Bài: Ép kiểu ngầm định trong C++ (Implicit type conversion in C++)

Xem bài học trên website để ủng hộ Kteam: Ép kiểu ngầm định trong C++ (Implicit type conversion in 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 học trước, bạn đã nắm được BIẾN TĨNH TRONG C++ (Static variables).

Hôm nay, mình sẽ chia sẽ tổng quan về ép kiểu dữ liệu và kỹ thuật **Ép kiểu ngầm định trong C++ (Implicit type conversion)**.

Nội dung

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

- BIẾN TRONG C++ (Variables in C++)
- CÁC KIỂU DỮ LIỆU CƠ BẢN trong C++

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Tổng quan về ép kiểu dữ liệu
- Ép kiểu ngầm định trong C++ (Implicit type conversion)

Tổng quan về ép kiểu dữ liệu

Trong bài BIẾN TRONG C++ (Variables in C++), bạn đã biết được **biến là tên của một vùng trong bộ nhớ RAM**, và giá trị của biến được **lưu trong bộ nhớ** dưới dạng các **bit nhị phân**.

> Các biến có **kiểu dữ liệu khác nhau**, mặc dù chúng giữ những **giá trị giống nhau**, nhưng khi lưu vào bộ nhớ, chúng là những **dãy nhị phân hoàn toàn khác nhau**.

Ví dụ: giá trị 1 là một số nguyên, 1.0 là một số chấm động. Dù trong toán học nó là như nhau, nhưng khi lưu vào bộ nhớ, nó được biểu diễn bằng những dãy bit nhị phân hoàn toàn khác nhau.

Quan sát trường hợp bên dưới:

C++:

```
int nValue = 1;    // initialize integer variable with integer 1
float fValue = 1;  // initialize floating point variable with integer 1
```

Ở dòng lệnh đầu tiên, trình biên dịch (compiler) chỉ cần sao chép các bit nhị phân của 1 vào vùng nhớ của biến nValue. Nhưng ở dòng lệnh thứ 2, trình biên dịch (compiler) phải thực hiện việc chuyển đổi giá trị số nguyên 1 sang số chấm động 1.0f, sau đó giá trị 1.0f mới được gán cho biến fValue.

> Quá trình chuyển đổi một giá trị **từ kiểu dữ liệu này sang kiểu dữ liệu khác** được gọi là **chuyển đổi kiểu dữ liệu (ép kiểu)**.

Ép kiểu dữ liệu trong C++ xảy ra trong những trường hợp sau (và một số trường hợp khác):

- **Gán hoặc khởi tạo** một biến với giá trị của một kiểu dữ liệu khác

C++:

```
double dValue(1);  // initialize double variable with integer value 1
dValue = 1;        // assign double variable with integer value 1
```

- Gọi một hàm với **đối số (argument)** và **tham số (parameter)** có **kiểu dữ liệu khác nhau**

C++:

```
void doSomething(double dValue)
{
}

doSomething(1); // pass integer value 1 to a function expecting a double parameter
```

- Kiểu dữ liệu của **giá trị trả về** khác với **kiểu dữ liệu trả về** của hàm đó

C++:

```
float doSomething()
{
    return 1.0; // Pass double value 1.0 to a function that returns a float
}
```

- Sử dụng toán tử 2 ngôi với **2 toán hạn khác kiểu dữ liệu**

C++:

```
double dValue = 4.0 / 3;  // division with a double and an integer
int nValue = 1.5 * 3;     // multiply with a double and an integer
```

Trong C++, **ép kiểu dữ liệu có hai loại**:

- **Ép kiểu ngầm định (Implicit type conversion)**: trình biên dịch (compiler) sẽ tự **động chuyển đổi** từ một kiểu dữ liệu này sang kiểu dữ liệu khác **(kiểu dữ liệu cơ sở)**.
- **Ép kiểu tường minh (Explicit type conversion)**: **lập trình viên sử dụng toán tử ép kiểu (casting operator)** để thực hiện việc chuyển đổi.

Trong bài học này, mình sẽ giới thiệu về **Ép kiểu ngầm định (Implicit type conversion)**.

Ép kiểu ngầm định trong C++ (Implicit type conversion)

Ép kiểu ngầm định (Implicit type conversion) là quá trình chuyển đổi giữa các kiểu dữ liệu cơ sở một cách ngầm định, **trình biên dịch (compiler) sẽ tự động chuyển đổi** từ một kiểu dữ liệu này sang kiểu dữ liệu khác. **Lập trình viên không can thiệp** trực tiếp vào quá trình chuyển đổi.

> Chuyển đổi giá trị **từ một kiểu sang một kiểu dữ liệu tương tự lớn hơn** thường **an toàn**, và **không mất dữ liệu**.

Ví dụ:

C++:

```
long l(1);         // 1 is a integer
double d(0.1f);    // 0.1 is a float
```

> Chuyển đổi giá trị t**ừ một kiểu sang một kiểu dữ liệu tương tự nhỏ hơn**, hoặc **giữa các kiểu dữ liệu khác nhau** thường **không an toàn**, nó có thể dẫn đến **mất mát dữ liệu** sau khi chuyển đổi.

Một số trường hợp lưu ý

Trường hợp 1: chuyển đổi giá trị sang một kiểu dữ liệu có miền giá trị nhỏ hơn sẽ gây ra kết quả không mong muốn

C++:

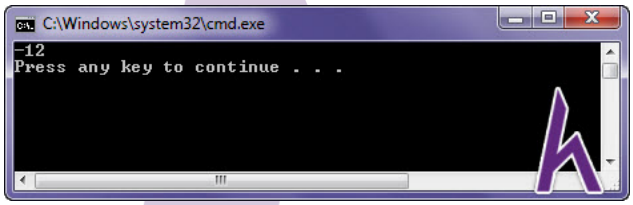
```
#include <iostream>
using namespace std;

int main()
{
	int n = 500;
	char c = n;

	cout << static_cast<int>(c) << endl;

	return 0;
}
```

Outputs:



Trong chương trình trên, giá trị **số nguyên (int)** được chuyển đổi sang **kiểu ký tự (char)**. Miền giá trị của char từ **-128 đến 127**, nhỏ hơn kiểu int nên vấn đề **tràn số** đã xảy ra. Kết quả là biến c có một giá trị không như mong muốn.

Trường hợp 2: đối với kiểu **số chấm động (floating point)**, **chuyển đổi giá trị về kiểu dữ liệu nhỏ hơn** có thể gây **mất độ chính xác**.

C++:

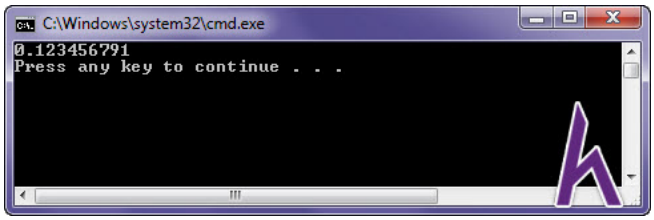
```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
	// double value 0.123456789 has 9 significant digits, but float can only support about 7
	float f = 0.123456789;

	cout << setprecision(9) << f << endl; // std::setprecision defined in iomanip header

	return 0;
}
```

Outputs:



Trong chương trình trên, 0.123456789 là một **giá trị kiểu double có 9 chữ số**. Khi gán giá trị 0.123456789 vào biến f, **biến f** chỉ có thể giữ **độ chính xác đến 7 chữ số**, làm kết quả sau khi chuyển đổi bị **mất đi độ chính xác**.

Trường hợp 3: Chuyển đổi một giá trị từ số chấm động sang số nguyên sẽ làm mất đi phần thập thân.

C++:

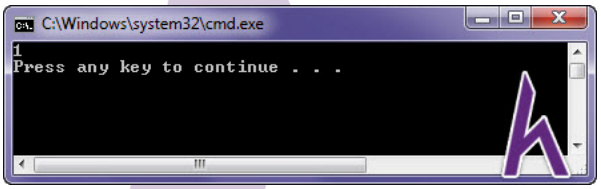
```
#include <iostream>
using namespace std;

int main()
{
	double d = 1.5;
	int n = d;

	cout << n << endl;

	return 0;
}
```

Outputs:



Trong chương trình trên, biến số nguyên n chỉ nhận giá trị 1, phần thập phân 0.5 đã bị mất đi.

Ép kiểu ngầm định trong một biểu thức

Khi tính toán biểu thức, trình biên dịch sẽ **tách mỗi biểu thức thành những biểu thức con nhỏ hơn**. Các toán tử số học đòi hỏi các toán hạng phải **cùng kiểu dữ liệu**.

Để đảm bảo điều này, trình biên dịch sử dụng các quy tắc sau đây:

- Nếu toán hạng là một số nguyên có **miền giá trị nhỏ hơn kiểu int** (bool, char, unsigned char, signed char, unsigned short, signed short), toán hạng đó sẽ **tự động chuyển về kiểu int hoặc unsigned int.**
- Nếu vẫn không phù hợp, trình biên dịch (compiler) sẽ chuyển các toán hạng về **cùng kiểu dữ liệu với toán hạng có độ ưu tiên cao nhất**.

Độ ưu tiên kiểu dữ liệu của các toán hạng như sau:

1. long double (highest)
2. double
3. float
4. unsigned long long
5. long long
6. unsigned long
7. long
8. unsigned int
9. int (lowest)

Ví dụ 1:

C++:

```
#include <iostream>
using namespace std;

int main()
{
	bool b(2);
	char c(6);
	short s(9);

	// show the type of b + c + s
	cout << typeid(b + c + s).name() << " " << b + c + s << endl;

	return 0;
}
```

Outputs:

Trong chương trình trên, 3 biến b, c, s có kiểu dữ liệu nhỏ hơn kiểu int, nên kết quả biểu thức được chuyển đổi ngầm định về kiểu int.

Ví dụ 2:

C++:

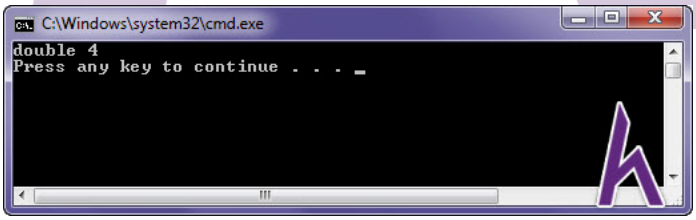
```
#include <iostream>
using namespace std;

int main()
{
	short s(1);
	int n(2);
	double d(1.0);

	// show the type of (s + d) * n
	cout << typeid((s + d) * n).name() << " " << (s + d) * n << endl;

	return 0;
}
```

Outputs:



Trong chương trình trên, kiểu double có độ ưu tiên cao nhất, nên cả biểu thức sẽ chuyển đổi ngầm định về kiểu double.

Kết luận

Qua bài học này, bạn đã nắm được tổng quan về ép kiểu dữ liệu và kỹ thuật Ép kiểu ngầm định trong C++ (Implicit type conversion).

Ở bài tiếp theo, bạn sẽ được học về ÉP KIỂU TƯỜNG MINH TRONG C++ (Explicit type conversion).

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**".