Chapter 7 - Modularity Using Functions: Part II

At a Glance

Lesson Contents

Chapter Notes

Overview

In Chapter 7, you learn about variable scope and storage classes. The concept of pass by value is refreshed and the concept of pass by reference is introduced. In the case study, you create a function that uses pass by reference to swap the values of two variables. In this chapter, you also learn about recursion and when to use recursion instead of iteration. Finally, you learn to identify and avoid common programming and compiler errors.

Objectives

- Variable scope
- Variable storage class
- Pass by reference
- Case study: Swapping values
- Recursion
- Common Programming and Compiler Errors

Variable Scope

Quick Quiz 1

1. What are local variables?

2. What is scope?

3. A variable with a(n)_____scope is simply one that has had storage locations set aside for it by a declaration statement made within a function body.

4. A variable with_____scope is one whose storage has been created for it by a declaration statement located outside any function.

Variable Storage Classes

Topic Tip	It is important to note that some compilers initialize local static variables the first time the definition statement is executed rather than when the program is compiled.

Topic Tip	The Programming Tip on page 341 summarizes the storage classes rules.

Pass by Reference

Topic Tip	Note that addresses have their own data type that more correctly are displayed in hexadecimal notation using the `%p` conversion sequence.

Quick Quiz 2

1. Where and how long a variable's storage locations are kept before they are released can be determined by the_____of the variable.

2. What are the registers in a computer?

3. What is pass by reference?

4. A variable that can store an address is known as a(n)_____.

Recursion

Topic Tip	In 1936, Alan Turing showed that although not every possible problem can be solved by computer, those problems that have recursive solutions also have computer solutions, at least in theory. For more information, see **http://en.wikipedia.org/wiki/Church-Turing_thesis** and **http://en.wikipedia.org/wiki/Turing_machine**.

Topic Tip	For an interesting example of another use of recursion, you may wish to research the Sierpinski triangle (**http://en.wikipedia.org/wiki/Sierpinski_triangle**).

Quick Quiz 3

1. Functions that call themselves are referred to as self-referential or _____functions.

2. When a function invokes itself, the process is called_____recursion.

3. What is mutual recursion?

4. With respect to computer program execution, what is the stack?

Additional Resources

1. Scope in C:
   https://en.cppreference.com/w/c/language/scope

2. Pass by Value vs. Pass by Reference:
   https://www.interviewsansar.com/pass-by-value-and-pass-by-reference-address-in-c/

3. Recursion:
   http://en.wikipedia.org/wiki/Recursion

4. Cprogramming.com Tutorial: Lesson 16: Recursion:
   www.cprogramming.com/tutorial/lesson16.html

Key Terms

- With respect to storage classes, the term `auto` is short for **automatic自动的**.
- When a function invokes itself, the process is called **direct recursion直接递归**.
- A variable with **global scope全局** is one whose storage has been created for it by a declaration statement located outside any function.
- A variable with global scope is more commonly termed a **global variable全局变量**.
- When using a pointer variable, the value that is obtained is always found by first going to the pointer for an address; this is called **indirect addressing间接寻址**.
- To use a stored address, C provides us with an **indirection operator间接运算符**, *.
- A variable with a **local scope局部** is simply one that has had storage locations set aside for it by a declaration statement made within a function body.
- Variables created inside a function are available only to the function itself; they are said to be local to the function, or **local variables局部变量**.
- A function can invoke a second function, which in turn invokes the first function; this type of recursion is referred to as **indirect间接** or **mutual recursion互递归**.
- Passing an address is referred to as a function **pass by reference传址**, because the called function can reference, or access, the variable using the passed address.
- In **pass by value传值**, a called function receives values from its calling function, stores the passed values in its own local parameters, manipulates these parameters appropriately, and directly returns, at most, a single value.
- A variable that can store an address is known as a **pointer variable指针变量**.
- Pointer variables are also **pointers指针**.
- **Registers寄存器** are high-speed storage areas physically located in the computer's processing unit.
- In **run-time initialization运行时初始化**, initialization occurs each time the declaration statement is encountered.
- **Scope作用域** is defined as the section of the program where the variable is valid or "known."
- Functions that call themselves are referred to **as self-referential自引用** or **recursive递归** functions.
- C allocates new memory locations for all function arguments and local variables as each function is called. This allocation is made dynamically, as a program is executed, in a memory area referred to as the **stack栈**.
- Where and how long a variable's storage locations are kept before they are released can be determined by the **storage class存储类** of the variable.