Lecture #5: Functions and Scope

Reading: Holmes, Chapter 5
Handouts: Scope Example
Homework: Assignment #3, due October 28, 1997 at 11:59:59PM

---

1. Greetings and felicitations
2. Functions
   - declarations, prototypes (ANSI, K&R)
   - definitions
   - pointers to functions
   - note problems of forward references (default type int, x - sqrt(4))
   - note parameter passing convention for floats (K&R raises to double, ANSI doesn't unless no or old prototype)
   - passing arrays
3. Scope
   - heap (persistent) v. stack (transient)
   - show example

Scope Example

This simple program shows how the C language handles scope.

```
/*
 * this is a do-nothing program that demonstrates the
 * scope rules of C.
 */
#include <stdio.h>
#include <stdlib.h>

/*
 * forward declarations
 */
void g(int);
void h(void);

/* the top-level definition */
int variable = 1;

/*
 * the main routine
 */
int main(void)
{
        /* scope is function main */
        int variable = 2;

        printf("main(%d):variable = %d\n", __LINE__,
                                                        variable);

        /* now an inner block */
        {
                /* scope is the rest of this block */
                int variable = 3;

                printf("main(%d):variable = %d\n", __LINE__,
                                                        variable);

                /* now an even more inner block */
                {
                        /* scope is this block */
                        extern int variable;

                        printf("main(%d):variable = %d\n",
                                                __LINE__, variable);
                }
                /* end innermost block */

                printf("main(%d):variable = %d\n", __LINE__,
                                                        variable);
        }
        /* end inner block -- back to main block */

        printf("main(%d):variable = %d\n", __LINE__,
                                                        variable);
```

```
        /* now let's show how functions interact with scope */
        g(variable);

        /* bye! */
        return(EXIT_SUCCESS);
}

/*
 * now notice "variable" is a parameter
 * so it (effectively) overrides references to the
 * top-level variable)
 */
void g(int variable)
{
        printf("g(%d):variable = %d\n", __LINE__, variable);
        /* now let's call another function */
        h();
}

/*
 * this function has no declarations, so
 * let's see what it prints
 */
void h(void)
{
        printf("h(%d):variable = %d\n", __LINE__, variable);
}
```