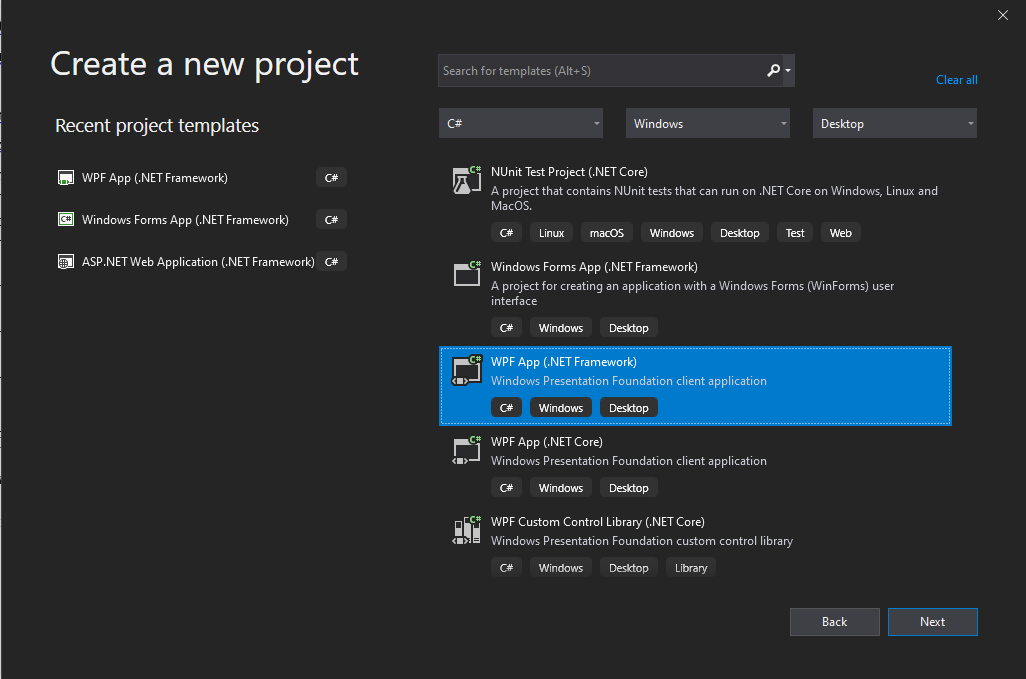
1. Một số thao tác trên chuỗi:

|  |  |
| --- | --- |
| ***Phương thức*** | ***Ý nghĩa*** |
| Length | Chiều dài của chuỗi |
| Substring() | Lấy chuỗi con |
| ToLower() | Trả về bản sao của chuỗi ở kiểu chữ thƣờng |
| ToUpper() | Trả về bản sao của chuỗi ở kiểu chữ IN HOA |

Hình 3: Một số thao tác trên chuỗi

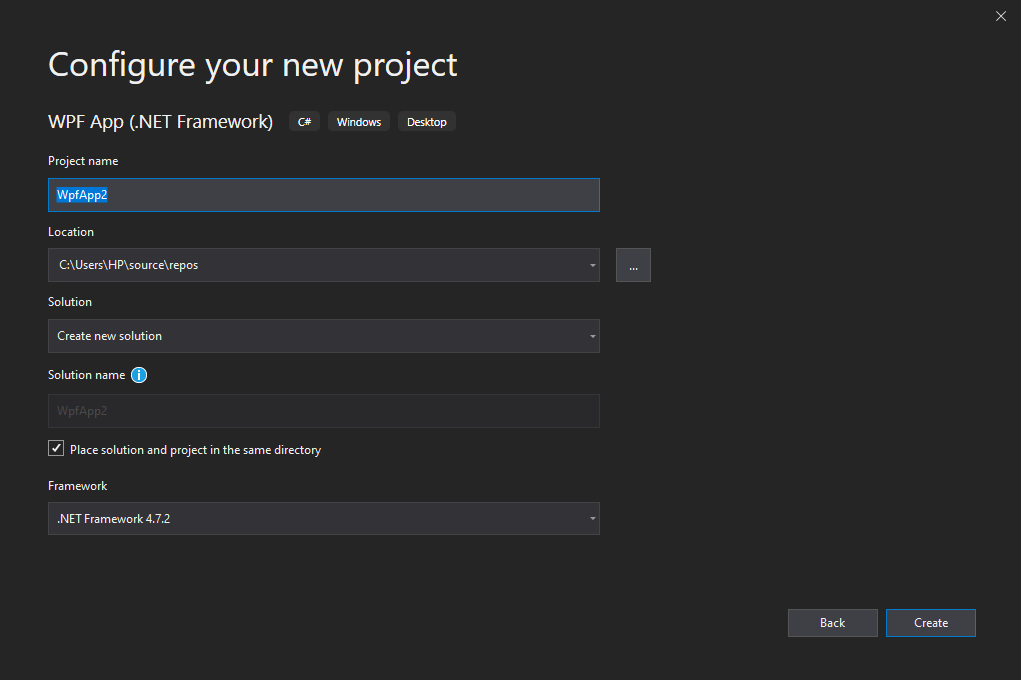
### Xây dựng Windows form application

### Khởi tạo chương trình WPF trên Visual Studio 2019



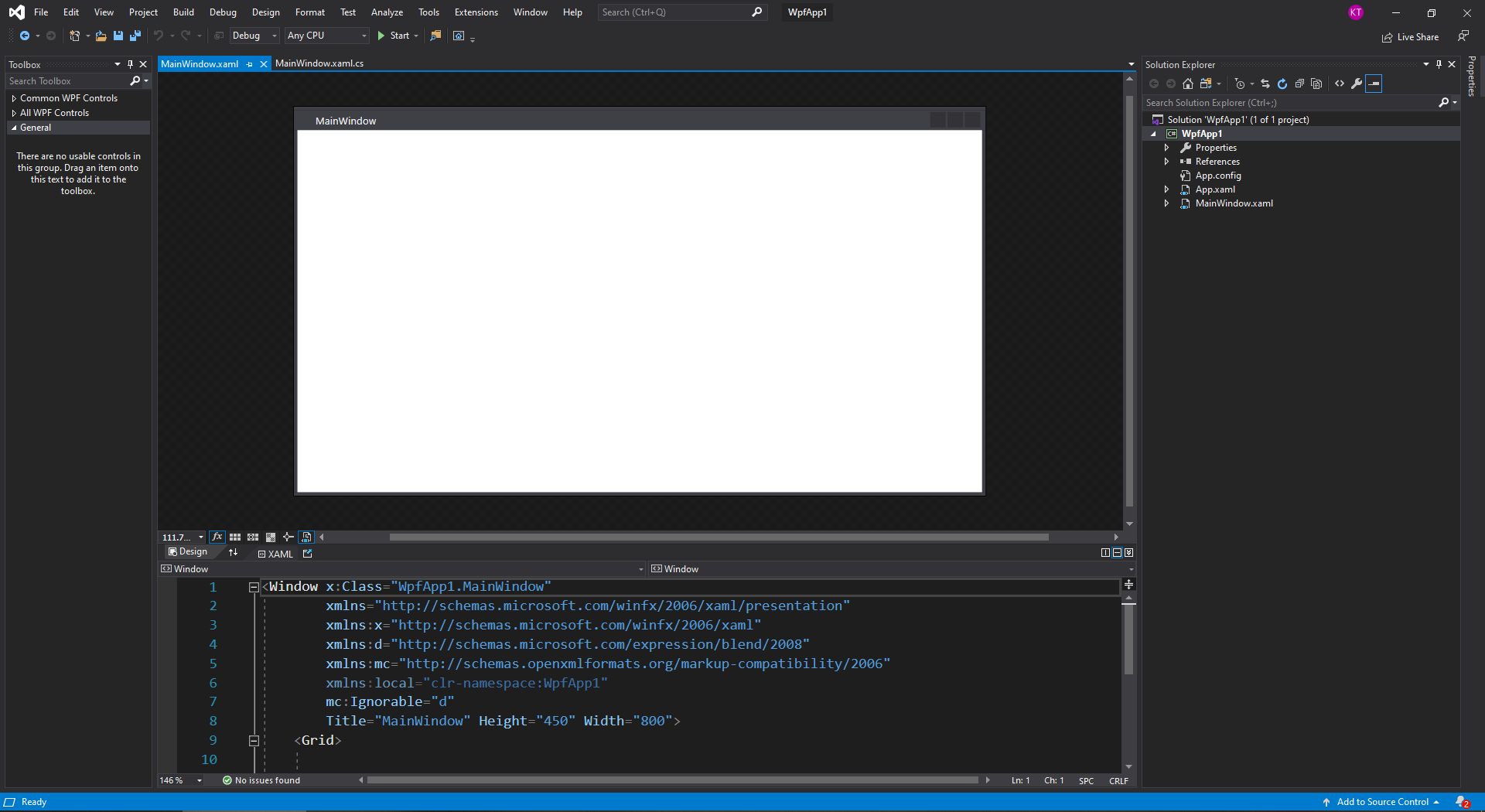
Hình 4: Khởi tạo project WPF

### Cấu hình cho project



Hình 5: Cấu hình cho project WPF

### Màn hình mặc định của project



Hình 6: Màn hình chính của project

* 1. **Mô hình MVVM trong lập trình C#**
     1. **Mô hình MVVM là gì?**

MVVM có thể nói là mô hình kiến trúc được rất nhiều các cư dân trong cộng đồng ưa chuộng. Điểm tinh hoa của kiến trúc này là ở ViewModel, mặc dù rất giống với Presenter trong MVP tuy nhiên có 2 điều làm nên tên tuổi của kiến trúc này đó là:

ViewModel không hề biết gì về View, một ViewModel có thể được sử dụng cho nhiều View (one-to-many). ViewModel sử dụng Observer design pattern để liên lạc với View (thường được gọi là binding data, có thể là 1 chiều hoặc 2 chiều tùy nhu cầu ứng dụng). Chính đặc điểm này MVVM thường được phối hợp với các thư viện hỗ trợ Reactive Programming hay Event/Data Stream, đây là triết lý lập trình hiện đại và hiệu quả phát triển rất mạnh trong những năm gần đây.

* + 1. **MVVM được hiểu như thế nào?**
       1. **View**

Tương tự như trong mô hình MVC, View là phần giao diện của ứng dụng để hiển thị dữ liệu và nhận tương tác của người dùng. Một điểm khác biệt so với các ứng dụng truyền thống là View trong mô hình này tích cực hơn. Nó có khả năng thực hiện các hành vi và phản hồi lại người dùng thông qua tính năng binding, command.

* + - 1. **Model**

Cũng tương tự như trong mô hình MVC. Model là các đối tượng giúp truy xuất và thao tác trên dữ liệu thực sự.

* + - 1. **ViewModel**

Lớp trung gian giữa View và Model. ViewModel có thể được xem là thành phần thay thế cho Controller trong mô hình MVC. Nó chứa các mã lệnh cần thiết để thực hiện data binding, command.

* + 1. **Ưu nhược điểm của mô hình MVVM**
       1. **Ưu điểm**
* MVVM sẽ tạo sự tương tác hiệu quả giữa designer và developer
* Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần thiết phải viết lại code quá nhiều
* Thể hiện tính chuyên nghiệp trong lập trình, phân tích thiết kế. Do được chia thành các thành phần độc lập nên giúp phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…
  + - 1. **Nhược điểm**
* Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần.
* Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel.
* Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc debug khi cơ sở dữ liệu phức tạp.
  + 1. **Kỹ thuật Data Binding trong C#**
* Nguồn dữ liệu là một object nào đó, nơi sử dụng dữ liệu là các điều khiển, Data Binding cho phép liên kết một điều khiển với 1 object hoặc danh sách các object. Điều khiển hiển thị thông tin chứ trong object.
* Khi người dùng thay đổi giá trị trên điều khiển, giá trị mới sẽ được caaph nhật ngay về object. Ở đây là sự đồng bộ dữ liệu diễn ra theo chiều điều kiển => object.
* Ở chiều ngược lại, khi thay đổi giá trị của object chưa chắc đã dẫn đến sự cập nhật của điều khiển. Để thưc hiện động bộ theo chiều ngược lại (tức là từ object => điều kiểm), chúng ta phải sử dụng một sử dụng hàng Mode.
* Nhiều điều khiển có thể cùng bind với 1 object. Nếu một trong số đó thay đổi giá trị của object, sự thay đổi này cần cập nhật đến tất cả các điều khiển khác. Đây là sự đồng bộ dữ liệu giữa các điều khiển.
  1. **Hệ quản trị cơ sở dữ liệu SQL Server**
     1. **Khái niệm SQL**

SQL Server chính là một hệ quản trị dữ liệu quan hệ sử dụng câu lệnh SQL để trao đổi dữ liệu giữa máy cài SQL Server và máy Client. Một Relational Database Management System – RDBMS gồm có: databases, datase engine và các chương trình ứng dụng dùng để quản lý các bộ phận trong RDBMS và những dữ liệu khác.

* + 1. **Các thành phần cơ bản trong SQL Server**

Các thành cơ bản trong SQL Server gồm có: Reporting Services, Database Engine, Integration Services, Notification Services, Full Text Search Service,… Tất cả kết hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc phân tích và lưu trữ dữ liệu trở nên dễ dàng hơn.

* Database Engine: Đây là một engine có khả năng chứa dữ liệu ở các quy mô dưới dạng support và table. Ngoài ra, nó còn có khả năng tự điều chỉnh ví dụ: trả lại tài nguyên cho ệ điều hành khi một user log off và sử dụng thêm các tài nguyên của máy khi cần.
* Integration Services: là tập hợp các đối tượng lập trình và các công cụ đồ họa cho việc sao chép, di chuyển và chuyển đổi dữ liệu.  Khi bạn làm việc trong một công ty lớn thì dữ liệu được lưu trữ ở nhiều nơi khác nhau như được chứa trong: Oracle, SQL Server, DB2, Microsoft Access,… và bạn chắc chắn sẽ có nhu cầu di chuyển dữ liệu giữa các server này. Ngoài ra, bạn còn muốn định dạng dữ liệu trước khi lưu vào database. Chắc chắn Integration Services sẽ giúp bạn giải quyết được công việc này dễ dàng.
* Analysis Services: Đây là một dịch vụ phân tích dữ liệu rất hay của Microsoft. Dữ liệu khi được lưu trữ vào trong database mà bạn không thể lấy được những thông tin bổ ích thì coi như không có ý nghĩa gì. Chính vì thế, công cụ này ra đời giúp bạn trong việc phân tích dữ liệu một cách hiệu quả và dễ dàng bằng cách dùng kỹ thuật khai thác dữ liệu – datamining và khái niệm hình khối nhiều chiều – multi dimendion cubes.
* Notification Services: Dịch vụ thông báo này là nền tảng cho sự phát triển và triển khai các ứng dụng soạn và gửi thông báo. Ngoài ra, dịch vụ này còn có chức năng gửi thông báo theo dịch thời đến hàng ngàn người dăng ký sử dụng trên nhiều loại thiết bị khác nhau.
* Reporting  Services: là một công cụ tạo, quản lý và triển khai báo cáo bao gồm: server và client. Ngoài ra, nó còn là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.
* Full Text Search Service: là một thành phần đặc biệt trong việc truy vấn và đánh chỉ mục dữ liệu văn bản không cấu trúc được lưu trữ trong các cơ sở dữ liệu SQL Server.
* Service Broker: là một môi trường lập trình cho việc tạo ra các ứng dụng trong việc nhảy qua các Instance.
  + 1. **Cách truy cập CSDL**
       1. **Tạo đối tượng Connection**

Chúng ta sử dụng code sau:

using System.Data.SqlClient;

string chuoi = "Server=(local);Database=AdventureWorks;Integrated Security=SSPI";

SqlConnection ket\_noi = new SqlConnection(chuoi);

* + - 1. **Tạo đối tượng Command**

Chúng ta sử dụng code sau

using System.Data.SqlClient;

string lenh = "select count(\*) from production.product";

SqlCommand bo\_lenh = new SqlCommand(lenh, ket\_noi);

* 1. **Phân tích thiết kế hệ thống hướng đối tượng với UML**
     1. **Khái niềm về phân tích thiết kế hệ thống hướng đối tượng**

Trong kỹ nghệ phần mềm để sản xuất được một sản phẩm phần mềm người ta chia quá trình phát triển sản phẩm ra nhiều giai đoạn như thu thập và phân tích yêu cầu, phân tích và thiết kế hệ thống, phát triển (coding), kiểm thử, triển khai và bảo trì. Trong đó, giai đoạn phân tích, thiết kế bao giờ cũng là giai đoạn khó khăn và phức tạp nhất. Giai đoạn này giúp chúng ta hiểu rõ yêu cầu đặt ra, xác định giải pháp, mô tả chi tiết giải pháp. Nó trả lời 2 câu hỏi What (phần mềm này làm cái gì?) và How (làm nó như thế nào?).

Để phân tích và thiết kế một phần mềm thì có nhiều cách làm, một trong những cách làm đó là xem hệ thống gồm những đối tượng sống trong đó và tương tác với nhau. Việc mô tả được tất cả các đối tượng và sự tương tác của chúng sẽ giúp chúng ta hiểu rõ hệ thống và cài đặt được nó. Phương thức này gọi là Phân tích thiết kế hướng đối tượng (OOAD)

* + 1. **Khái niệm về UML**

UML là ngôn ngữ mô hình hóa hợp nhất dùng để biểu diễn hệ thống. Nói một cách đơn giản là nó dùng để tạo ra các bản vẽ nhằm mô tả thiết kế hệ thống. Các bản vẽ này được sử dụng để các nhóm thiết kế trao đổi với nhau cũng như dùng để thi công hệ thống (phát triển), thuyết phục khách hàng, các nhà đầu tư v.v… (Giống như trong xây dựng người ta dùng các bản vẽ thiết kế để hướng dẫn và kiểm soát thi công, bán hàng căn hộ v.v..)

* + 1. **Tại sao lại là OOAD vsf UML**

OOAD cần các bản vẽ để mô tả hệ thống được thiết kế, còn UML là ngôn ngữ mô tả các bản vẽ nên cần nội dung thể hiện.  Do vậy, chúng ta phân tích và thiết kế theo hướng đối tượng và sử dụng UML để biểu diễn các thiết kế đó nên chúng thường đi đôi với nhau.

# CHƯƠNG II: XÂY DỰNG GAME DÒ MÌN

# 1 Phân Tích Và Giải Thuật

### Phân tích

Luật lệ và cách chơi của game (Game mô phỏng trò chơi dò mìn Minesweeper)

* Người chơi khởi đầu với một bảng ô vuông trống thể hiện "bãi mìn".
* Click chuột vào một ô vuông trong bảng. Nếu không may trúng phải ô có  
  mìn (điều này ít xảy ra hơn) thì người chơi trò chơi kết thúc. Trường hợp  
  thường xảy ra hơn là ô đó không có mìn và một vùng các ô sẽ được mở ra  
  cùng với những con số. Số trên một ô là chỉ số ô có mìn trong cả thảy 8 ô  
  nằm lân cận với ô đó.
* Nếu chắc chắn một ô có mìn, người chơi đánh dấu vào ô đó bằng hình lá cờ  
  (click chuột phải).

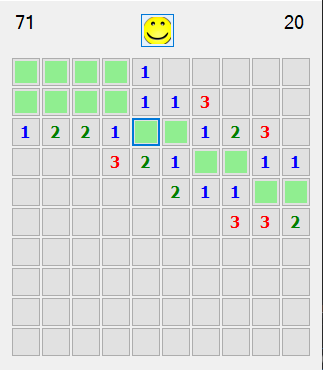
- Trò chơi kết thúc với phần thắng dành cho người chơi nếu tìm được tất cả  
các ô có mìn và mở được tất cả các ô không có mìn.

### Giải thuật

Thuật toán loang (thuật toán vết dầu loang) là một thuật toán dùng khá nhiều trong tin học, ứng dụng trong các bài toán thực tế như các bài toán tìm đường đi, game dò mìn, game line98…

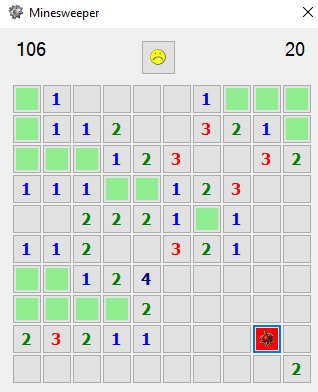
Gọi là thuật toán loang vì nguyên lí của thuật toán này rất giống với hiện tượng loang của chất lỏng. Khi ta đổ nước xuống một về mặt phẳng khô, nước có thể loang ra những khu vực xung quanh. Tương tự, thuật toán loang (trong một ma trận) sử dụng kĩ thuật “xét các ô liền kề”, rồi “loang” dần ra từ một vị trí ban đầu.

**Trong trò chơi dò mìn:** Ta sẽ thấy được sự tương tự “loang” chất lỏng trong trò dò mìn. Khi bạn mở một ô không chứa mìn sẽ kéo theo việc mở các không chứa mìn xung quanh nó. Từ ô bạn đã mở, việc mở này sẽ “loang” rộng ra các ô không chứa mìn xung quanh.



## Thiết Kế Ứng Dụng

### Thiết Kế Giao Diện



* Sử dụng panel chứa button làm các ô để làm giao diện chính để thao tác
* Textbox dùng để thể hiện số ô đã được mở và số mìn chưa được đánh dấu
* Button hình smile để tạo lại trò chơi khi click vào

### Thiết Kế Thuật Toán

private void LoadData()

{

lbsomin.Text = somin.ToString();

int i = 0, j = 0;

int width = 30;

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

{

for (j = 0; j < 10; j++)

{

btnmin[i, j] = new Button();

btnmin[i, j].Width = width;

btnmin[i, j].Height = width;

btnmin[i, j].Location = new Point(i \* width, j \* width);

btnmin[i, j].Text = "";

btnmin[i, j].Click += new EventHandler(bammin);

paldomin.Controls.Add(btnmin[i, j]);

a[i, j] = 0;

}

}

int dem = 0;

while (dem < somin)

{

Random rnd = new Random();

i = rnd.Next(10);

j = rnd.Next(10);

if (a[i, j] == 0)

{

a[i, j] = 1;

dem++;

}

}

}

private void bammin(object sender, EventArgs e)

{

if (control == 0)

{

int width = 30;

int x = 0, y = 0;

x = ((Button)sender).Location.X;

y = ((Button)sender).Location.Y;

int i = 0, j = 0;

i = x / width;

j = y / width;

mo\_o(i, j);

}

if(control == 1)

{

MessageBox.Show("You Win", "", MessageBoxButtons.OK);

}

else

if(control == -1)

{

MessageBox.Show("You Lose", "", MessageBoxButtons.OK);

}

}

private void mo\_o(int i, int j)

{

if (a[i, j] == 0)

{

if (btnmin[i,j].Text == "")

{

so\_o\_mo++;

if (sominxungquanh(i, j) == 0)

{

btnmin[i, j].Text = " ";

btnmin[i, j].BackColor = Color.LightGreen;

mo\_o\_lan\_can(i, j);

}

else

{

switch (sominxungquanh(i,j))

{

case 1:

btnmin[i, j].ForeColor = Color.Blue;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 2:

btnmin[i, j].ForeColor = Color.Green;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 3:

btnmin[i, j].ForeColor = Color.Red;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 4:

btnmin[i, j].ForeColor = Color.Navy;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 5:

btnmin[i, j].ForeColor = Color.Maroon;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 6:

btnmin[i, j].ForeColor = Color.Teal;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 7:

btnmin[i, j].ForeColor = Color.Black;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

case 8:

btnmin[i, j].ForeColor = Color.Gray;

btnmin[i, j].Text = sominxungquanh(i, j).ToString();

break;

}

}

lbsoo.Text = so\_o\_mo.ToString();

}

}

else

{

btnmin[i, j].Image = Image.FromFile("./Image/mine.ico");

btnplay.Image = Image.FromFile("./Image/smiley3.ico");

btnmin[i, j].BackColor = Color.Red;

control = -1;

}

}

private void mo\_o\_lan\_can(int i, int j)

{

for (int u = i - 1; u <= i + 1; u++)

{

for (int v = j - 1; v <= j + 1; v++)

{

if (u < 0 || u >= 10 || v < 0 || v >= 10)

{

continue;

}

if (u == i && v == j)

{

continue;

}

if (btnmin[u,v].Text == "" && a[u,v] == 0)

{

so\_o\_mo++;

mo\_o(u, v);

}

}

}

}

private void danhdau(object sender, MouseEventArgs e)

{

}

private int sominxungquanh(int i, int j)

{

int minxungquanh = 0;

for (int u = i - 1; u <= i + 1; u++)

{

for (int v = j - 1; v <= j + 1; v++)

{

if (u < 0 || u >= 10 || v < 0 || v >= 10)

{

continue;

}

if (u == i && v == j)

{

continue;

}

minxungquanh += a[u, v];

}

}

return minxungquanh;

}

private void btnplay\_MouseClick(object sender, MouseEventArgs e)

{

paldomin.Controls.Clear();

LoadData();

control = 0;

so\_o\_mo = 0;

somin = 20;

lbsoo.Text = "00";

lbsomin.Text = somin.ToString();

btnplay.Image = Image.FromFile("./Image/smiley1.ico");

}

* Đã làm được
  + Chương trình đã hoạt động
  + Khi mở 1 ô mà xung quanh không có bom thì sẽ tự động mở các ô lân cận
  + Khi mở hết các ô mà không mở phải bom thì hiện thông báo người chơi thắng
  + Nếu mở ô bom thì sẽ thông báo thua
  + Khi bấm vào icon smile trong giao diện game thì sẽ tạo lại game để chơi lại từ đầu
* Chưa làm được
  + Chương tính các phép toán phức tạp như” căn, luỹ thừa, sin, cos, tag”
* Hướng pháp triển
  + Thiết kế chương trình thực hiện những pháp tính phức tạp hơn
  + Thiết kế giao diện tính phương trình bậc 2-3 khi người dùng click chuột sẽ hiện ra giao diện.

## 2.4. Tài Liệu Tham Khảo

* Từ nhiều nguồn trên Internet
* Từ giáo viên hướng dẫn
* Từ anh chị khoá trên
* Từ các bạn học tốt trên lớp.