**C++ Ternary Operator**

In this tutorial, we will learn about the ternary operator in C++ with the help of examples.

In C++, the **ternary operator** (also known as the **conditional operator**) can be used to replace [if...else](https://www.programiz.com/cpp-programming/if-else) in certain scenarios.

**Ternary Operator in C++**

A ternary operator evaluates the test condition and executes a block of code based on the result of the condition.

Its syntax is

condition ? expression1 : expression2;

Here, condition is evaluated and

* if condition is true, expression1 is executed.
* And, if condition is false, expression2 is executed.

The ternary operator takes **3 operands** (condition, expression1 and expression2). Hence, the name **ternary operator**.

**Example : C++ Ternary Operator**

#include <iostream>

#include <string>

using namespace std;

int main() {

double marks;

// take input from users

cout << "Enter your marks: ";

cin >> marks;

// ternary operator checks if

// marks is greater than 40

string result = (marks >= 40) ? "passed" : "failed";

cout << "You " << result << " the exam.";

return 0;

}

**Output 1**

Enter your marks: 80

You passed the exam.

Suppose the user enters **80**. Then, the condition marks >= 40 evaluates to true. Hence, the first expression "passed" is assigned to result.

**Output 2**

Enter your marks: 39.5

You failed the exam.

Now, suppose the user enters **39.5**. Then, the condition marks >= 40 evaluates to false. Hence, the second expression "failed" is assigned to result.

**When to use a Ternary Operator?**

In C++, the ternary operator can be used to replace certain types of if...else statements.

For example, we can replace this code

#include <iostream>

using namespace std;

int main() {

// Create a variable

int number = -4;

if (number > 0)

cout << "Positive Number";

else

cout << "Negative Number!";

return 0;

}

with

#include <iostream>

#include <string>

using namespace std;

int main() {

int number = -4;

string result;

// Using ternary operator

result = (number > 0) ? "Positive Number!" : "Negative Number!";

cout << result << endl;

return 0;

}

**Output**

Negative Number!

Here, both programs give the same output. However, the use of the ternary operator makes our code more readable and clean.

**Note:** We should only use the ternary operator if the resulting statement is short.

**Nested Ternary Operators**

It is also possible to use one ternary operator inside another ternary operator. It is called the nested ternary operator in C++.

Here's a program to find whether a number is positive, negative, or zero using the nested ternary operator.

#include <iostream>

#include <string>

using namespace std;

int main() {

int number = 0;

string result;

// nested ternary operator to find whether

// number is positive, negative, or zero

result = (number == 0) ? "Zero" : ((number > 0) ? "Positive" : "Negative");

cout << "Number is " << result;

return 0;

}

**Output**

Number is Zero

In the above example, notice the use of ternary operators,

(number == 0) ? "Zero" : ((number > 0) ? "Positive" : "Negative");

Here,

* (number == 0) is the first test condition that checks if number is 0 or not. If it is, then it assigns the string value "Zero" to result.
* Else, the second test condition (number > 0) is evaluated if the first condition is false.

**Note**: It is not recommended to use nested ternary operators. This is because it makes our code more complex.