

CSCI 1380

FEBRUARY 15, 2016

REVIEW:

THE **#INCLUDE** DIRECTIVE & USING PRE-
DEFINED FUNCTIONS

THE `#include` DIRECTIVE

To use a library, you must tell the program to include its **header file**.

The syntax for including the `iostream` library is:

```
#include <iostream>
```

```
#include <iostream>
using namespace std;
```

```
int main()
{
    double x = 4;

    cout << x;
}
```

```
#include <iostream>
using namespace std;
```

```
int main()
{
    double x = 4;

    cout << x;
}
```

This is the **#include directive**. It directs the compiler to *include* (prepend) the **iostream library** in this program.

```
#include <iostream>
using namespace std;
```

```
int main()
{
    double x = 4;
```

```
    cout << x;
```

```
}
```

Including the `iostream` library allows us to use things like `cin` and `cout`.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    double x = 4;
    double result = sqrt(x);
    cout << x;
}
```

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    double x = 4;
    double result = sqrt(x);
    cout << x;
}
```

Another **#include directive**.
It directs the compiler to include the **math library** here.

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
    double x = 4;
    double result = sqrt(x);
    cout << x;
}
```

As a result, we can use various **pre-defined math functions.**

```
#include <iostream>
#include <cmath>
using namespace std;
```

```
int main()
{
    double x = 4;
    double result = sqrt(x);
    cout << x;
}
```

SOME `cmath` LIBRARY FUNCTIONS...

- `sqrt(x)`:
 - Requires one input parameter (**x**) of type **double**.
 - Returns a **double** (the square root of **x**).
- `pow(x, y)`:
 - Requires two input parameters (**x**, **y**) of type **double**.
 - Returns a **double** (x^y).

SOME `cmath` LIBRARY FUNCTIONS...

- `floor(x)`:
 - Requires one parameter (**x**) of type **double**.
 - Returns a **double** (the largest whole number $\leq x$).
- `ceil(x)`:
 - Requires one input parameter (**x**) of type **double**.
 - Returns a **double** (the smallest whole number $\geq x$).

EXERCISE

With a partner, write out the function **headers** for the `floor` and `ceil` (predefined) math functions.

SOME `cctype` (CHARACTER) LIBRARY FUNCTIONS...

- `tolower(x)`:
 - Requires one parameter (`x`) of type `int` or `char`.
 - Returns an `int` (the lowercase value of `x`).
- `toupper(x)`:
 - Requires one parameter (`x`) of type `int` or `char`.
 - Returns a `int` (the uppercase value of `x`).

```
#include <iostream>
#include <cctype>
using namespace std;
```

```
int main()
{
    char c, lowercase, uppercase;
    cout << "Enter a letter:";
    cin >> c;
    lowercase = tolower(c);
    uppercase = toupper(c);
}
```

EXERCISE

With a partner, write out the function **headers** for the `tolower` and `toupper` (predefined) character functions.

FUNCTIONS (CONT.):

SCOPE

SCOPE

- **Every function has its own memory space.** Meaning:
 - All variables and parameters declared in a function refer to memory *allocated* in that space.
 - When a function ends, its variables are *deallocated*.

EXAMPLE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

EXAMPLE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

EXAMPLE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

CONCEPTUAL WALK-THROUGH

```
sum = sum_three(5, 10, 15);
```

- Computer does the following:
 - Allocates memory for formal parameters.
 - Assigns actual parameter values.
 - Allocates memory for declared variable sum.
 - Calculates the sum.
 - Returns the sum (all memory is de-allocated).

IMPLICATIONS

- Variables with the same name in different functions do not refer to the same memory.
- Functions cannot use variables declared in another function...They are **out of scope**.

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

IMPLICATIONS

- Variables with the same name in different functions do not refer to the same memory.
- Functions cannot use variables declared in another function...They are **out of scope**.

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
```

```
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
```

```
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
13
14 }
```

The diagram illustrates function scope using two orange boxes. The first box encloses the function definition for `sum_three`, which includes the return type `double`, the function signature, and the return statement. The second box encloses the `main` function, which calls `sum_three`. A curved arrow points from the right side of the `sum_three` box to the right side of the `main` box, indicating the call site.

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```

FUNCTION SCOPE

```
1  #include <iostream>
2  using namespace std;
3
4  double sum_three(double x, double y, double z) {
5      double sum;
6      sum = x + y + z;
7      return sum;
8  }
9
10 int main() {
11     double sum;
12     sum = sum_three(5, 10, 15);
14 }
```