

# Welcome To Sherry Forex Group

Tư vấn đầu tư, hoạt động dịch vụ tài chính. Cung cấp tin tức, dấu hiệu, thời điểm vào ra thị trường tài chính, EA Robot trade tự động. Dịch vụ chu đáo, tận tình hướng dẫn, luôn luôn sát cánh cùng bạn trong thị trường tài chính. Thành công của bạn là mục đích của tôi. LH: 0908.396.209 YM: sherryfxgroup

[Trang chủ](#)


## Bài đăng Phổ biến

[biến static và hàm static trong java](#)

[JSON là gì? JSON làm việc như thế nào](#)

[Thủ thuật tối ưu, tăng tốc Google Chrome, Firefox!](#)

[Cài đặt Eclipse 4.2 Juno trên Ubuntu 12.04](#)

[DANH SÁCH LIÊN KẾT \(LINKED LIST\) - NODES](#)

[Qui đổi giá vàng thế giới sang vàng trong nước](#)

[Vùng Vào Lệnh? Điểm Vào Lệnh?](#)

[Hệ thống giao dịch Leo System](#)

[Overload - override Java \(Nạp chồng - ghi đè\)](#)

[Hướng dẫn cho các bạn sử dụng FPT không vào được 360 Plus](#)

## Blog Archive

- ▼ 2014 (52)
  - [September](#) (1)
  - [August](#) (1)

Thursday, June 12, 2014

## static\_cast C++

### Casting

Casting là cách chuyển đổi kiểu dữ liệu theo mong muốn của lập trình viên. Trong chương trình C chuẩn, Cast được thực hiện thông qua toán tử (), với tên của loại dữ liệu muốn cast ở bên trong. Ví dụ

```
int nValue1 = 10;
```

```
int nValue2 = 4 ;
```

```
float fValue = (float)nValue1 / nValue2 ;
```

Trong chương trình ở trên, chúng ta sử dụng float cast để báo cho trình biên dịch biết để thăng cấp nValue1 thành kiểu float. Bởi vì nValue1 có kiểu float nên nValue2 sau đó cũng được chuyển đổi thành kiểu float. Và khi đó, phép chia là phép chia kiểu số thực(float) thay vì số nguyên (integer)

C++ cũng sẽ cho bạn sử dụng kiểu C cast này trong chương trình C++ của bạn :

```
int nValue1 = 10;
```

```
int nValue2 = 4 ;
```

```
float fValue = (float)nValue1 / nValue2 ;
```

C++ giới thiệu một toán tử casting mới gọi là static\_cast. Static cast làm việc tương tự như kiểu C cast.

```
int nValue1 = 10;
```

```
int nValue2 = 4 ;
```

```
float fValue = static_cast(nValue1) / nValue2 ;
```

Bạn nên tránh sử dụng casting trong tất cả mọi trường hợp nếu có thể, bởi vì bất kỳ ở đâu sử dụng cast, ở đó có thể là nguyên nhân gây ra nhiều vấn đề cho chương trình của bạn. Nhưng đôi khi bạn bắt buộc phải sử dụng nếu không thể tránh được. Trong trường hợp này, trong C++ bạn nên sử dụng static\_cast thay vì S-style cast :d

- July (2)
- ▼ June (20)
  - Cách may mũ vải cho bé điệu  
đầu mùa thu đông và nhữ...
  - hash\_map in C++
  - Abstract Class vs Interface in  
C++
  - Lớp Trừu tượng & Giao diện  
(Abstract Classes & Int...
  - HashMap in Java
  - Cloneable interface - Clone()  
Object Java
  - SuppressWarnings và một vài  
value
  - Serializable trong Java
  - Comparable: Java
  - Kế thừa trong C++ (Inheritance)
  - static\_cast C++
  - DANH SÁCH LIÊN KẾT  
(LINKED LIST) - NODES
  - How to create Linked list using  
C/C++ - NODES
  - STD::VECTOR -  
STD::VECTOR::PUSH\_BAC  
K
  - NAMESPACE LÀ GÌ????
  - How To Install And Run  
Android OS On Windows
  - Overload - override Java (Nạp  
chồng - ghi đè)
  - Tìm Hiểu Thread Trong JAVA
  - Kiểu dữ liệu enum trong C++
  - Tìm hiểu Thread trong JAVA
- May (9)
- April (16)
- March (3)
- 2012 (1)
- 2011 (20)
- 2010 (4)

Total Pageviews

6,602

Để có thể điều khiển việc chuyển đổi kiểu giữa các lớp, chuẩn ANSI-C++ đã định nghĩa bốn toán tử chuyển đổi kiểu mới: **reinterpret\_cast**, **static\_cast**, **dynamic\_cast** và **const\_cast**. Tất cả chúng đều có cùng dạng thức khi sử dụng:

```
reinterpret_cast <new_type> (expression)
dynamic_cast <new_type> (expression)
static_cast <new_type> (expression)
const_cast <new_type> (expression)
```

Trong đó **new\_type** kiểu mà **expression** phải được chuyển đổi thành. Để tạo ra sự tương tự dễ hiểu với các toán tử chuyển đổi truyền thống các biểu thức này có nghĩa là:

```
(new_type) expression
new_type (expression)
```

#### reinterpret\_cast

**reinterpret\_cast** chuyển đổi một con trỏ sang bất kì kiểu con trỏ nào khác. Nó cũng cho phép chuyển đổi từ con trỏ sang dạng số nguyên và ngược lại.

Toán tử này có thể chuyển đổi con trỏ giữa các lớp không có quan hệ với nhau. Kết quả của toán tử này là một bản copy giá trị của con trỏ này sang con trỏ kia. Dữ liệu được trỏ đến không hề được kiểm tra hay chuyển đổi.

Trong trường hợp chuyển đổi giữa con trỏ và số nguyên, cách chuyển nội dung của nó phụ thuộc vào hệ thống

```
class A {};
class B {};
A * a = new A;
B * b = reinterpret_cast(a);
```

**reinterpret\_cast** đối xử với tất cả các con trỏ giống như các toán tử chuyển đổi truyền thống.

**static\_cast** cho phép thực hiện bất kì phép chuyển đổi nào

**static\_cast** allows to perform any casting that can be implicitly performed as well as also the inverse cast (even if this is not allowed implicitly).

Applied to pointers to classes, that is to say that it allows to cast a pointer of a derived class to its base class (this is a valid conversion that can be implicitly performed) and can also perform the inverse: cast a base class to its derived class.

In this last case the base class that is being casted is not checked to determine whether this is a complete class of the destination type or not.

```
class Base {};
class Derived: public Base {};
Base * a = new Base;
Derived * b = static_cast(a);
```

**static\_cast**, aside from manipulating pointers to classes, can also be used to perform conversions explicitly defined in classes, as well as to perform standard conversions between fundamental types:

```
double d=3.14159265;
int i = static_cast(d);
```

#### dynamic\_cast

**dynamic\_cast** is exclusively used with pointers and references to objects. It allows any type-casting that can be implicitly performed as well as the inverse one when used with polymorphic classes, however, unlike **static\_cast**, **dynamic\_cast** checks, in this last case, if the operation is valid. That is to say, it checks if the casting is going to return a valid complete object of the requested type.

Checking is performed during run-time execution. If the pointer being casted is not a pointer to a valid complete object of the requested type, the value returned is a **NULL** pointer.

```
class Base { virtual dummy(){}; };
class Derived : public Base { };
```

```
Base* b1 = new Derived;
Base* b2 = new Base;
Derived* d1 = dynamic_cast(b1); // succeeds
Derived* d2 = dynamic_cast(b2); // fails: returns
NULL
```

If the type-casting is performed to a reference type and this casting is not possible an *exception* of type **bad\_cast** is thrown:

```
class Base { virtual dummy(){}; };
class Derived : public Base { };
```

```
Base* b1 = new Derived;
Base* b2 = new Base;
Derived d1 = dynamic_cast(b1); // succeeds
Derived d2 = dynamic_cast(b2); // fails: exception
thrown
```

#### const\_cast

This type of casting manipulates the *const* attribute of the passed object, either to be set or removed:

```
class C {};
const C * a = new C;
C * b = const_cast (a);
```

Neither of the other three new **cast** operators can modify the constness of an object.

**typeid**

ANSI-C++ also defines a new operator called **typeid** that allows to check the type of an expression:

**typeid (expression)**

this operator returns a reference to a constant object of type **type\_info** that is defined in standard header file. This returned value can be compared with another using operators **==** and **!=** or can serve to obtain a string of characters representing the data type or class name by using its **name()** method.

*static\_cast: chuyển kiểu dữ liệu bình thường như int -> char,...*

*dynamic\_cast: chuyển đổi kiểu con trỏ (hoặc kiểu tham chiếu) giữa các lớp đa hình trong đa kế thừa.*

*reinterpret\_cast: chuyển đổi giữa 2 kiểu dữ liệu ko có mối liên hệ, vd như là int -> pointer,...*

*const\_cast: bỏ const ra khỏi dữ liệu được chuyển đổi.*

Posted by **SherryFxGroup** at 12:02 PM



Recommend this on Google

## 2 comments:



**Quang Nguyen van** September 27, 2014 at 9:02 AM

Sàn giao dịch bất động sản Đất Vàng

Web: [www.datvanggroup.com/](http://www.datvanggroup.com/)

Click vào Keywords: [Căn hộ Vinhomes Central Park](#)

Click vào Keywords: [Vinhomes Central Park Sài Gòn](#)

[Reply](#)



**NGHĨA PHAN TRỌNG** September 27, 2014 at 10:40 PM

Chà học cái này mà ra đầu tư thì okê y như rằng !

Em nhớ hum bữa cũng làm cái kia mà ra thanh công không thể  
Kỷ niệm đó hay ghê ta

Click vào đây nhé : [Bán Căn Hộ Sunview Town quận Thủ Đức tại TPHCM](#) hoặc click vào  
đây nhé bạn [bán căn hộ sunview town quận thủ đức tại tphcm](#)

[Reply](#)

Enter your comment...



Comment as:

Google Accour ▼

[Publish](#)

[Preview](#)



[Newer Post](#)

[Home](#)

[Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)