Typecasting in C Language

By

Prof. Muhammad Iqbal Bhat

Department of Higher Education,

Government Degree College Beerwah

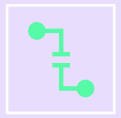
Topics

- What is Typecasting in C?
- Why do we need Typecasting
- Types of Typecasting

Typecasting in C Language:

In C language, a variable has a specific data type, such as integer, float, char, etc.

The data type of a variable determines the range of values it can hold and the operations that can be performed on it.



However, there are situations when we need to convert a variable of one data type to another data type in order to perform a specific operation. This is where typecasting comes into play.

Typecasting is the process of converting a variable of one data type to another data type.

For example, if we want to perform division on two integer variables but get a float result, we can use typecasting to convert one of the integer variables to float before performing the division operation.

Why do we need Typecasting?

In C, sometimes it is necessary to convert one data type into another to perform a particular operation.

Typecasting enables us to perform such operations by converting the data type of a variable.

For example, let's say we have two variables - one of type int and one of type float - and we want to perform division on them. Since the division operator (/) only works with variables of the same data type, we cannot perform division directly on these variables. Therefore, we need to use typecasting to convert one of the variables to float before performing the division operation.

```
int a = 10;
float b = 2.0;
float result = (float)a / b; // typecasting a to float before division
printf("Result: %f", result);
```

Types of Typecasting in C Language:

- Implicit typecasting
- Explicit typecasting

Implicit typecasting:

In C language, implicit typecasting, also known as type coercion, is the automatic conversion of one data type to another data type by the compiler. This is done when an expression contains operands of different data types, and one of the operands needs to be converted to the other data type for the operation to be performed.

The compiler follows a set of rules for implicit typecasting, known as type promotion, to determine which operand should be converted to which data type. The rules are as follows:

If either operand is of type 'long double', the other operand is converted to 'long double'.

Otherwise, if either operand is of type 'double', the other operand is converted to 'double'.

Otherwise, if either operand is of type 'float', the other operand is converted to 'float'.

Otherwise, the integer promotions are performed on both operands. This means that if either operand is of type 'long', the other operand is converted to 'long'. If both operands are of type 'int' or 'short', they are converted to 'int'.

Examples of Implicit typecasting:

```
int a = 10;
float b = 2.0;
float result = a / b; // implicit typecasting of 'a' to float
printf("Result: %f", result);
```

```
char c = 'a';
int result = c + 1; // implicit typecasting of 'c' to int
printf("Result: %d", result);
```

It's important to note that implicit typecasting can sometimes lead to unexpected results or loss of precision, especially when converting between data types of different sizes. It's recommended to use explicit typecasting when converting between data types to ensure clarity and accuracy in your code.

```
int a = 200;
char b = a; // implicit typecasting of 'a' to char
printf("b: %d", b);
```

Explicit Typecasting in C Language:

In C language, explicit typecasting is the manual conversion of one data type to another data type using the typecast operator.

The typecast operator is denoted by enclosing the target data type in parentheses before the value to be converted.

The syntax for explicit typecasting is as follows:

```
(target_data_type) value_to_be_converted;
```

Explicit typecasting is useful when we want to convert a variable of one data type to another data type explicitly. It can also help in preventing loss of data or precision when converting between data types.

Examples of explicit typecasting:

```
float a = 3.14159;
int b = (int) a; // explicit typecasting of 'a' to int
printf("b: %d", b);
```

```
int a = 65;
char b = (char) a; // explicit typecasting of 'a' to char
printf("b: %c", b);
```

Typecasting in Expressions

In C language, typecasting can also be performed within expressions to ensure that operands of different data types are treated in a consistent manner. This is known as typecasting in expressions.

When performing operations with operands of different data types, C language automatically performs implicit typecasting to ensure that the operands are of compatible data types. However, this can sometimes lead to unexpected results or loss of precision.

To avoid such issues, we can use explicit typecasting within expressions to ensure that the operands are treated in a consistent manner. The syntax for typecasting within expressions is the same as that for explicit typecasting. We simply enclose the operand to be converted in parentheses, followed by the target data type.

Examples of typecasting within expressions

```
int a = 10;
float b = 3.14159;
float c = (float) a / b; // typecasting of 'a' to
float
printf("c: %f", c);
```

```
int a = 10;
float b = 3.14159;
int c = a / (int) b; // typecasting of 'b' to int
printf("c: %d", c);
```

Valid and Invalid typecasting examples:

Type of Typecasting	Valid Examples	Reason
Implicit Typecasting	int a = 10; float b = 3.14; float c = a + b;	In this example, C automatically performs implicit typecasting of the int variable a to float before adding it to b.
Implicit Typecasting	int a = 10; float b = 3.14; int c = a + b;	In this example, C also performs implicit typecasting of the float variable b to int before adding it to a.
Implicit Typecasting	int a = 10; float b = 3.14; double c = a * b;	In this example, C automatically performs implicit typecasting of the int variable a to float before multiplying it with b. Then, the result of the multiplication is implicitly typecast to double.
Invalid Implicit Typecasting	char a = 'A'; int b = a + 10;	In this example, C automatically performs implicit typecasting of the char variable a to int before adding it to 10. However, this can lead to unexpected results as the ASCII value of 'A' is 65, which when added to 10 results in 75.

Valid and Invalid typecasting examples:

Type of Typecasting	**Example**	**Reason**
Explicit Typecasting	int a = 10; float b = (float) a / 3;	In this example, we explicitly typecast the int variable a to float before dividing it by 3.
Explicit Typecasting	float a = 3.14; int b = (int) a;	In this example, we explicitly typecast the float variable a to int before assigning it to b.
Explicit Typecasting	double a = 3.14; float b = (float) a;	In this example, we explicitly typecast the double variable a to float before assigning it to b.
Invalid Explicit Typecasting	int a = 100000; char b = (char) a;	In this example, we explicitly typecast the int variable a to char. However, this can lead to data loss as the maximum value that can be stored in a char variable is only 127. Therefore, the resulting value of b may not be accurate.
Invalid Explicit Typecasting	float a = 3.14; int b = (int) &a;	In this example, we explicitly typecast the address of a to int. However, this is an invalid typecasting as it does not make sense to typecast a memory address to an integer value.

Questions?