

CSE 142 Computer Programming I

Functions I

© 2000 UW CSE

10/9/00

G1-1

Overview

Concepts this lecture

Functions

Function control flow

Two meanings of void

Pre-written functions

G1-2

Chapter 3

Read All!

3.1: Reusing program parts

3.2: Built-in math functions

3.3: Top-Down Design

3.4: Functions with no parameters

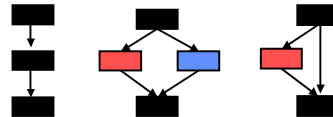
3.5: Functions with parameters

G1-3

Control Flow: Review

“Control flow” is the order in which statements are executed

We’ve discussed two forms of control flow so far: sequential and conditional (in more than one flavor)

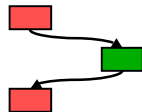


G1-4

Another Form of Control Flow

“**Functions**” (or “**procedures**” or “**subroutines**”) allow you to “visit” a chunk of code and then come back

The function maybe elsewhere in your own program, or may be code in another file altogether



G1-5

Why Use Functions?

Here’s one example:

Suppose we are writing a program that displays many messages on the screen, and...

We’d like to display two rows of asterisks (*’s) to separate sections of output:

```
*****  
*****
```

G1-6

Moving Toward a Solution

The result we want is this:

```
*****  
*****
```

And the basic code needed is this:

```
printf("*****\n");  
printf("*****\n");
```

G1-7

A Full Solution

```
#include <stdio.h>  
int main(void)  
{  
    /* produce some output */  
    ...  
    /* print banner lines */  
    printf("*****\n");  
    printf("*****\n");  
    /* produce more output */  
    ...  
    /* print banner lines */  
    printf("*****\n");  
    printf("*****\n");  
    /* produce even more output */  
    ...  
    /* print banner lines */  
    printf("*****\n");  
    printf("*****\n");  
    /* produce final output */  
    ...  
    return 0;  
}
```

G1-8

Anything Wrong With This?

It's correct C code

It fulfills the problem specification, i.e., gives the desired result

G1-9

Anything Wrong With This?

It's correct C code

It fulfills the problem specification, i.e., gives the desired result

What's "wrong" has to do with other issues such as

- how hard it would be change the program in the future

- How much work is it to write the same statements over and over

...

G1-10

What if...

Later on the client wants us to change...

- The number of rows of asterisks
- The number of asterisks per row
- Use hyphens instead of asterisks
- Print the date and time with each separator

...

How much work is involved?

G1-11

If We Want to Change Anything

... have to edit every "copy" of the code in the program.

... it's easy to overlook some copies.

... it can be hard to find them all (because they might not be written identically).

... it can be hard to find them all because code written identically may not serve the same logical purpose.

G1-12

One (Big) Idea Behind Functions

Identify a “sub-problem” that has to be solved in your program

Solve that sub-problem and write the code for it only once

Give that code a name: that makes it a function

Whenever you see that same sub-problem again, use the function name to say “go to that code now to take care of this problem, and don’t come back until you’re done”

G1-13

PrintBannerLines Function

For our print banner program, that strategy means this:

Take the repeated lines of code

```
printf("*****\n");  
printf("*****\n");
```

and wrap them up as a function, which we can call printBannerLines

G1-14

```
#include <stdio.h>  
int main(void)  
{  
    /* produce some output */  
    PrintBannerLines();  
    /* produce more output */  
    PrintBannerLines();  
    /* produce more output */  
    PrintBannerLines();  
    /* produce final output */  
    return 0 ;  
}
```

The code named **PrintBannerLines**

```
printf("*****\n");  
printf("*****\n");
```

G1-15

Discussion Question

In the new version of the program:

what do we have to do now if we want to change the banner? How many places in the program have to be changed?

G1-16

The Big Picture, So Far

You’ve now some colossal concepts:

- Functions

- Function control flow

- The motivation for functions

Coming right up...

- Syntax for defining a function

- Built-in C functions

G1-17

Syntax for Defining the PrintBannerLines Function

This is a typical pattern for a function declaration

```
/* write separator line on output */  
void PrintBannerLines (void)  
{  
    printf("*****\n");  
    printf("*****\n");  
}
```

G1-18

Two Key Features

1. The name of the function and
2. the function body: code that is to be executed when the function is called.

```
/* write separator line on output*/  
void PrintBannerLines (void)  
{  
    printf("*****\n");  
    printf("*****\n");  
}
```

function name
heading comment
function body (statements to be executed).
A function can have ANY number of ANY kind of statements.

Further details: void

The keyword **void** has two different roles in this function definition

```
/* write separator line on output*/  
void PrintBannerLines (void)  
{  
    printf("*****\n");  
    printf("*****\n");  
}
```

indicates that the function does not return a value.
indicates that the function has no parameters.

Oops – Two New Concepts

1. Return values: we will postpone for now
 2. Parameters: We will postpone this, too!
- Both concepts are very important in general, but not for this particular example

```
/* write separator line on output*/  
void PrintBannerLines (void)  
...
```

Using *PrintBannerLines*

```
#include <stdio.h>  
void PrintBannerLines (void)  
{  
    printf("*****\n");  
    printf("*****\n");  
}  
int main (void)  
{  
    /* produce some output */  
    ...  
    PrintBannerLines ();  
    ...  
    return 0;  
}
```

The definition of the function must precede all calls to it in the file.
Empty () is required when a parameter-less (void) function is called.

Some C Functions

We have already seen and used several functions:

```
int main (void)  
{  
    return 0;  
}
```

Function definition for **main()**

```
printf ("control", list);  
scanf ("control", &list);
```

Calls to the functions **printf()** and **scanf()**

Pre-written functions

Pre-written functions are commonly packaged in "libraries"

Every standard C compiler comes with a set of standard libraries

Remember **#include <stdio.h>** ?

- Tells the compiler you intend to use the "standard I/O library" functions
- **printf** and **scanf** are in the standard I/O library
- So are lots of other I/O related functions

There are (many) other useful functions in other libraries

Next Time

We'll continue our discussion about functions. We will examine how values are passed to functions, and how values come back

G1-25