Conditional Expression

This lesson explains what conditional expressions are, how to use them and their basic syntax using an example

There are expressions of a special kind, the *conditional expressions*, these are **not** *statements*, but they are one sort of contraction of the `if-then` construct.

This kind of expression can help to produce highly readable assignment statements fitting onto *one* line of the source code.

Syntax

This is the syntax:

```
( condition ) ? expressionIfTrue : expressionIfFalse;
```

First the condition is evaluated and the side effects of this evaluation carry out their impact on the local environment.

- If the result is **true** then only the `expressionIfTrue` is evaluated (causing side effects) and this second result is the *value* of the whole *conditional expression*, and the `expressionIfFalse` is **not** evaluated (and hence cause no side effects).
- If the condition evaluates to **false**, then the situation is converse, the resulting values is given by the evaluation of the **false** branch of the *conditional expression*, and the **true** branch is **not** evaluated.

A common use of the conditional expression is to assign the value `x` or `y` to `a`, depending on an easily decidable condition, say `x>y`.

Sample Code

See the sample code below:

```
#include <iostream>
using namespace std;

int main() {
int x = 7;
int y = 5;
int a = ( x > y ) ? x : y; // here we are using conditional expression to evaluate
cout << "value of a using conditional expression is: " << a <<endl;
//this is equivalent to:

if (x > y){      // here we are usig if-else which will gave same output
    a = x;
    cout << "value of a using if-else is: " << a<<endl;
}else {
    a = y;
    cout << "value of a using if-else is: " << a <<endl;
}
return 0;
}
```

As you can see, this makes simple conditionals all the simpler.

> **Note:** Use the conditional expression only if you feel that it really enhances the readability.

Extra Task:

See if you can come up with a few uses of the conditional constructs you have just learned.

This marks the end of the chapter on *conditional statements*. In the next chapter, we'll discuss the interesting concept of *loops* in C++.