C Programming

Functions
Part I

Functions

Concept and Design

What is a function?

A piece of code (with variables) that performs one or more actions for a specific purpose.

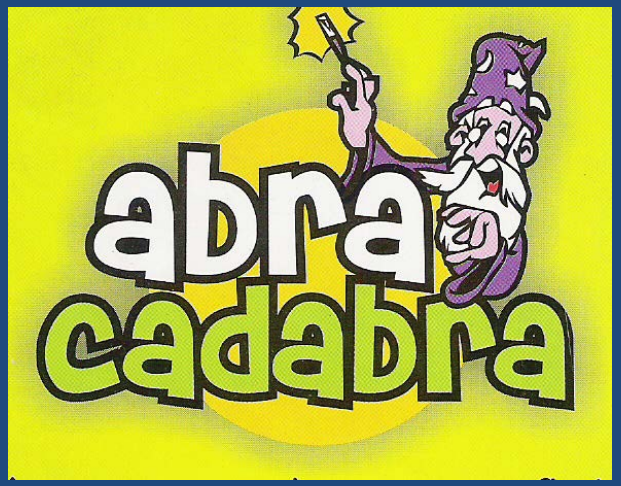
Examples so far:

- main()
- printf()

Two Ways to Look At Functions

- Creating our own

- Using someone else's



Why Use Functions?

"Do I know how to do this easily?"

"No" ➡ Create a function

Design Example

Sample program:

Initialize variables.

Load data.

Process data.

Save changed data.

Display report.

In the sample program, we could make a distinct function for each of the items in yellow.

Make a main() with
the five actions described.

Take each of the
four yellow actions
and
make a function
for each.

```
int main(void)
{
int i = 0;
   i = loadData();
   processData(i);
   saveData(i);
   displayReport(i);
}
```

Review!

How can you decide when to make a function?

Review!

What does a function do?

main()

Each console program needs exactly one main() function.

Aside:

If you get an error stating that you are missing WinMain, you set up the project wrong

Another aside:

void means "nothing"

e.g. no return

e.g. no parameters/arguments

Functions

Use

"Calling" Functions

Using functions

Calling functions

It's easier to see what the program does.

Design

Code

Easier!

So, Where Did They Come From?

Already exist?

Call them!

Don't already exist?

Create and call them!

Review!

What's a good reason to create a function?

Function Definition

Function Definition has:

- return type
- name
- parameter list
- opening brace
- statements that make up the body
- closing brace

```
int main(void)
{
int i = 0;
    i = loadData();
    processData(i);
    saveData(i);
    displayReport(i);
    return 0;
}
```

return type

name

parameter list

body

opening and closing braces

Everything before the opening brace is called the header of the function definition.

- return type
- name
- parameter list

Parameter?

Info going into a function

Review!

In the printf() call found in the sample program from week 1, what are the parameters?

Function Calls

Function calls have:

- name
- brackets (surrounding a (possibly-empty) argument list)

Example:

- loadData();

Other parts for a function call often found:

- assignment of the returned value
- non-empty argument list
- semicolon at end

Example:

```
i = loadData(value, 10);
```

i = loadData(value, 10);

- loadData is the function being called
- it takes two arguments as parameters
- it assigns a return value to the variable i

Arguments vs. Parameters

Arguments provide the values for parameters

Functions

Parameters

Communicating to a Function

Parameters / arguments contain information going into a function.

Two Forms

Calling a function:

Send information in as arguments.

Defining a function:

Receive information as parameters.

Very Important Distinction!

Parameter lists **must** contain datatype information about the data coming in.

Parameter lists **must** receive the information into parameters (which look like variables).

Argument lists **do not** have type information.

Argument lists could contain variables, constants, expressions, or even other function calls.

Example: Parameter List

int loadData(int count, long max)

- count and max can be used as variables inside the loadData() function
- Major difference: they both get their initial values from function calls.

Example: Argument List

loadData(counter, 40);

the value of the counter variable is assigned to the count parameter in loadData()

the value of the number 40 is assigned to the max parameter in loadData()

Note: no data types in the function call

Function definition

```
int loadData(int count, long max)
```

Parameters!

vs.

Function call

```
loadData(counter, 40);
```

Arguments!

Review!

Arguments? Parameters?
What's the difference?

How Many?

As many as you want

We can look at
an example
to understand
how functions are useful

Functions

Part 1 Summary

Functions: In Summary

- Create a function when you want to do something specific. Have it follow your design.
- You need exactly one main() function.
- You define functions by creating them.
- You call functions to use them within other functions.

- You send information to a function through a parameter list (a.k.a. arguments).