**Scope**

**Learn what scope is, the different ways of defining scope, and how a program uses scope.**

When we write programs in C, we define many different names for variables, functions, and other identifiers. We may find that our access to these named components is unavailable in certain parts of our program.

Is there a way for us to know when certain names are available? As it turns out, the *scope* of the name tells us exactly this.

**What Is Scope?**

Scope refers to the part of a program where a name has a meaning.  
For example:

int someFunction() {  
  int myVariableName = 20;  
}    
  
int main() {  
  char myVariableName[] = "10";   
}

In the example above:

* **myVariableName** in **someFunction()** refers to an **int** variable whose value is **20**
* **myVariableName** in **main()** refers to an **char[]** variable whose value is **"10"**
* Both these variables have the same name but different meaning depending on their scope

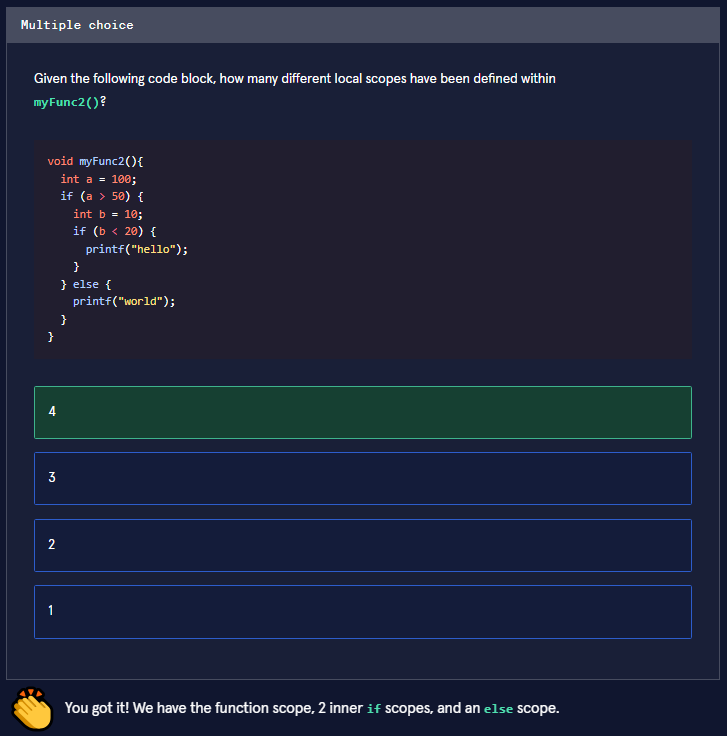
We’ve actually already done a lot of scoping through out this course!

In C, scopes are defined within curly braces, **{}**, and therefore can be created using constructs like **if** statements and functions.  
For example:

void myFunc() {  
  // function scope  
  int b = 10;  
  if (b > 5) {  
    // if scope  
    int c = 4;  
  }  
}

There are two scopes in the above example:

* A scope within the **myFunc()** function defined with the outer braces
* A scope within the **if** statement curly braces, also inside the **myFunc()** scope
* The **if** braces “enclose” the meaning of **c** within it and make it unknown outside of **if** block.
* The **myFunc()** braces “enclose” the meaning of **b** within it and make it unknown outside of the function.



#### Local Scope

Let’s look at another example:

void myFunc() {  
  int b = 200;  
}  
  
int main() {  
  int a = 10;  
  printf("The value of b is %d", b); // `printf()` trying to access `b`  
}

The above example defines the name **a** known only to **main()** and **b** known only to **myFunc()**.

The local scope means that:

* **b** can only be accessed within the function body of **myFunc()**
* Outside of **myFunc()**, **b** cannot be accessed. In other words **b** is not known to other scopes, and therefore they don’t have it declared

If we tried to run this program, we would get a compilation error.

**error: 'b' undeclared (first use in this function)**

This is because the name **b** is not defined in the **main()** function local scope. Similar situations occur when we use other constructs like **if** statements.

void myFunc() {  
  printf("The value of c is %d", c);  
}  
  
int main() {  
  int c = 10;  
  if (c > 4) {  
    int b = 20;  
  }  
  printf("The value of b is %d", b);  
}

In the example above, the following would occur:

* The name **c** is defined locally within the body of **main()** and is unknown outside of the function. **c** is part of the **main()** scope.
* **b** is defined locally within the body of the **if** statement and is unknown outside of the **if** block. **b** is part of the **if** block scope.
* **b** may appear as if it’s part of **main()** scope but it’s really the **if** scope that is defined inside the **main()** scope.

We would get another compilation error here because we are trying to access the name **b** in the **main()** scope, when it is only defined in the **if** scope.

