

# Expression and Implicit conversions

## C++ Type Conversion

In this tutorial, we will learn about the basics of C++ type conversion with the help of examples.

C++ allows us to convert data of one type to that of another. This is known as type conversion.

There are two types of type conversion in C++.

- 1. Implicit Conversion
- 2. Explicit Conversion (also known as Type Casting)

## Implicit Type Conversion

The type conversion that is done automatically done by the compiler is known as implicit type conversion. This type of conversion is also known as automatic conversion.

Let us look at two examples of implicit type conversion.

### Example 1: Conversion From int to double

```
// Working of implicit type-conversion

#include <iostream>
using namespace std;

int main() {
    // assigning an int value to num_int
    int num_int = 9;

    // declaring a double type variable
    double num_double;

    // implicit conversion
    // assigning int value to a double variable
    num_double = num_int;

    cout << "num_int = " << num_int << endl;
    cout << "num_double = " << num_double << endl;

    return 0;
}
```

#### Output

```
num_int = 9
num_double = 9
```

In the program, we have assigned an `int` data to a `double` variable.

```
num_double = num_int;
```

Here, the `int` value is automatically converted to `double` by the compiler before it is assigned to the `num_double` variable. This is an example of implicit type conversion.

### Example 2: Automatic Conversion from double to int

```
//Working of Implicit type-conversion

#include <iostream>
using namespace std;

int main() {

    int num_int;
    double num_double = 9.99;

    // implicit conversion
    // assigning a double value to an int variable
    num_int = num_double;

    cout << "num_int = " << num_int << endl;
    cout << "num_double = " << num_double << endl;

    return 0;
}
```

#### Output

```
num_int = 9
num_double = 9.99
```

In the program, we have assigned a `double` data to an `int` variable.

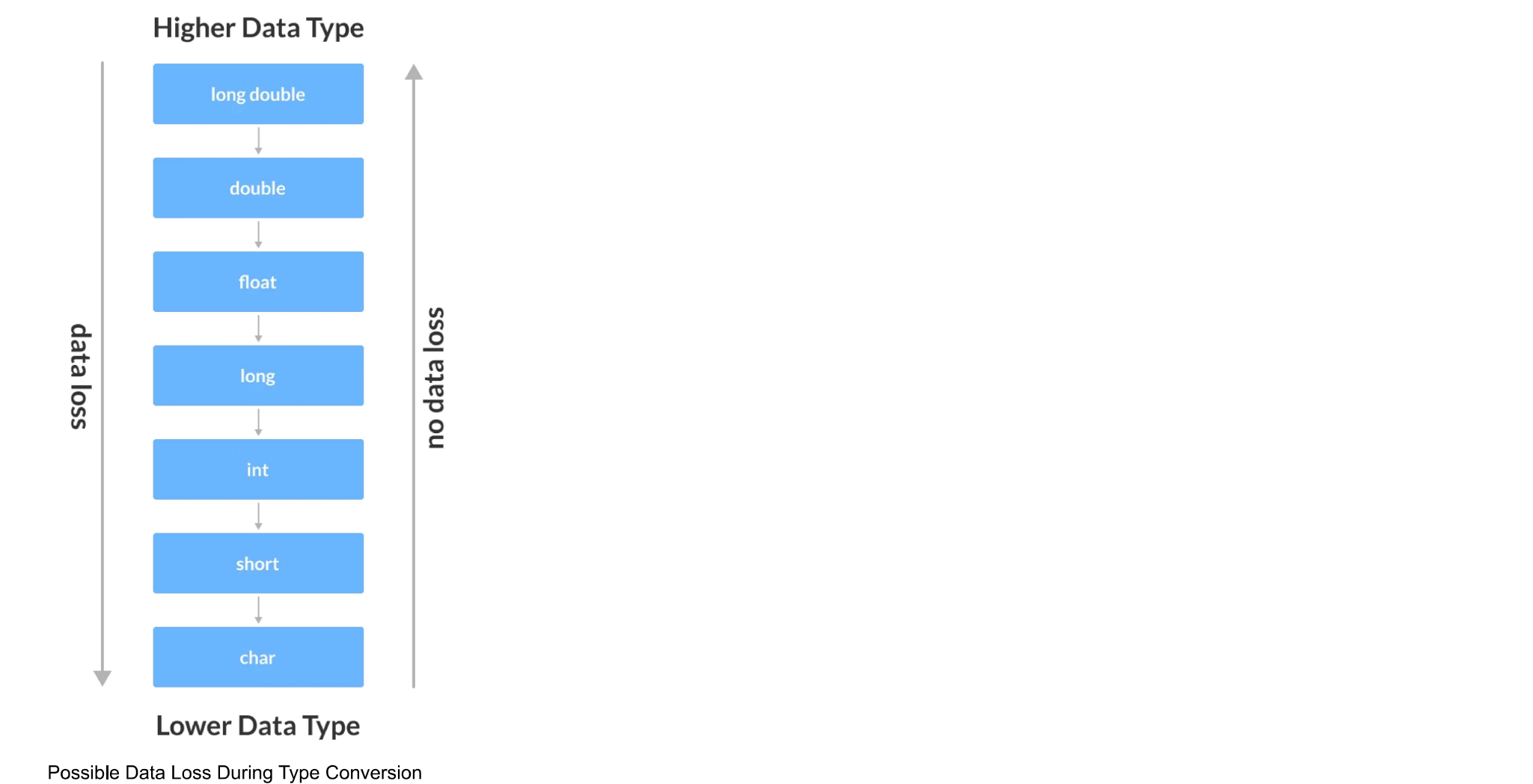
```
num_double = num_int;
```

Here, the `double` value is automatically converted to `int` by the compiler before it is assigned to the `num_int` variable. This is also an example of implicit type conversion.

**Note:** Since `int` cannot have a decimal part, the digits after the decimal point are truncated in the above example.

## Data Loss During Conversion (Narrowing Conversion)

As we have seen from the above example, conversion from one data type to another is prone to data loss. This happens when data of a larger type is converted to data of a smaller type.



## C++ Explicit Conversion

When the user manually changes data from one type to another, this is known as **explicit conversion**. This type of conversion is also known as **type casting**.

There are three major ways in which we can use explicit conversion in C++. They are:

- 1. C-style type casting (also known as **cast notation**)
- 2. Function notation (also known as **old C++ style type casting**)
- 3. Type conversion operators

### C-style Type Casting

As the name suggests, this type of casting is favored by the **C programming language**. It is also known as **cast notation**.

The syntax for this style is:

```
(data_type)expression;
```

For example,

```
// initializing int variable
int num_int = 26;

// declaring double variable
double num_double;

// converting from int to double
num_double = (double)num_int;
```

### Function-style Casting

We can also use the function like notation to cast data from one type to another.

The syntax for this style is:

```
data_type(expression);
```

For example,

```
// initializing int variable
int num_int = 26;

// declaring double variable
double num_double;

// converting from int to double
num_double = double(num_int);
```

---

## Example 3: Type Casting

```
#include <iostream>

using namespace std;

int main() {
    // initializing a double variable
    double num_double = 3.56;
    cout << "num_double = " << num_double << endl;

    // C-style conversion from double to int
    int num_int1 = (int)num_double;
    cout << "num_int1  = " << num_int1 << endl;

    // function-style conversion from double to int
    int num_int2 = int(num_double);
    cout << "num_int2  = " << num_int2 << endl;

    return 0;
}
```

### Output

```
num_double = 3.56
num_int1    = 3
num_int2    = 3
```

We used both the **C style type conversion** and the **function-style casting for type conversion** and displayed the results. Since they perform the same task, both give us the same output.

---

## Type Conversion Operators

Besides these two type castings, C++ also has four operators for type conversion. They are known as **type conversion operators**. They are:

- `static_cast`
- `dynamic_cast`
- `const_cast`
- `reinterpret_cast`