The C++ Language

Scope and Lifetime of Variables

Scope

- The region of program code where it is legal to reference (use) an identifier.
- Example

```
if (alpha > 3)
{
    int n;
    cin >> n;
    beta = beta + n;
}
```

The scope of the identifier n is the body of the if statement.

Categories of Scope

- Local scope
  - The scope of an identifier declared inside a block extends from the point of declaration to the end of that block.
- Global scope
  - The scope of an identifier declared outside of all blocks extends from the point of declaration to the end of the entire file.

Local Scope (Example)

```
int x;
…
while (x < 0)
{
    int y;
    cin >> y;
    x = x + y;
}
```

The scope of y is *local* to the body of the while loop.

The scope of x is *nonlocal* to the body of the while loop.

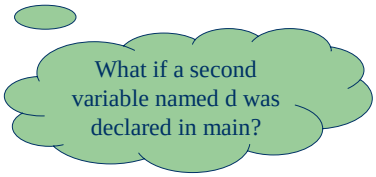
Global Scope (Example)

```
double d;

int main ()
{
    int y = 0;
    cin >> d;
    if (d < 0)
    {
        d = y;
    }
    return 0;
}
```

The identifier d has a *global* scope.

It can be referenced anywhere in the program below where it is declared, including inside main and in the body of the if statement.



Name Precedence

- When a block declares a local identifier with the same name as a global identifier, the local identifier takes precedence.
- Example

```
double d = 1.2;
int main()
{
    double d = 3.5;
    cout << d;          // 3.5 is printed
}
```

Scope Rules

- Global identifier
  - Declaration to the end of the file.
- Local identifier
  - Declaration to the end of the block.
- Does not include any nested block that contains a locally declared identifier with the same name.

Scope Examples

```
int a;
char c;

int main ()
{
    int a;
    while (…)
    {
        int b;
        char c;
    }
    return 0;
}
```

Variables

- Recall that variables in C++ are made up of four parts
  - Name
  - Type
  - Memory location
  - Value

Lifetime

- The period of time during program execution when an identifier has memory allocated to it.
- A local variable's lifetime
  - Starts when entering the block
  - Ends when the block is exited.
- A global variable's lifetime
  - Duration of the entire program.

Lifetime Example

```
                    // c is neither in scope, nor has
                    // memory allocated to it

while (…)
{                   // c is allocated memory
    char c;         // c is in scope and has memory

}                   // c's memory is deallocated

                    // c is neither in scope, nor has
                    // memory allocated to it
```


Lifetime != Scope

```
int a;
char c;

int main ()
{
    int a;
    while (…)
    {
        int b;
        char c;
    }
    return 0;
}
```

The *lifetime* of global identifiers a and c extends for the duration of the entire program.

However, the global variable a is not in *scope* in main. It is hidden by the local variable a declared in main.

Likewise in the while loop, the global variable c has memory allocated to it but is not in scope.