

## Writing Your Own Functions and Libraries

- Sets of reusable libraries which provide generally usable functions are the basis for *all* modern software development, regardless of language or operating system.
- A *value-returning* function is one that returns a value to the calling program; a *void* function is one that does not return a value. At this time we'll consider both value-returning and void functions.

```

/*-----
| Screen.c
| Mini-terminal controller for vt100 (ANSI) terminals
| Author: M.B. Feldman, The George Washington University
| Last Modified: March 2006
|-----*/
#include <stdio.h>
/*-----
| clearScreen () clears the screen of the terminal
|-----*/
void clearScreen()
{
    printf("\033[2J");
}
/*-----
| beep () causes the terminal to beep once
|-----*/
void beep()
{
    printf("\007");
}
/*-----
| moveCursor (int row, int column) moves the cursor to the
| given row/column position. Generally terminal is 24 x 80.
|-----*/
void moveCursor(int row, int column)
{
    printf("%s%02d%c%02d%c", "\033[", row, ';', column, 'H');
}

```

```
/*-----  
| Smiley.java  
| Draws a stupid smiley face on the screen  
| Author: M.B. Feldman, The George Washington University  
| Last Modified: March 29, 2006  
-----*/  
#include <stdio.h>  
#include "Screen.c"
```

```

int main()
{
    clearScreen();
    beep();
    moveCursor (7, 34);
    printf ( "HAVE A NICE DAY!");
    moveCursor (9, 39);
    printf(    "_____");
    moveCursor (10, 37);
    printf(   "/          \\\");
    moveCursor (11, 36);
    printf(  "/          \\\");
    moveCursor (12, 35);
    printf(" |          |");
    moveCursor (13, 35);
    printf(" |    0    0    |");
    moveCursor (14, 36);
    printf( "\\      o      /");
    moveCursor (15, 37);
    printf(  "\\  \\  \\____/  /");
    moveCursor (16, 38);
    printf(   "\\      /");
    moveCursor (17, 39);
    printf(    "-----");
    moveCursor (24, 1);

    return 0
}

```

```

/*-----
| MinMax.c
| Provides minimum and maximum functions for float values
| Author: M.B. Feldman, The George Washington University
| Last Modified: March 2006
|-----*/

/*-----
| computes and returns the lesser of its float parameters
|-----*/
float minimum (float value1, float value2)
{
    if (value1 < value2)
    {
        return value1;
    }
    else
    {
        return value2;
    }
}

```

```
/* -----  
| computes and returns the greater of its float parameters  
| -----*/  
float maximum (float value1, float value2)  
{  
    if (value1 > value2)  
    {  
        return value1;  
    }  
    else  
    {  
        return value2;  
    }  
}
```

```

/*-----
MinMaxThree.c
Demonstrates MinMax library
finds minimum and maximum of three float values
Author: M.B. Feldman, The George Washington University
Last Modified: March 2006
-----*/
#include <stdio.h>
#include "MinMax.c"
int main()
{
    float num1, num2, num3, min, max = 0.0;

    printf ("Please enter three float values > ");
    scanf ("%f", &num1);
    scanf ("%f", &num2);
    scanf ("%f", &num3);

    min = minimum (num1, num2);
    min = minimum (min, num3);
    printf ("Minimum value: %f\n", min);

    max = maximum (num1, maximum (num2, num3));
    printf ("Maximum value: %f\n", max);

    return 0;
}

```

```
----- HERE IS MINIMUM -----  
float minimum (float value1, float value2)  
{  
    if (value1 < value2)  
    {  
        return value1;  
    }  
    else  
    {  
        return value2;  
    }  
}
```

```
----- HERE IS PART OF MAIN -----  
float num1, num2, num3, min, max = 0.0;  
  
printf ("Please enter three float values > ");  
scanf ("%f", &num1);  
scanf ("%f", &num2);  
scanf ("%f", &num3);  
  
min = minimum (num1, num2);  
min = minimum (min, num3);  
printf ("Minimum value: %f\n", min);
```

```
----- HERE IS MAXIMUM -----  
float maximum (float value1, float value2)  
{  
    if (value1 > value2)  
    {  
        return value1;  
    }  
    else  
    {  
        return value2;  
    }  
}
```

```
----- HERE IS PART OF MAIN -----  
float num1, num2, num3, min, max = 0.0;  
  
printf ("Please enter three float values > ");  
scanf ("%f", &num1);  
scanf ("%f", &num2);  
scanf ("%f", &num3);  
  
. . .  
  
max = maximum (num1, maximum (num2, num3));  
printf ("Maximum value: %f\n", max);
```

## "Call by Value" and "Call by Reference"

- In C, parameters are passed from a calling program to a function by *copying* them. This is known as "call by value".
- Let's write a function to *interchange* two values that we'll pass as parameters.

```

void interchange1 (float value1, float value2)
{
    float temp;    /* to hold a value during the swap */

    temp = value1;
    value1 = value2;
    value2 = temp;
}

```

```

----- MAIN -----
#include <stdio.h>
#include "interchange1.c"
int main()
{
    float x, y;

    x = 3.5;
    y = -10.8;

    /* Display before swapping */
    printf("before swap, x = %f, y = %f\n", x, y);

    /* Now swap and display after swapping */
    interchange1 (x, y);
    printf("after swap, x = %f, y = %f\n", x, y);

    return 0;
}

```

```

void interchange2 (float *pointer1, float *pointer2)
{
    float temp;    /* to hold a value during the swap */

    temp = *pointer1; /* value that pointer1 points to */
    *pointer1 = *pointer2;
    *pointer2 = temp;
}

```

```

----- MAIN -----
#include <stdio.h>
#include "interchange2.c"
int main()
{
    float x, y;

    x = 3.5;
    y = -10.8;

    /* Display before swapping */
    printf("before swap, x = %f, y = %f\n", x, y);

    /* Now swap and display after swapping */
    interchange2 (&x, &y); /* Addresses of x and y */
    printf("after swap, x = %f, y = %f\n", x, y);

    return 0;
}

```