**Casting Operators**

The word cast is used in the sense of “casting into a mold.” The compiler will automatically change one type of data into another if it makes sense.

**In C**, the result of the cast operation is not an lvalue

**In C++**, the cast result belongs to one of the following value categories:

* If type is an lvalue reference type (C++11 or an rvalue reference to a function type), the cast result is an lvalue.
* In all other cases, the cast result is a (C++11 prvalue) rvalue.
* If type is an rvalue reference to an object type, the cast result is an xvalue.

In C++ you can also use the following objects in cast expressions:

1. Function-style casts
2. type(expression)

# C++ conversion operators

cast\_Operator<target\_Type>(variable)

#include <iostream>

using namespace std;

int main() {

float num = 98.76;

int x1 = (int) num; // C - style

int x2 = int(num);

int x3 = static\_cast<int>(num);

cout << "x1 = " << x1 << endl;

cout << "x2 = " << x2 << endl;

cout << "x3 = " << x3 << endl;

return 0;

}

Output:

x1 = 98

x2 = 98

x3 = 98

For C++, the operand of a cast expression can have class type.

* If the operand has class type, it can be cast to any type for which the class has a user-defined conversion function.
* Casts can invoke a constructor, if the target type is a class, or they can invoke a conversion function, if the source type is a class.
* They can be ambiguous if both conditions hold.

# C++ explicit casts

Standard C++ includes an explicit cast syntax that can be used to completely replace the old C-style casts (of course, C-style casts cannot be outlawed without breaking code, but compiler writers could easily flag old-style casts for you).

|  |  |
| --- | --- |
| **static\_cast** | For “well-behaved” and “reasonably well-behaved” casts, including things you might now do without a cast (such as an automatic type conversion). |
| **const\_cast** | To cast away const and/or volatile. |
| **reinterpret\_cast** | To cast to a completely different meaning. The key is that you’ll need to cast back to the original type to use it safely. The type you cast to is typically used only for bit twiddling or some other mysterious purpose. This is the most dangerous of all the casts. |
| **dynamic\_cast** | For type-safe downcasting |

# END