Conversiones de Tipos C++

Antonio Espín Herranz

Conversiones de tipo en C++

- Soporta las conversiones implícitas de C:
  - Asignación de un valor:
    - long a; int b = 10; a = b; // Se convierte a long.
  - Operación Aritmética:
    - float a = 10.5, c;
    - int b = 5; c = a + b; // A float.
  - Argumento a una función:
    - int a = 2; double b = logaritmo(a);
    - Siendo la función: float logaritmo(float x); // Se convierte a float.
  - Cuando se retorna un valor desde una función:
    - double b = logaritmo(a); // El valor devuelto pasa a double.
- Explícitas: Mediante un casting. (otroTipo) variable / expresión.

Conversiones de tipo en C++

- En C++ también se puede expresar así:

  nombre-de-tipo(expresión);

```
// 1ª forma:                    // 2ª forma
int a; double n = 8;
a = (int)sqrt(n+2);             a = int(sqrt(n+2));
```

- La notación funcional no se puede utilizar con tipos que no sean un tipo simple.

Operadores de cambio de tipo

- Todos convierten la expresión **v** al tipo **T**.
  - **static_cast<T>(v):**
    - Convertir tipos relacionados, punteros o entre tipo real y entero.
    - int a = static_cast<int>(sqrt(n+2));
  - **reinterpret_cast<T>(v):**
    - Tipos no relacionados, peligrosas, permite de double * a int * que la anterior no la permite.
    - Se usa para hacer cambios de tipo a nivel de bits, es decir, el valor de **v** se interpreta como si fuese un objeto del tipo “**T**". Los modificadores const y volatile no se modifican, permanecen igual que en el valor original de "expresión". Este tipo de conversión es peligrosa, desde el punto de vista de la compatibilidad, hay que usarla con cuidado.
  - **dynamic_cast<T>(v):**
    - Realizar conversiones en t. de ejecución. No se puede ejecutar entre punteros, devuelve 0.
  - **const_cast<T>(v):**
    - Se utiliza para eliminar la acción ejercida por const o volatile

Ejemplo: static_cast

```
int main() {
    Derivada *pDer = new Derivada(10, 23.3);
    Base *pBas;
    pDer->Mostrar();
    // Ambas formas válidas:
      //1) Derivada pBas = static_cast<Base *> (pDer);
      // 2) pBas = pDer; // Igualmente legal, pero implícito
    pBas->Mostrar(); // Base
    delete pDer;
    return 0;
}
```

Ejemplo: reinterpret_cast

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

int main() {
    int x = 0x12dc34f2;
    int *pix = &x;
    unsigned char *pcx;
    pcx = reinterpret_cast<unsigned char *> (pix);
    cout << hex << x << " = " <<
        static_cast<unsigned int> (pcx[0]) << ", " <<
        static_cast<unsigned int> (pcx[1]) << ", " <<
        static_cast<unsigned int> (pcx[2]) << ", " <<
        static_cast<unsigned int> (pcx[3]) << endl;
    return 0;
}
```

Ejemplo: dynamic_cast

Referenciar la clase Base mediante la Derivada:

class Base { // Clase base **virtual** …};

class Derivada : **public** Base { // Clase derivada ... };

// Modalidad de puntero:

Derivada *p = **dynamic_cast**<Derivada *> (&Base);

// Modalidad de referencia:

Derivada &refd = **dynamic_cast**<Derivada &> (Base);

Al revés no sería necesario:

```
Base *b = new Derivada();
```

Ejemplo: const_cast

```
#include <iostream>
using namespace std;
int main() {
    const int x = 10;
    int *x_var;
    x_var = const_cast<int*> (&x); // Válido
    //x_var = &x; // Ilegal, el compilador da error
    *x_var = 14;

    cout << *x_var << ", " << x << endl;
    return 0;
}
```