Type Conversion in C

In C, type conversion refers to the process of changing one data type into another. This conversion can only occur between compatible data types. This conversion can be of two types:

- **Implicit** (Done by the compiler)
- **Explicit** (Done by the programmer)

Implicit Conversion

Implicit conversion, also known as type promotion, occurs when the compiler automatically converts a smaller or lower data type to a larger or higher one during an operation. This ensures no loss of information (except when explicitly stated). But when converting larger data type to smaller, it can cause loss of information.

Example: When converting a float to an int, there is a loss of information during the process.

> float x = 10.5;
> int y = x; // y = 10
> bool z = y; // z = 1

When dealing with an expression that includes multiple types, the given rule of conversion is followed:

> bool -> char -> short int -> int -> unsigned int -> long -> unsigned -> long long -> float -> double -> long double

If int variable has a negative value, the negative value is stored in 2's complement. If type conversion happens in int to unsigned then 2's complement representation is considered as it is in an unsigned variable.

Examples of Implicit Conversion

The below example illustrates the implicit conversion by the compiler:

Example 1:

C

```c
#include <stdio.h>

int main()
```

```
{

    int x = 10;
    float y = 10.5;
    float z = x + y;
    printf("%f", z);
    return 0;
}
```

Output

```
20.500000
```

Explanation: In the expression "float z = x + y", integer x is promoted to float.

Example 2:

C

```c
#include <stdio.h>

int main()
{
    char a = 'B';
    double b = 10.5;
    double c = a + b;
    printf("%g", c);
    return 0;
}
```

Output

```
76.5
```

Explanation: The character **'B'** is converted to its ASCII value **(66)**, and the operation is performed in the double data type.

Explicit Conversion

Explicit conversion, or typecasting is done explicitly by the programmer according to the requirement.

Syntax:

> (type) expression;

Example:

In the below code, we are dividing two integers:

C

```c
#include <stdio.h>
int main()
{
    int x = 15, y = 2;
    double z = x/y;
    printf("%g", z);
    return 0;
}
```

Output

7

Explanation: In the above program, both x and y are integers. When x is divided by y, the calculation is performed as in integer division, and the expected result is 7.5 but we get 7 because the calculation happens integer then assigned to the variable z. So, we lose some data in this process.

To resolve this, we can convert x variable into double using explicit conversion:

C

```c
#include <stdio.h>
int main()
{
    int x = 15, y = 2;
    double z = (double) x/y;
    printf("%g", z);
    return 0;
}
```

Output

7.5