Scope
Global & Local Identifiers
Definition:
the portions of a program
where an identifier “is defined”
(may be used).
Local Variables

Definition:
a variable declared inside a Block.

Scope:
from the Declaration statement to the end of the Block
Local Variables

Example 1:

```c
void fun() {
    float f;
    int a;    // scope of a begins
    ...
}        // scope of a ends

void main() {
    ...
    a++;    // syntax error: a undeclared
}
```
Local Variables

Example 2:

```cpp
void fun() {
    for (int i = 0; i < 3; i++) {
        cout << i;
    } // scope of i ends here
    cout << i; // syntax error: i undef
}
```
void fun(int x) {
    cout << i; // syntax error: i undefined
    if (x < 0) {
        int i=x;  // scope of i begins
        cout << i; // OK
        ...
    }          // scope of i ends
    cout << i; // syntax error: i undefined
}
// scope of arguments is entire function
void fun(int x) // scope of x begins
{
    ...
} // scope of x ends

void main() { // syntax error: x undefined.
    ... 
    x++; // syntax error: x undefined.
}
Global Variables

Definition:
a variable declared outside any Block.

Scope:
from the Declaration statement to the end of the source file
Global Variables

Allocation:
before main() begins

Deallocation:
after main() ends
using namespace std;

int x = 3; // scope of x begins

void fun() {
    ...
    x--; // OK
}

void main() {
    ...
    x++; // OK
}

// scope of x ends
but...

Use of Global Variables is a Major Violation of Structured Programming!

(almost as bad as TWTSNBU)

[The Word That Shall Not Be Uttered]
Global Variables

Why are Global Variables a Major Violation of SP?

1. Inhibits Reuse of Functions

```c
int x; // global
...
// 300 lines of code here

// So Great! copy/paste/use in ANY program!
void geniusFunction() {
    ...
    x = x * 2; // don’t forget to copy/paste
    ...
    // global declaration of x!
}
```
Global Variables

Why are Global Variables a Major Violation of SP?

2. Side Effect: value of a global variable changes for no apparent reason

```c
int x;   // global
void fun() { ... x++; ... }
void main() {
  x = 0;
  fun();
  cout << x; // prints 1
}  // where did x change??
```
Global Constants

Definition: a constant declared outside any block.

Scope: same as Global Variables

Allocation: same as Global Variables

DeAllocation: same as Global Variables
but... Global Constants

They are NOT considered a Major Violation of Structured Programming
Global Constants

Why are they not a Major Violation of SP?

1. No Side Effects
   since constants don’t change value

2. Promote Consistency
   value is the same for ALL functions

But: They DO inhibit reuse,
   but techniques can be used to limit problems

Therefore: Advantages outweigh Disadvantages
Global Constants

// same value of pi for all functions
const double pi = 3.1415926;
double circleArea(double r) {
    return pi * r * r;
}
double circumference(double r) { ... }
double sphereVolume(double r) { ... }
double sphereSurface(double r) { ... }
Definition: when a local identifier has the same name as a global (oll) identifier, blocking access to the global (oll) identifier.

oll: “or less-local”
Masking
Also called
Hole In Scope
int x = 0;
void fun1 () {
    x = 5;
}
void fun2() {
    float x;
    x = 3;
}
void main() {
    x = 4;
}
void main() {
    int x = 0;       // local x
    if (x <= 0) {
        float x;    // more-local x
        x = 4;
        cout << x;  // prints 4
    }
    cout << x;      // prints 0
}
**Global Functions**

**Definition:** a function declared outside any other any block.

**Scope:** where the function may be invoked.
- begins where the function is Declared or Prototyped
- ends at the end of the source file.
Local Functions

Definition: a function declared inside the declaration of another function.

Scope: inside the “outer” function only

Not allowed by most C++ compilers but available in other languages
Local Functions

void globalFun() {
    void localFun() { // C++ syntax error
        ...
        // just cut/paste
        // outside/before
        localFun();
        // globalFun() {
    }
    ...
} // end globalFun()
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--; g *=2;
    cout << a << " " << b << " " << c << " "
    << g << endl;
}
void main() { 
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
    << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--;  g *=2;
    cout << a << " " << b << " " << c << " "
         << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
         << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--; g *=2;
    cout << a << " " << b << " " << c << " "
     << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
     << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--; g *=2;
    cout << a << " " << b << " " << c << " "
         << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
         << endl;
}
int g = 3;

void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *= 2;
    cout << a << " " << b << " " << c << " "
         << g << endl;
}

void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
         << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *=2;
    cout << a << " " << b << " " << c << " " << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
a++; b--; g *=2;
    cout << a << " " << b << " " << c << " " << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--;  g *=2;
    cout << a << " " << b << " " << c << " " << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--;  g *=2;
    cout << a << " " << b << " " << c << " "
         << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y, z);
    cout << y << " " << z << " " << g
         << endl;
}
int g = 3;

void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *= 2;
    cout << a << " " << b << " " << c << " "
        << g << endl; //note: y & z out of scope!
}

void main() {
    int y=2, z=10;
    fun(y, z);
    cout << y << " " << z << " " << g
        << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--;  g *=2;
    cout << a << " " << b << " " << c << " "
    << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
    << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *=2;
    cout << a << " " << b << " " << c << " "
        << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
        << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++;  b--;  g *=2;
    cout << a << " " << b << " " << c << " " << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *=2;
    cout << a << " " << b << " " << c << " "
         << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y,z);
    cout << y << " " << z << " " << g
         << endl;
}
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *= 2;
    cout << a << " " << b << " " << c << " "
         << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y, z);
    cout << y << " " << z << " " << g
         << endl;
} program ending...
```cpp
int g = 3;
void fun(int a, int & b) {
    int c = a + b;
    a++; b--; g *= 2;
    cout << a << " " << b << " " << c << " "
        << g << endl;
}
void main() {
    int y=2, z=10;
    fun(y, z);
    cout << y << " " << z << " " << g
        << endl;
} program ended
```
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4;
        cout << x;
    }
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4;
        cout << x;
    }
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) { True!
        float x;
        x = 4;
        cout << x;
    }
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4;
        cout << x;
    }
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4; Changes the Most-Local
        cout << x;
    }
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4;
        cout << x << " 4";
    }
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4;
        cout << x << " ";
    }
    // locals declared in this block are deallocated
    cout << x;
}
void main() {
    int x = 0;
    if (x <= 0) {
        float x;
        x = 4;
        cout << x << " ";
    }
    cout << x;
}
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope</strong></td>
<td>the portions of a program where an Identifier is defined</td>
</tr>
<tr>
<td><strong>Global</strong></td>
<td>an identifier declared <em>outside</em> of any function. Its scope is from its declaration through the end of the source file.</td>
</tr>
<tr>
<td><strong>Local</strong></td>
<td>an identifier declared <em>inside</em> a function. Its scope is from its declaration through the ending brace of the block in which it is declared.</td>
</tr>
<tr>
<td><strong>Side Effect</strong></td>
<td>when an invoked function changes the value of a global variable, the change is not obvious in the invoking function.</td>
</tr>
<tr>
<td><strong>Masking</strong></td>
<td>when a local identifier has the <em>same name</em> as a global identifier, blocking access to the global identifier.</td>
</tr>
<tr>
<td><strong>Hole In Scope</strong></td>
<td>synonym for “Masking”</td>
</tr>
</tbody>
</table>