

Bài 8: ÉP KIỂU TRONG C#

Xem bài học trên website để ủng hộ Kteam: [Ép kiểu trong C#](#)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage [How Kteam](#) nhé!

Dẫn nhập

Ở bài [TOÁN TỬ TRONG C#](#) chúng ta đã một câu lệnh như sau: `SoNguyen = Int32.Parse(strSoNguyen)` kèm theo dòng chú thích **"Ép kiểu dữ liệu vừa nhập vào (dạng chuỗi) sang dạng số rồi gán giá trị vào biến SoNguyen"**.

Vậy **"ép kiểu"** là gì? Sử dụng chúng như thế nào? Bài học hôm nay sẽ giúp chúng ta trả lời những câu hỏi này – **Ép kiểu trong C#**.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- [CẤU TRÚC LỆNH CỦA C# viết trên nền Console Application.](#)
- [CẤU TRÚC NHẬP XUẤT của C# trên nền Console Application.](#)
- [BIẾN trong C#.](#)
- [KIỂU DỮ LIỆU trong C#.](#)
- [TOÁN TỬ trong C#.](#)

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Ép kiểu là gì? Tại sao phải ép kiểu?
- Có mấy loại ép kiểu? Cách sử dụng từng loại?
- Ví dụ chương trình sử dụng ép kiểu.

Ép kiểu là gì? Tại sao phải ép kiểu?

Ép kiểu là **biến đổi dữ liệu** thuộc kiểu dữ liệu này thành kiểu dữ liệu khác.

Tại sao phải ép kiểu?

- Để chuyển dữ liệu sang một kiểu dữ liệu mong muốn phục vụ cho việc thao tác xử lý.
- Đưa dữ liệu về định dạng mà mình mong muốn (ví dụ chuyển kiểu ngày tháng bên Mỹ sang dạng ngày tháng bên Việt Nam).

Có mấy loại ép kiểu? Cách sử dụng từng loại

Trong C#, ép kiểu có 4 loại:

- Chuyển đổi kiểu ngầm định (**implicit**).
- Chuyển đổi kiểu tường minh (**explicit**).
- Sử dụng phương thức, lớp hỗ trợ sẵn:
 - Dùng phương thức **Parse()**, **TryParse()**.
 - Dùng lớp hỗ trợ sẵn (**Convert**).
- Người dùng tự định nghĩa kiểu chuyển đổi.

Chuyển đổi kiểu ngầm định (implicit)

Việc chuyển đổi này được thực hiện bởi trình biên dịch và chúng ta không cần tác động gì.

Được thực hiện khi chuyển kiểu từ

- Kiểu dữ liệu nhỏ sang kiểu dữ liệu lớn hơn (xem lại bài [KIỂU DỮ LIỆU TRONG C#](#))
- Chuyển từ lớp con đến lớp cha (khái niệm lớp con lớp cha sẽ được trình bày trong bài [TÍNH KẾ THỪA TRONG C#](#)).

Ví dụ:

```
int intValue = 10;
```

```
long longValue = intValue; /* Chuyển kiểu ngầm định vì kiểu long có miền giá trị  
lớn hơn kiểu int nên có thể chuyển từ int sang long.*/  
float floatValue = 10.9f;  
double doubleValue = floatValue; /* Tương tự vì kiểu double có miền giá trị lớn  
hơn kiểu float nên có thể chuyển từ float sang double.*/
```

Chuyển đổi kiểu tường minh (explicit)

Chuyển đổi kiểu tường minh là việc chuyển kiểu một cách rõ ràng và dùng từ khóa chỉ định chứ không dùng phương thức.

Một số đặc điểm của chuyển đổi kiểu tường minh:

- Thường được dùng để chuyển đổi giữa **các kiểu dữ liệu có tính chất tương tự nhau** (thường thì với số).
- Hỗ trợ trong việc boxing và unboxing đối tượng (sẽ được trình bày trong bài [KIỂU DỮ LIỆU OBJECT TRONG C#](#)).
- Cú pháp ngắn gọn do không sử dụng phương thức.
- Chỉ khi chúng ta biết rõ biến đó thuộc kiểu dữ liệu nào mới thực hiện chuyển đổi. Ngược lại có thể lỗi khi chạy chương trình.

Cú pháp:

(<kiểu dữ liệu> <biến cần ép kiểu>

- <kiểu dữ liệu> là kiểu dữ liệu mà mình muốn chuyển sang.
- <biến cần ép kiểu> là biến chứa dữ liệu cần ép kiểu.
- Phải có cặp dấu ngoặc tròn ().

Ý nghĩa:

Ép kiểu của <biến cần ép kiểu> về <kiểu dữ liệu> nếu thành công sẽ trả ra giá trị kết quả ngược lại sẽ báo lỗi chương trình. Đặc biệt đối với số:

- Ta có thực hiện ép **kiểu dữ liệu lớn hơn** về **kiểu dữ liệu nhỏ hơn** mà không báo lỗi.

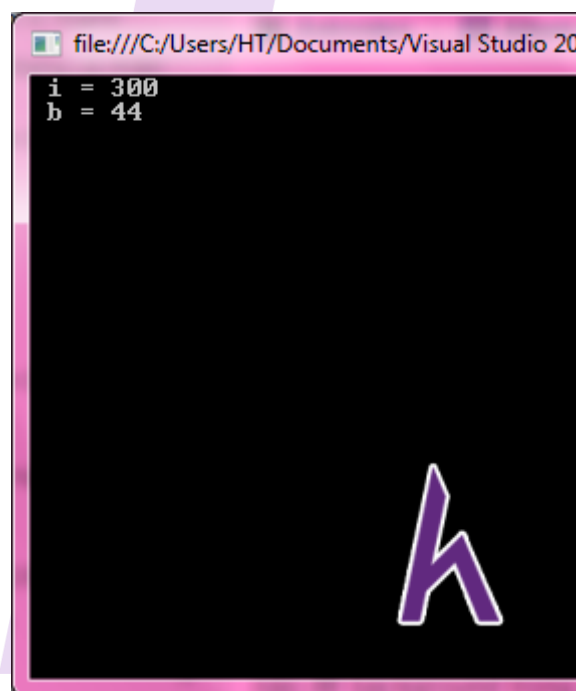
- Nếu dữ liệu cần ép kiểu **vượt quá miền giá trị** của kiểu dữ liệu muốn ép kiểu về thì chương trình sẽ **tự cắt bit** cho phù hợp với khả năng chứa kiểu dữ liệu đó (cắt từ bên trái qua).

Ví dụ:

```
int i = 300; // 300 có mã nhị phân là 100101100
byte b = (byte)i;
/* do kiểu byte có giới hạn đến giá trị 255 thôi nên không thể chứa số 300 được
mà kiểu byte có kích thước là 1 bytes tương đương 8 bit. Như vậy ta cần cắt mã
nhị phân của số 300 về còn 8 bit là được. Mã nhị phân 300 là 100101100 cắt từ trái
qua 1 bit ta được 00101100 (đủ 8 bit) tương đương với số 44. Cuối cùng biến b sẽ
mang giá trị là 44.*/
```

```
Console.WriteLine(" i = " + i);
Console.WriteLine(" b = " + b);
```

Kết quả khi chạy chương trình trên:



Việc ép kiểu thế này làm mình nhớ lại một vấn đề nho nhỏ sau: Ở bài [TOÁN TỬ TRONG C#](#) ta có nói là toán tử `"/"` là thực hiện chia lấy phần nguyên nếu cả 2 toán hạng đều là số nguyên. Vậy nếu chúng ta muốn chia ra kết quả chính xác luôn thì phải làm sao?

- Ý tưởng đơn giản là ta ép kiểu 1 trong 2 toán hạng đó về dạng số thực là xong.
- Cách khác, ta có thể làm đơn giản là nhân 1 trong 2 toán hạng với số 1.0 cũng sẽ cho ra kết quả như ý muốn.

Ví dụ:

```
double d = 2 / 3; // kết quả ra 0 vì 2 và 3 đều là số nguyên nên thực hiện 2 chia lấy phần nguyên với 3 được 0
double k = (double)2 / 3; // Ép kiểu số 2 từ kiểu nguyên sang kiểu số thực.
Như vậy kết quả phép chia sẽ ra số thực
double t = 1.0 * 2 / 3; // Thực hiện nhân 1.0 với 2 mục đích để biến số 2 (số nguyên) thành 2.0 (số thực)

Console.WriteLine(" d = {0} \n k = {1} \n t = {2}", d, k, t);
```

- Kết quả khi chạy chương trình thì biến k và biến t đều cho giá trị như nhau:



Sử dụng phương thức, lớp hỗ trợ sẵn

Phương thức Parse(), TryParse()

Parse()

Cú pháp:

```
<kiểu dữ liệu>.Parse(<dữ liệu cần chuyển đổi>);
```

- <kiểu dữ liệu> là kiểu dữ liệu cơ bản mình muốn chuyển đổi sang.
- <dữ liệu cần chuyển đổi> là dữ liệu thuộc kiểu chuỗi, có thể là biến kiểu chuỗi hoặc giá trị hằng kiểu chuỗi (giá trị hằng đã giải thích trong bài [HẰNG TRONG C#](#)).

Ý nghĩa:

- Chuyển đổi một chuỗi sang một kiểu dữ liệu cơ bản tương ứng.
- Phương thức trả về giá trị kết quả chuyển kiểu nếu chuyển kiểu thành công. Ngược lại sẽ báo lỗi chương trình.

Ví dụ:

```
string stringValue = "10";  
int intValue = int.Parse(stringValue); // Chuyển chuỗi stringValue sang kiểu  
int và lưu giá trị vào biến intValue - Kết quả intValue = 10  
double HowKteam = double.Parse("10.9"); // Chuyển chuỗi giá trị hằng  
"10.9" sang kiểu int và lưu giá trị vào biến HowKteam - Kết quả HowKteam = 10.9
```

TryParse()

Cú pháp:

```
<kiểu dữ liệu>.TryParse(<dữ liệu cần chuyển đổi>, out <biến chứa  
kết quả>);
```

- **<kiểu dữ liệu>** là kiểu dữ liệu cơ bản mình muốn chuyển đổi sang.
- **<dữ liệu cần chuyển đổi>** là dữ liệu thuộc kiểu chuỗi, có thể là biến kiểu chuỗi hoặc giá trị hằng kiểu chuỗi.
- **<biến chứa kết quả>** là biến mà bạn muốn lưu giá trị kết quả sau khi chuyển kiểu thành công. Từ khóa **out** là từ khóa bắt buộc phải có, ý nghĩa của từ khóa này sẽ được trình bày chi tiết trong bài [HÀM TRONG C#](#).

Ý nghĩa:

- Chuyển một chuỗi sang một kiểu dữ liệu cơ bản tương ứng.
- Phương thức sẽ trả về **true** nếu chuyển kiểu thành công và giá trị kết quả chuyển kiểu sẽ lưu vào **<biến chứa kết quả>**. Ngược lại sẽ trả về **false** và **<biến chứa kết quả>** sẽ mang giá trị 0.

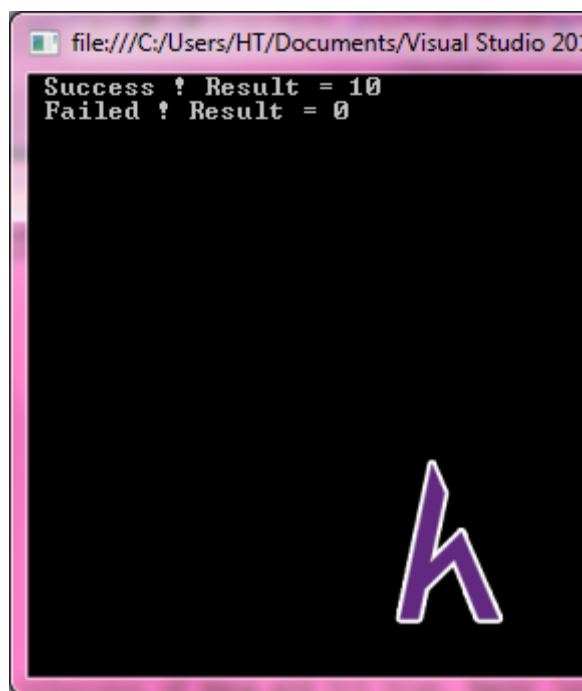
Ví dụ:

```
int Result; // Biến chứa giá trị kết quả khi ép kiểu thành công
bool isSuccess; // Biến kiểm tra việc ép kiểu có thành công hay không
string Data1 = "10", Data2 = "Kteam"; // Dữ liệu cần ép kiểu

isSuccess = int.TryParse(Data1, out Result); // Thử ép kiểu Data1 về int nếu thành
công thì Result sẽ chứa giá trị kết quả ép kiểu và isSuccess sẽ mang giá trị true.
Ngược lại Result sẽ mang giá trị 0 và isSuccess mang giá trị false
Console.Write(isSuccess == true ? " Success !" : " Failed !"); // Sử dụng toán tử 3
ngôi để in ra màn hình việc ép kiểu đã thành công hay thất bại.
Console.WriteLine(" Result = " + Result); // In giá trị Result ra màn hình

isSuccess = int.TryParse(Data2, out Result); // Tương tự như trên nhưng thao tác
với Data2
Console.Write(isSuccess == true ? " Success !" : " Failed !"); // Tương tự như trên
Console.WriteLine(" Result = " + Result); // Tương tự như trên
```

Kết quả khi chạy đoạn ví dụ trên là



Vì Data1 có thể ép kiểu về `int` nên kết quả thành công và in giá trị ra. Còn Data2 không thể ép kiểu về kiểu `int` nên kết quả không thành công và in ra giá trị 0.

Lưu ý khi sử dụng `Parse()` và `TryParse()`:

- Tham số truyền vào phải là một chuỗi.
- Cả 2 phương thức được gọi thông qua tên kiểu dữ liệu.
- `Parse()` trả về giá trị kết quả ép kiểu nếu ép kiểu không thành công thì sẽ báo lỗi. Còn `TryParse()` trả về xem ép kiểu có thành công hay không, giá trị kết quả ép kiểu sẽ nằm trong `<biến chứa kết quả>`.
- Ngoài `TryParse()` ra thì vẫn có một cách ép kiểu không báo lỗi chương trình. Đó là sử dụng toán tử `as`:
 - Trong bài [TOÁN TỬ TRONG C#](#) chúng ta có giới thiệu toán tử `as` dùng để "Ép kiểu mà không gây ra lỗi. Nếu ép kiểu không thành công sẽ trả về `null`".
 - Chỉ áp dụng cho việc chuyển kiểu giữa các kiểu tham chiếu tương thích (thường là các kiểu có quan hệ kế thừa (sẽ được trình bày ở bài [TÍNH KẾ THỪA TRONG C#](#))) hoặc các kiểu **nullable** (là các kiểu có thể chứa giá trị `null`).
 - Qua bài [KIỂU DỮ LIỆU OBJECT TRONG C#](#) mình sẽ ví dụ rõ hơn.

Lớp hỗ trợ sẵn (Convert)

Convert là lớp tiện ích được C# hỗ trợ sẵn cung cấp cho chúng ta nhiều phương thức chuyển đổi kiểu dữ liệu.

Các đặc điểm của các phương thức trong lớp **Convert**:

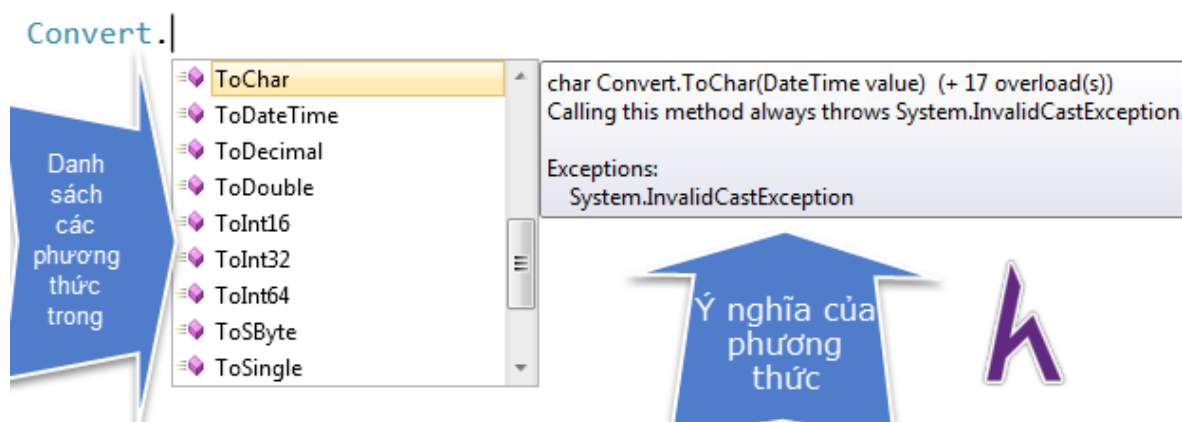
- Tham số truyền vào của các phương thức **không nhất thiết là chuỗi** (có thể là `int`, `bool`, ...).
- Nếu tham số truyền vào là `null` thì các phương thức sẽ **trả về giá trị mặc định** của kiểu dữ liệu.
- Các trường hợp tham số truyền vào sai định dạng hoặc vượt quá giới hạn thì chương trình sẽ báo lỗi như phương thức **Parse()**.

Một số phương thức chuyển kiểu mà **Convert** có:

Tên phương thức	Ý nghĩa
ToBoolean	Chuyển đổi một kiểu thành một bool (true, false) nếu có thể
ToByte	Chuyển đổi một kiểu thành một byte
ToChar	Chuyển đổi một kiểu thành một char nếu có thể
ToDateTime	Chuyển đổi một kiểu thành các cấu trúc date-time
ToDecimal	Chuyển đổi một kiểu thành một kiểu thập phân
ToDouble	Chuyển đổi một kiểu thành kiểu double
Chuyển đổi một kiểu thành kiểu int	
ToString	Chuyển đổi một kiểu thành kiểu string

- Các phương thức chuyển kiểu trong lớp **Convert** đều có thể nhận các kiểu dữ liệu cơ bản (`int`, `bool`, `byte`, ...) làm tham số truyền vào.
- Ở bảng trên mình chỉ liệt kê những phương thức hay dùng. Để tìm hiểu thêm các phương thức chuyển kiểu khác các bạn có thể sử dụng tính năng gợi ý của Visual Studio:
 - Đầu tiên các bạn gõ **"Convert."** (lưu ý dấu chấm liền sau **Convert**)

- Visual Studio sẽ hiển thị ra tất cả các phương thức có trong lớp **Convert**. Giờ bạn chỉ việc lựa chọn phương thức muốn sử dụng là được.



Người dùng tự định nghĩa kiểu chuyển đổi

Khi chúng ta tạo ra một kiểu dữ liệu mới chúng ta cũng cần định nghĩa các chuyển đổi kiểu dữ liệu từ kiểu cơ bản sang kiểu tự định nghĩa và ngược lại.

Nhưng trong phạm vi kiến thức hiện tại thì mình chỉ có thể giới thiệu cho các bạn biết là mình có thể tự định nghĩa kiểu chuyển đổi còn việc thực hiện như thế nào sẽ được trình bày trong bài [CLASS TRONG C#](#).

Ví dụ chương trình sử dụng ép kiểu

Ở các phần trên mình đều có trình bày ví dụ minh họa và giải thích chi tiết rồi nên các bạn xem lại.

Ví dụ: viết chương trình nhập vào 2 số a và b sau đó in ra tổng, hiệu, tích, thương của 2 số đó

Gợi ý/Hướng giải quyết bài toán ví dụ:

- Đầu tiên ta yêu cầu người dùng nhập vào 2 số a và b (lúc này giá trị nhập vào đang ở kiểu chuỗi).
- Ép kiểu giá trị vừa nhập vào kiểu chuỗi. Nếu thất bại sẽ thông báo lỗi và kết thúc chương trình, ngược lại thì thực hiện tiếp.
- Tính tổng, hiệu, tích, thương thì 2 số đó và in kết quả ra màn hình.

Chương trình minh họa:

```
int A, B; // Biến chứa giá trị 2 số vừa nhập vào (kiểu số)
int Tong, Hieu, Tich; // Biến chứa kết quả tổng, hiệu, tích
double Thuong; // Vì phép chia có thể cho ra số thập phân nên dùng biến
kiểu double để chứa nó.
string strA, strB; // Biến chứa giá trị 2 số nhập vào từ bàn phím (kiểu chuỗi)

Console.WriteLine("*****");
Console.WriteLine("
*");
Console.WriteLine("
*  Chương trình tính tong, hieu, tich, thuong  *");
Console.WriteLine("
*");
Console.WriteLine("*****");

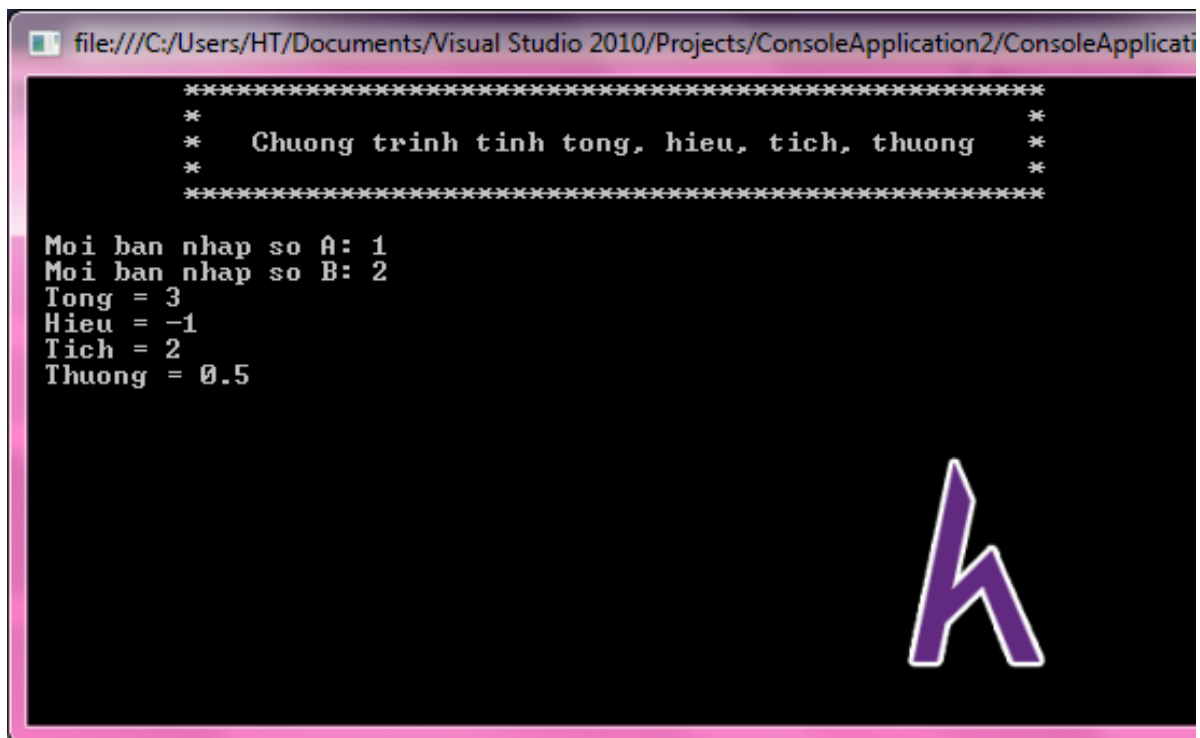
Console.Write("\n Moi ban nhap so A: ");
strA = Console.ReadLine(); // Nhận giá trị nhập vào từ bàn phím cho số A
Console.Write(" Moi ban nhap so B: ");
strB = Console.ReadLine(); // Nhận giá trị nhập vào từ bàn phím cho số B

A = int.Parse(strA); // Ép kiểu giá trị nhập vào từ kiểu chuỗi sang kiểu số
nguyên, sử dụng phương thức Parse()
B = int.Parse(strB); // Tương tự

Tong = A + B;
Hieu = A - B;
Tich = A * B;
Thuong = (double)A / B; // Ép kiểu số A về số thập phân để phép chia cho
ra số thập phân

Console.WriteLine(" Tong = " + Tong);
Console.WriteLine(" Hieu = " + Hieu);
Console.WriteLine(" Tich = " + Tich);
Console.WriteLine(" Thuong = " + Thuong);
```

Kết quả khi chạy chương trình trên:

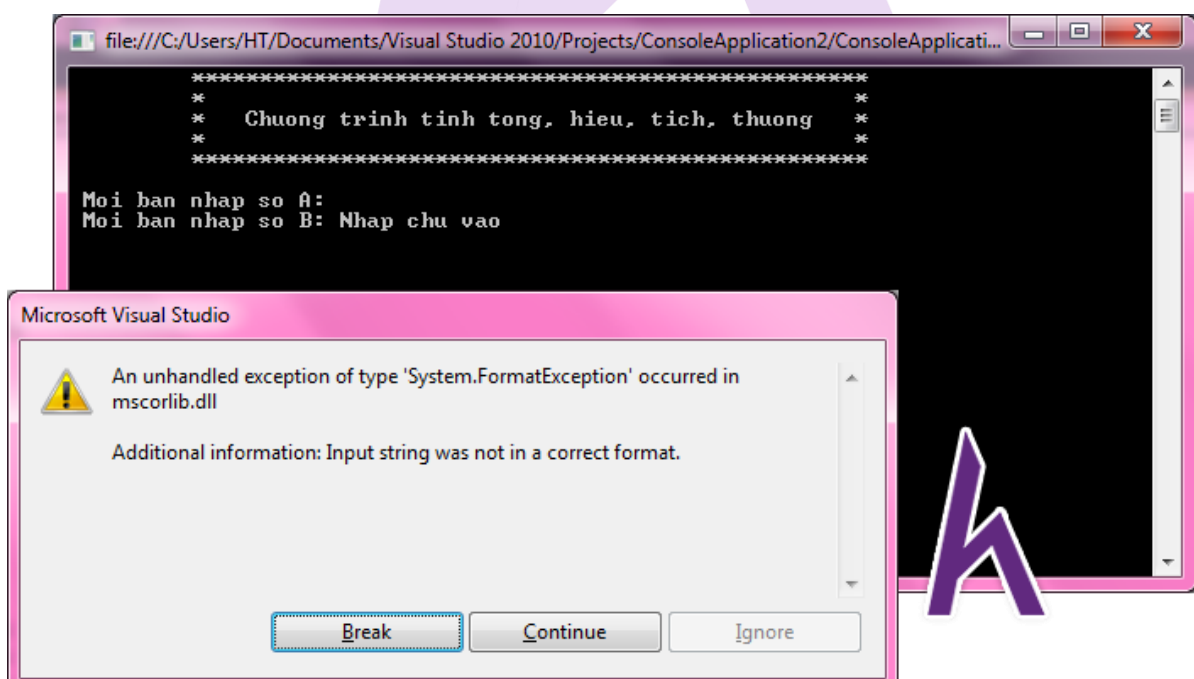


```
file:///C:/Users/HT/Documents/Visual Studio 2010/Projects/ConsoleApplication2/ConsoleApplicati

*****
*                                     *
*   Chương trình tính tong, hieu, tich, thuong   *
*                                     *
*****

Moi ban nhap so A: 1
Moi ban nhap so B: 2
Tong = 3
Hieu = -1
Tich = 2
Thuong = 0.5
```

- Các bạn để ý 2 trường hợp ép kiểu trong ví dụ trên, đó là 2 trường hợp thường gặp nhất của ép kiểu được sử dụng trong hầu hết các chương trình tính toán từ đơn giản đến phức tạp.
- Một vấn đề xảy ra trong ví dụ trên là "nếu người dùng không nhập vào một số mà lại nhập chữ hoặc bỏ trống thì sao?"
 - Khi đó chương trình sẽ bị lỗi ngay:



- Lúc này ta nghĩ ngay đến phương thức **TryParse()** để giải quyết vấn đề trên. Nào hãy thử tự giải quyết và gửi lại bài giải ngay cho mình kiểm tra nhé. Chúc các bạn thành công!

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Khái niệm về ép kiểu và tại sao phải ép kiểu.
- Các loại ép kiểu và cách sử dụng.
- Ví dụ về sử dụng ép kiểu.

Bài học sau chúng ta sẽ cùng tìm hiểu một khái niệm tiếp theo đó là [CẤU TRÚC IF - ELSE TRONG C#](#)

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên **"Luyện tập – Thử thách – Không ngại khó"**.