Variable Scope

Variable Scope

- **Scope** is a characteristic of a variable that defines from which functions that variable may be accessed.
  - **Local variables** can only be accessed within the functions in which they are created.
  - **Global variables** can be accessed by any function in the program.

Variable Scope

- So far in the course, you've almost assuredly been working only with local variables.

```
int triple(int x); // declaration

int main(void)
{
   int result = triple(5);
}

int triple(int x)
{
   return x * 3;
}
```

- Here, `x` is **local** to the function `triple()`. No other function can refer to that variable, not even `main()`. `result` is **local** to `main()`.

Variable Scope

- Global variables exist too. If a variable is declared outside of all functions, **any** function may refer to it.

```
int triple(int x); // declaration

#include <stdio.h>

float global = 0.5050;

int main(void)
{
   triple();
   printf(“%f\n”, global);
}

void triple(void)
{
   global *= 3;
}
```

Variable Scope

- Why does this distinction matter? For the most part, local variables in C are **passed by value** in function calls.

- When a variable is passed by value, the **callee** receives a copy of the passed variable, not the variable itself.

- That means that the variable in the **caller** is unchanged unless overwritten.

Variable Scope

- No effect on `foo`. (Function declarations omitted for space.)

```
int main(void)
{
    int foo = 4;
    triple(foo);
}

int triple(int x)
{
    return x *= 3;
}
```

Variable Scope

- Overwrites `foo`. (Function declarations omitted for space.)

```
int main(void)
{
    int foo = 4;
    foo = triple(foo);
}

int triple(int x)
{
    return x *= 3;
}
```

Variable Scope

- Things can get particularly insidious if the same variable name appears in multiple functions, which is perfectly okay as long as the variables exist in different scopes.

Variable Scope

```
int increment(int x);

int main(void)
{
   int x = 1;
   int y;
   y = increment(x);
   printf(“x is %i, y is %i\n”, x, y);
}

int increment(int x)
{
   x++;
   return x;
}
```

Variable Scope

```
int increment(int x);

int main(void)
{
    int x = 1;
    int y;
    y = increment(x);
    printf("x is %i, y is %i\n", x, y);
}

int increment(int x)
{
    x++;
    return x;
}
```

Variable Scope

```
~~int increment(int x);~~

int main(void)
{
    int x_m = 1;
    int y;
    y = increment(x_m);
    printf("x is %i, y is %i\n", x_m, y);
}

int increment(int x_i)
{
    x_i++;
    return x_i;
}
```

Variable Scope

```
x is 1, y is 2
```