**Type Conversion in C++**

A type cast is basically a conversion from one type to another. There are two types of type conversion:

1. **Implicit Type Conversion** Also known as ‘automatic type conversion’.
   * Done by the compiler on its own, without any external trigger from the user.
   * Generally takes place when in an expression more than one data type is present. In such conditions type conversion (type promotion) takes place to avoid loss of data.
   * All the data types of the variables are upgraded to the data type of the variable with largest data type.

bool -> char -> short int -> int ->

unsigned int -> long -> unsigned ->

long long -> float -> double -> long double

* It is possible for implicit conversions to lose information, signs can be lost (when signed is implicitly converted to unsigned), and overflow can occur (when long long is implicitly converted to float).

**Example of Type Implicit Conversion:**

C++

// An example of implicit conversion

#include <iostream>

using namespace std;

int main()

{

int x = 10; // integer x

char y = 'a'; // character c

// y implicitly converted to int. ASCII

// value of 'a' is 97

x = x + y;// x is implicitly converted to float

float z = x + 1.0;

cout << "x = " << x << endl

<< "y = " << y << endl

<< "z = " << z << endl;

return 0;

}

**Output**

x = 107

y = a

z = 108

**2.Explicit Type Conversion**: This process is also called type casting and it is user-defined. Here the user can typecast the result to make it of a particular data type.

In C++, it can be done in two ways:

**Conversion using Cast operator:** A Cast operator is a **unary operator** which forces one data type to be converted into another data type.  
C++ supports four types of casting:

1. Static Cast
2. Dynamic Cast
3. Const Cast
4. Reinterpret Cast

**Example:**

C++

#include <iostream>

using namespace std;

int main()

{

float f = 3.5;

// using cast operator

int b = static\_cast<int>(f);

cout << b;

}

**Output**

3

**Advantages of Type Conversion:**

* This is done to take advantage of certain features of type hierarchies or type representations.
* It helps to compute expressions containing variables of different data types.

Marked as Read

Report An Issue