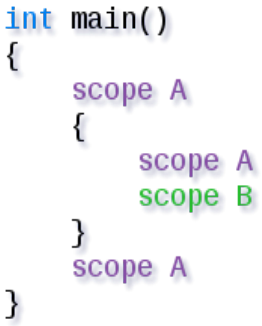
Variable Scope

Variables that are defined at the top of the program before the main function are called global variables. These variables are usually constant and are able to be used anywhere in the program (all functions). In other words, they have a **global scope**.



An example of a **global scope** variable is:

```
const int globalVar = 50; //Usually const
int main() { … } //Rest of program can use globalVar
```

Variables declared in between curly brackets have a **local scope**. This scope can be found by finding the closest left curly bracket above the declaration of the variable (beginning of scope) and finding the matching left curly bracket (end of scope). Even if there are nested curly brackets, that is still a part of the scope. Here is an example:

```
int main () {
    int x = 100;
    if(x == 100) { //Nested curly bracket
        cout << x << endl; //x valid within nested curly brackets
        int y = 200;
    } //Nested curly bracket
    x += 300;
    cout << y << endl; //ERROR! Cannot do! Out of local scope
}
```

As you can see, you cannot access a variable out of scope. Variable **y** was only valid between the blue curly brackets. However, **x** can be used anywhere between the **red** curly brackets.

One quirk to this is that you can use the same variable name in nested scopes.

```
int main () {
    int x = 100;
    if(x == 100) {
        cout << x << endl; //100
        int x = 200;
        cout << x << endl; //200
    }
    cout << x << endl; //100
}
```

Never do this as it obfuscates the program. However, it is valid. If this happens, the **x** in the blue curly brackets (**x = 200**) will override the **x** from the **red** curly bracket scope. The first cout will display 100. The second cout will display 200 (the override). The third cout is in the **red** curly bracket scope and will display 100 again.