# 8.4 — Constexpr if statements

Normally, the conditional of an if-statement is evaluated at runtime.

However, consider the case where the conditional is a constant expression, such as in the following example:

```
#include <iostream>
int main()
{
    constexpr double gravity{ 9.8 };

    // reminder: low-precision floating point literals of the same type can be tested for equality
    if (gravity == 9.8) // constant expression, always true
        std::cout << "Gravity is normal.\n"; // will always be executed
    else
        std::cout << "We are not on Earth.\n"; // will never be executed

    return 0;
}</pre>
```

Because gravity is constexpr and initialized with value 9.8, the conditional gravity == 9.8 must evaluate to true. As a result, the else-statement will never be executed.

Evaluating a constexpr conditional at runtime is wasteful (since the result will never vary). It is also wasteful to compile code into the executable that can never be executed.

### Constexpr if statements C++17

C++17 introduces the **constexpr if statement**, which requires the conditional to be a constant expression. The conditional of a constexpr-if-statement will be evaluated at compile-time.

. . .

0

If the constexpr conditional evaluates to true, the entire if-else will be replaced by the true-statement. If the constexpr conditional evaluates to false, the entire if-else will be replaced by the false-statement (if it exists) or nothing (if there is no else).

To use a constexpr-if-statement, we add the constexpr keyword after the if:

```
#include <iostream>
int main()
{
    constexpr double gravity{ 9.8 };

    if constexpr (gravity == 9.8) // now using constexpr if
        std::cout << "Gravity is normal.\n";
    else
        std::cout << "We are not on Earth.\n";

    return 0;
}</pre>
```

When the above code is compiled, the compiler will evaluate the conditional at compile time, see that it is always true, and keep only the single statement std::cout << "Gravity is normal.\n";.

In other words, it will compile this:

```
int main()
{
   std::cout << "Gravity is normal.\n";
   return 0;
}</pre>
```

## **Best practice**

Favor constexpr if statements over non-constexpr if statements when the conditional is a constant expression.

Modern compilers and if statements with constexpr conditionals C++17

• • •

0

For optimization purposes, modern compilers will generally treat non-constexpr if-statements that have constexpr conditionals as if they were constexpr-if-statements. However, they are not required to do so.

A compiler that encounters a non-constexpr if-statement with a constexpr conditional may issue a warning advising you to use if constexpr instead. This will ensure that compile-time evaluation will occur (even if optimizations are disabled).



#### **Next lesson**

8.5 Switch statement basics



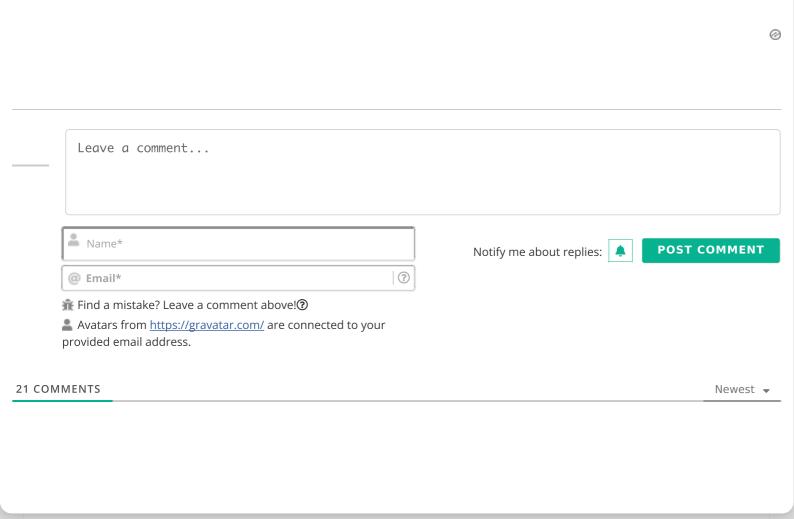
**Back to table of contents** 



#### **Previous lesson**

8.3 Common if statement problems

• • •



We and our partners share information on your use of this website to help improve your experience.

Do not sell my info:

OKAY

×