CSE 142
Computer Programming I

Functions I

© 2001 UW CSE

Overview

Concepts this lecture

- Functions
- Function control flow
- Two meanings of the keyword void
- Pre-written functions (*library routines*)

Chapter 3

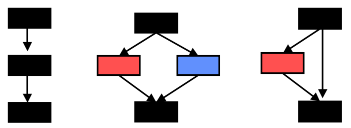
Read All!

- 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

Control Flow: Review

"Control flow" is the order in which statements are executed

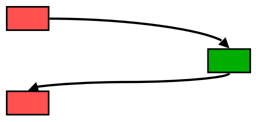
We've discussed two forms of control flow so far: sequential and conditional (in more than one flavor)



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



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 to separate sections of output:

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

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 we wants 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

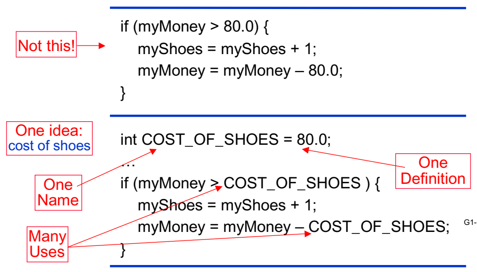
Deceptively Simple Big Idea

One idea ⇒One definition, many uses
One idea ⇒One definition, many uses
One idea ⇒One definition, many uses
One idea ⇒One definition, many uses
One idea ⇒One definition, many uses
One idea ⇒One definition, many uses
One idea ⇒One definition, many uses
One idea ⇒One definition, many uses

G1-13

Big Idea for Data: Symbolic Constants

One idea ⇒One definition, many uses



```
if (myMoney > 80.0) {
   myShoes = myShoes + 1;
   myMoney = myMoney – 80.0;
}
```

```
int COST_OF_SHOES = 80.0;
...
if (myMoney > COST_OF_SHOES ) {
   myShoes = myShoes + 1;
   myMoney = myMoney – COST_OF_SHOES;
}
```

G1-14

Big Idea for Code: Functions

One idea ⇒One definition, many uses

- One idea → Identify a “sub-problem” that has to be solved in your program
- Many Name → Choose a name to represent “*the solution of that problem by code”*
- One definition → Write that solution code (only once)
- Many Uses → 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*”

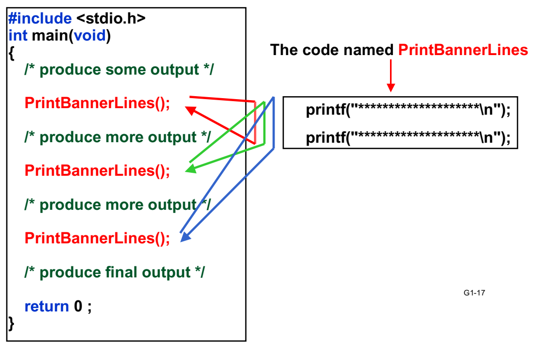
G1-15

PrintBannerLines Function

For our print banner program, that idea means this:

- Identify the idea
  print a banner
  (NOT *print two rows of asterisks*)
- Give the function that does that a name
  PrintBannerLines
- Define the solution by writing the code
  ```
  printf("*********************\n");
  printf("*********************\n");
  ```
- Whenever you want to print a banner, use the function name
  PrintBannerLines();

G1-16



```
#include <stdio.h>
int main(void)
{
   /* produce some output */

   PrintBannerLines();

   /* produce more output */

   PrintBannerLines();

   /* produce more output */

   PrintBannerLines();

   /* produce final output */

   return 0 ;
}
```

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

G1-17

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?

What if we want to print two rows of asterisks for something that isn’t a banner?

G1-18

The Big Picture, So Far

You've now some colossal concepts:

- Abstraction
- Functions
- Function control flow
- The motivation for functions

Coming right up...

- Syntax for defining a function
- Built-in C functions

G1-19

Syntax for PrintBannerLines

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

G1-20

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.

G1-21

Further details: *void*

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

indicates that the function does not return a value.

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

indicates that the function has no parameters.

G1-22

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)
...
```

G1-23

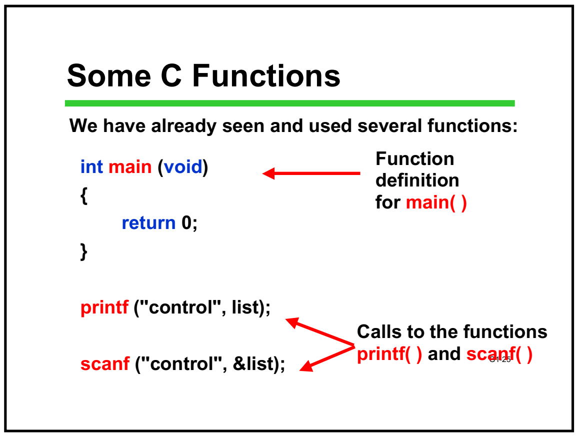
Using *PrintBannerLines*

```
#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.

G1-24



Some C Functions

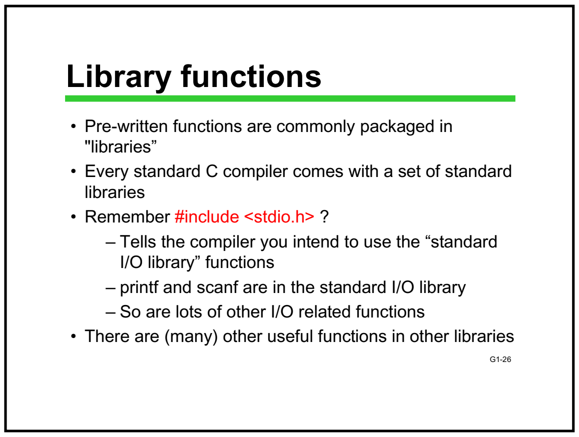
We have already seen and used several functions:

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

printf ("control", list);

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

Function definition for main( )

Calls to the functions printf( ) and scanf( )

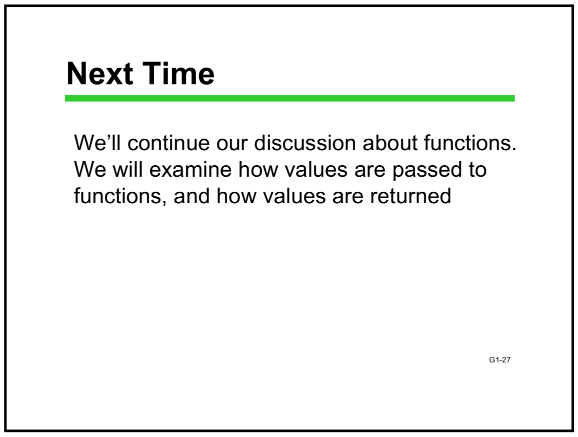
G1-25



Library 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

G1-26



Next Time

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

G1-27