FUNCTIONS AND VARIABLE SCOPE

Each function defines a new, unique scope
SCOPE

• Scope is the location in a program where an identifier is visible or accessible

• Named scopes
  • Global
  • Class
  • Local
  • Block / control statement

• Scope resolution takes place from the tightest to the widest
GLOBAL VARIABLES AND FUNCTION COUPLING

- Functions that only operate on parameter values can be tested independently.
- Functions that share data through global variables are coupled and must be tested together.
- The level of complexity increases rapidly as each new coupled function is added.
- The complexity of coupled functions limit the size of programs.
VISUALIZING THE COMPLEXITY OF COUPLED FUNCTIONS
VISUALIZING THE COMPLEXITY OF COUPLED FUNCTIONS
VISUALIZING THE COMPLEXITY OF COUPLED FUNCTIONS
One of the many advantages of the object-oriented programming model is that it provides an intermediate scope (between global and local):

- Some functions can see or access class scope variables or data
- Class scope variables are hidden from most of the program
- Covered in greater detail later
LOCAL VARIABLES

- Variables defined inside of a function; includes function parameters

double average(...) {
    double sum = 0;
    .
    .
    .
}

int to_seconds(int hrs, int mins, int secs) {
    int h = hrs * 3600;
    int m = mins * 60;
    return h + m + secs;
}
double average()
{
    double sum = 0;
    int count = 0;

    while ((...))
    {
        int data;
        cin >> data;
        sum += data;
        count++;
    }

    return sum / count;
}
The compiler searches for variables from the tightest to the widest scope.

```cpp
int nlines = 10;
int counter = 100; // global

void function() {
    int counter = 200; // local
    cout << "nlines " << nlines << "counter " << counter;
}
```
double average()
{
    double sum = 0;
    int count = 0;
    . . .
    return sum / count;
}

double random()
{
    static double x = 0;
    x = x * (x + 1) % 2147483648L;
    return x;
}
extern int counter;

int report()
{
    return counter;
}