The sizeof Operator

The `sizeof` operator returns the amount of memory allocated for a variable. The operator takes a single operand, which must be **either an expression**, such as the name of a variable or a **type name**. Type names must be enclosed in parentheses.

If used with a variable or an expression, the `sizeof` operator returns the **number of bytes** required to store the value of that expression. If used with a **type**, `sizeof` returns the number of bytes required to store a value **of that type**.

```
int a = 42;
cout << sizeof a << endl;       // 4 (on our platform)
cout << sizeof(double) << endl; // 8
cout << sizeof 7LL + 4 << endl; // 12-WHY?
```

The first line prints the bytes required to store the `int` variable `a`; the second line prints the number of bytes required to store **any value** of type `double`. The third is more confusing. On our platform, a `long long` should take `8` bytes, but this prints `12`. **Why?**

Simple: `sizeof` is a **unary** operator. That means that the expression shown here reads as `sizeof(7LL) + 4` which is `8 + 4` or `12`. The fix is equally simple: **always parenthesize arguments** to the `sizeof` operator, **even when they are not needed**.