2.2 — Void

BY ALEX ON FEBRUARY 11TH, 2015 | LAST MODIFIED BY ALEX ON SEPTEMBER 6TH, 2016

Void is the easiest of the data types to explain. Basically, it means “no type”!

Consequentially, variables can not be defined with a type of void:

```
void value; // won't work, variables can't be defined with a void type
```

Void is typically used in several different contexts:

1) Most commonly, as a way to indicate that a function does not return a value:

```
void writeValue(int x) // void here means no return value
{
    std::cout << "The value of x is: " << x << std::endl;
    // no return statement, because the return type is void
}
```

2) In C, as a way to indicate that a function does not take any parameters:

```
int getValue(void) // void here means no parameters
{
    int x;
    std::cin >> x;
    return x;
}
```

The explicit use of keyword void to mean “no parameters” is a holdover from C, and is not required in C++. The following code is equivalent, and preferred in C++:

```
int getValue() // empty function parameters is an implicit void
{
    int x;
    std::cin >> x;
    return x;
}
```

Rule: Use an empty parameter list instead of void to indicate no function parameters are expected

3) The void keyword has a third (more advanced) use in C++ that we cover in section **6.13 -- Void pointers**. Since we haven’t covered what a pointer is yet, you don’t need to worry about this case for now. ☺

2.3 -- Variable sizes and the sizeof operator

Index

2.1 -- Fundamental variable definition, initialization, and assignment

Share this:

Facebook Twitter Google Pinterest