Chapter 7
User-defined Functions

CISC 5004 - Programming C++
Sam Kamens
Fall, 2022

FORDHAM

Functions (general)

- function: group repeated statements together to reduce redundancy and confusion
- Example

FORDHAM

Basics of functions

- function: A named list of statements
- function declaration
    - New function’s name, parameters, and return value
- function definition
    - block of statements (code) implementing the function
    - (declaration and definition are often together)
- function call
    - Invocation of the function, causing the function’s code to execute
- Incidentally, `main()` is a function too
- Example

Returning a value from a function

- Return one value using a `return()` statement
- Example

Parameters and Arguments

- parameter: function input specified in the function definition
  - Can have no parameters, or multiple parameters separated by commas
  - With no parameters, the parentheses must still be included.
- argument: Value provided for the parameter during a function all
- Single Parameter
- Multiple Parameters
- Can call functions from within functions
  - Example

Print functions

- Move printing out of `main()` for clarity and readability
- May not need to return a value
  - We indicate this using the `void` keyword
- Example
- Can be called multiple times
- Example: Menu System

Why use functions?

- Improve readability (without/with functions)
- Modular and incremental development
  - Modular development
    - Split program into modules that can be developed and tested separately
  - Incremental Development
    - write/compile/test a bit at a time
  - Function stub
    - Function definition with placeholder code
    - Example with stubs

Avoid redundant code

- e.g. abs() function
  - Easy for programmers to implement, but why repeat the same code everywhere?
- Choosing the right set of functions is an art and a skill
  - Behavior should be easy to recognize
  - Program's overall behavior should be clear
- General guideline:
  - Function shouldn't have more than about 30 lines of code
- Example

Writing mathematical functions

- Mathematical Function: Accept one or more numeric parameters and return a numeric result
  - Example: Convert feet/inches to centimeters
  - Temperature conversion
- Can call a non-void function in an expression
  - Example
- Functions with complex calculations can call other functions
  - e.g. Cylinder Volume

Functions with branches & loops

- A function's implementation can contain any C++ code
- Branches
  - Calculate shipping cost
  - Auction website fee
  - Important: Make sure all code paths include a `return()` statement
- Loops
  - Example (VScode): Compute the average of a list of numbers
  - Least common multiple

Unit Testing

- Testing
  - Process of verifying that program behaves correctly
  - May be difficult for large programs
    - Bugs could be anywhere
    - Multiple bugs may interact
  - Good practice is to test small parts (units) individually
    - **Unit testing**: individually test a unit (typically a function)
  - **Testbench**
    - Separate program to test a particular function with different input values
    - **Test vector**: One set of inputs

Unit Testing

- Test harness example: HrMinToMin()
- Using assert()
  - http://www.cplusplus.com/reference/cassert/assert/
- GoogleTest (TODO)

FORDHAM

How Functions Work

- Each function call creates a new **stack frame**
    - All new local variables
    - Illustration
- Detailed example

Functions: Common Errors

- Copy Paste Errors
- Returning the wrong variable
- Missing return()

Pass by reference

- What if you would like to update the value of one of the arguments?
  - Does this work?
- Instead: Pass by reference
  - Instructs the compiler to allow the function to change the input variables
  - This works better
  - Effectively allows for multiple return values

Pass by reference

- Use sparingly
  - Multiple unrelated return values
    - Preferred:
      ```
      int StepsToFeet(int baseSteps)
      int StepsToCalories(int baseSteps)
      ```
    - Not Preferred:
      ```
      void StepsToFeetAndCalories(int baseSteps, int& baseFeet,
      int& totCalories)
      ```
  - Multiple related return values
    - Calculating change
  - Avoid assigning pass-by-value parameters

FORDHAM

Reference Variables

- Can declare a variable to be a reference
    - Must initialize using an existing variable
    - `int &userValRef = userValInt;`
    - Example
    - Why do this?
        - Effectively rename a variable
        - Can be used to simplify complex expressions
            - `int & middleValue = vectorOfValues.at(vectorOfValues.size()/2)`

Functions with string/vector parameters

- Functions often change strings
- Passing complex data types (e.g. vectors) may be expensive
  - Would prefer to pass by reference to avoid extra memory allocation and copying
  - Can use `const` to indicate that the value should not be changed
  - Example
- So why not always use pass-by-reference?
  - Making a local copy of small variables allows the compiler to optimize
- In summary
  - Use pass by value for all except small input parameters
  - Use pass by reference as needed, but prefer return values

Functions with C string parameters

- As above, functions often modify C strings
  - Parameter is specified with `[]`
    - `void StrSpaceToHyphen(char modString[])`
  - Call the function without `[]`
    - `StrSpaceToHyphen(userStr);`
- Compiler automatically passes the C string as a pointer
  - So this means an array is always passed by reference
- Can also define the parameter as a pointer
  - `void StrSpaceToHyphen(char *modString)`

Scope of variable/function definitions

- The name of a variable/function is only visible to part of the program
    - **Scope**
- Scope starts when the variable is declared, and proceeds from there to the end of the function/program
- Variables declared inside a function are invisible outside
- **Global Variable**: Declared outside any function
- Be careful with globals
    - A local variable with the same name overrides a global
    - Changing a global can have side effects
    - Globals are typically limited to const variables
- Example

Function Scope

- Function scope also extends from the definition to end of the file
- Programmers would like to include main() first
  - Can’t do that if functions are to be called in main()
  - Solution: **function declaration** (aka function prototype)
  - Example

Default Parameter Values

- Allows the last (or last few) parameters to be optional
- Example
- Can not have alternating default/non-default parameters

FORDHAM

Function Name Overloading

- Can have multiple functions with the same name, as long as they have different parameters
    - `void PrintDate(int currDay, int currMonth, int currYear)`
    - `void PrintDate(int currDay, string currMonth, int currYear)`
- OK as long as the compiler can tell the difference
- Not OK:
    - `void PrintDate(int currMonth, int currYear, int printStyle)`
    - `Void PrintDate(int currDay, int currMonth, int currYear, int printStyle = 0)`

FORDHAM

Parameter Error Checking

- Good practice: Check the values of parameters
    - If incorrect, take appropriate action
        - Output an error message
        - Assign a valid value
        - return an error code
        - Exit the program
    - May require updates to program logic to allow for error processing
        - e.g. Return a value instead of using a void function
    - Example

Preprocessor and include

- Preprocessor
  - Scans file from top to bottom looking for lines starting with `#`
    - Remember, these are not comments
    - These lines are called **preprocessor directives**
  - **include directive:**
    - Inserts the contents of a file at that point
    - `#include “file”`
      - With quotes, the preprocessor looks for files in the current directory first
      - Convention: Use `.h` suffix
    - `#include <file>`
      - Don’t use local directory; only look in system locations
      - Generally used for standard headers

Separate Files

- Benefits of separating code into multiple files
  - Keep files from being too large
  - Allows others to use parts of the code
- Example
- Separate function declarations (.h) from definitions (.cpp)
  - Easier reuse
  - Can only have one set of definitions
- Header guards
  - Keep header contents from being included more than once
  - All headers should be guarded (example)

FORDHAM

Examples/Labs

- Domain Name Validation with Functions
- Unit tests to evaluate your programs