Type Casting

➢**Type Casting** is also known as Type Conversion. For example, suppose the given data is an integer type, and we want to convert it into float type. So, we need to manually cast int data to the float type, and this type of casting is called the Type Casting in C++.

➢Typecasting can be done in two ways: automatically by the compiler and manually by the programmer or user. Type Casting is also known as Type Conversion.

➢Example 1

```
int num = 5;
float x;
x = float(num);
x = 5.0
```

Example 2

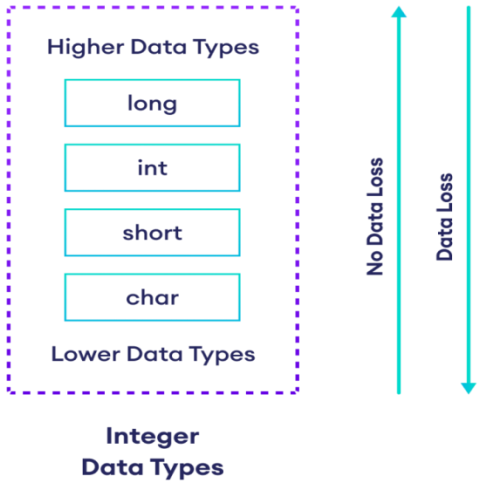
```
float num = 5.25;
int x;
x = int(num);
Output: 5
```

Types

- Type Casting is divided into two types: Implicit conversion or Implicit Type Casting and Explicit Type Conversion or Explicit Type Casting.

1. Implicit Type Casting or Implicit Type Conversion

- It is known as the automatic type casting.
- It automatically converted from one data type to another without any external intervention such as programmer or user. It means the compiler automatically converts one data type to another.
- All data type is automatically upgraded to the largest type without losing any information.
- It can only apply in a program if both variables are compatible with each other.
- **char** - short **int** -> **int** -> unsigned **int** -> **long int** -> **float** -> **double** -> **long double**, etc.



Program to use the implicit type casting in C++

```
#include <iostream.h>
#include<conio.h>
int main ()
{
   short x = 200;
   int y;
   y = x;
   cout << " Implicit Type Casting " << endl;
   cout << " The value of x: " << x << endl;
   cout << " The value of y: " << y << endl;

   int num = 20;
   char ch = 'a';
   int res = 20 + 'a';
   cout << " Type casting char to int data type ('a' to 20): " << res << endl;

   float val = num + 'A';
   cout << " Type casting from int data to float type: " << val << endl;
   return 0;
}
```

Output:
Implicit Type Casting
The value of x: 200
The value of y: 200
Type casting char to int data type ('a' to 20): 117
Type casting from int data to float type: 85

- In the above program, we declared a short data type variable x is 200 and an integer variable y. After that, we assign x value to the y, and then the compiler automatically converts short data value x to the y, which returns y is 200.

- In the next expressions, we declared an int type variable num is 20, and the character type variable ch is 'a', which is equivalent to an integer value of 97. And then, we add these two variables to perform the implicit conversion, which returns the result of the expression is 117.

Explicit Type Casting or Explicit Type Conversion

- It is also known as the manual type casting in a program.
- It is manually cast by the programmer or user to change from one data type to another type in a program. It means a user can easily cast one data to another according to the requirement in a program.
- It does not require checking the compatibility of the variables.
- In this casting, we can upgrade or downgrade the data type of one variable to another in a program.
- It uses the cast () operator to change the type of a variable.

Syntax of the explicit type casting

(type) expression;

- **type:** It represents the user-defined data that converts the given expression.
- **expression:** It represents the constant value, variable, or an expression whose data type is converted.

For example, we have a floating pointing number is 4.534, and to convert an integer value, the statement as:

```
int num;
num = (int) 4.534; // cast into int data type
cout << num;
```

When the above statements are executed, the floating-point value will be cast into an integer data type using the cast () operator. And the float value is assigned to an integer num that truncates the decimal portion and displays only 4 as the integer value.

Program to demonstrate the use of the explicit type casting in C++

```
#include <iostream>
using namespace std;
int main ()
{
   // declaration of the variables
   int a, b;
   float res;
   a = 21;
   b = 5;
   cout << " Implicit Type Casting: " << endl;
   cout << " Result: " << a / b << endl; // it loses some information

   cout << " \n Explicit Type Casting: " << endl;
   // use cast () operator to convert int data to float
   res = (float) 21 / 5;
   cout << " The value of float variable (res): " << res << endl;

   return 0;
}
```

Output:

- Implicit Type Casting:

Result: 4

- Explicit Type Casting:

The value of float variable (res): 4.2