Integer Promotions in C

The C compiler promotes certain data types to a higher rank for the sake of achieving consistency in the arithmetic operations of integers.

In addition to the standard int data type, the C language lets you work with its subtypes such as char, short int, long int, etc. Each of these data types occupy a different amount of memory space. For example, the size of a standard int is 4 bytes, whereas a char type is 2 bytes of length. When an arithmetic operation involves integer data types of unequal length, the compiler employs the policy of integer promotion.

Integer Promotions

As a general principle, the **integer** types smaller than **int** are promoted when an operation is performed on them. If all values of the original type can be represented as an int, the value of the smaller type is converted to an **int**; otherwise, it is converted to an **unsigned int**.

One must understand the concept of **integer promotion** to write reliable C code, and avoid unexpected problems related to the size of data types and arithmetic operations on smaller integer types.

Example

In this example, the two variables **a** and **b** seem to be storing the same value, but they are not equal.

```
#include <stdio.h>

int main(){

   char a = 251;
   unsigned char b = a;

   printf("a = %c", a);
   printf("\nb = %c", b);

   if (a == b)
      printf("\n Same");
   else
      printf("\n Not Same");

   return 0;
}
```

Output

When you run this code, it will produce the following output −

```
a = 
b = 
Not Same
```

You get this output because "a" and "b" are treated as integers during comparison. "a" is a signed char converted to int as -5, while "b" is an unsigned char converted to int as 251.

Example: Mechanism of Integer Promotions

Let us try to understand the mechanism of integer promotions with this example −

```
#include <stdio.h>

int main(){

   char a = 'e', b = '2', c = 'M';
   char d = (a * b) / c;

   printf("d as int: %d as char: %c", d, d);
```

```
    return 0;
}
```

Output

Run the code and check its output −

```
d as int: 65 as char: A
```

When Integer Promotion is Applied?

In the arithmetic expression "(a * b) / c", the bracket is solved first. All the variables are of signed char type, which is of 2 byte length and can store integers between -128 to 127. Hence the multiplication goes beyond the range of char but the compiler doesn't report any error.

The C compiler applies **integer promotion** when it deals with arithmetic operations involving small types like char. Before the multiplication of these char types, the compiler changes them to int type. So, in this case, (a * b) gets converted to int, which can accommodate the result of multiplication, i.e., 1200.

Example

Integer promotions are applied as part of the usual arithmetic conversions to certain argument expressions; operands of the unary +, -, and ~ operators; and operands of the shift operators. Take a look at the following example −

```
#include <stdio.h>

int main(){

    char a = 10;
    int b = a >> 3;

    printf("b as int: %d as char: %c", b, b);

    return 0;
}
```

Output

When you run this code, it will produce the following output −

```
b as int: 1 as char:
```

In the above example, shifting the bit structure of "a" to the left by three bits still results in its value within the range of char (a << 3 results in 80).

Example

In this example, the rank of the char variable is prompted to int so that its left shift operation goes beyond the range of char type.

```c
#include <stdio.h>

int main(){

   char a = 50;
   int b = a << 2;

   printf ("b as int: %d as char: %c", b, b);

   return 0;
}
```

Output

Run the code and check its output −

```
b as int: 200 as char:
```

Integer Promotion Rules

Promotion rules help the C compiler in maintaining consistency and avoiding unexpected results. The fundamental principle behind the rules of promotion is to ensure that the expression's type is adjusted to accommodate the widest data type involved, preventing data loss or truncation.

Here is a summary of promotion rules as per C11 specifications −

- The integer types in C are char, short, int, long, long long and enum. Booleans are also treated as an integer type when it comes to type promotions.
- No two signed integer types shall have the same rank, even if they have the same representation.
- The rank of a signed integer type shall be greater than the rank of any signed integer type with less precision.
- The rank of long int > the rank of int > the rank of short int > the rank of signed char.
- The rank of char is equal to the rank of signed char and unsigned char.
- Whenever a small integer type is used in an expression, it is implicitly converted to int which is always signed.
- All small integer types, irrespective of sign, are implicitly converted to (signed) int when used in most expressions.

In short, we have the following integer promotion rules −

- **Byte and short values** − They are promoted to int.
- **If one operand is a long** − The entire expression is promoted to long.
- **If one operand is a float** − The entire expression is promoted to float.
- **If any of the operands is double** − The result is promoted to double.

Example

Here, the variables **x** and **y** are of char data type. When the division operation is performed on them, they automatically get promoted to int and the resultant value is stored in **z**.

```
#include <stdio.h>

int main(){

   char x = 68;
   char y = 34;

```

```
    printf("The value of x is: %d", x);
    printf("\nThe value of y is: %d", y);
```

```
    return 0;
}
```

Output

When you run this code, it will produce the following output −

```
The value of x is: 68
The value of y is: 34
The value of z: 2
```

Ethical Hacking Tutorial

Docker Tutorial

Kubernetes Tutorial

DSA Tutorial

Spring Boot Tutorial

SDLC Tutorial

Unix Tutorial

CERTIFICATIONS

Business Analytics Certification

Java & Spring Boot Advanced Certification

Data Science Advanced Certification

Cloud Computing And DevOps

Advanced Certification In Business Analytics

Artificial Intelligence And Machine Learning

DevOps Certification

Game Development Certification

Front-End Developer Certification

AWS Certification Training

Python Programming Certification

COMPILERS & EDITORS

Online Java Compiler

Online Python Compiler

Online Go Compiler

Online C Compiler

Online C++ Compiler

Online C# Compiler

Online PHP Compiler

Online MATLAB Compiler

Online Bash Terminal

Online SQL Compiler

Online Html Editor