Chapter 14 - Additional Capabilities

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 14 introduces several additional capabilities of the C language: the `typedef` declaration statement, the conditional preprocessor directives, the use of enumerated constants, the `?:` operator and the `goto` statement. You also learn about the bit operators available in C, how to create and use macros and how to pass and use command-line arguments. Finally, several common programming and compiler errors are reviewed.

Objectives

- Additional features
- Bit operations
- Macros
- Command-line arguments
- Common programming and compiler errors

Additional Features

Topic Tip	C++ allows a variable to be declared as type of an enumeration type. C does not allow this. See: **http://cplus.about.com/od/beginnerctutoria1/l/aa031002b.htm**.

Topic Tip	Enumerations are an alternative to using multiple `#defines`.

Quick Quiz 1

1 Both the `#ifndef` and `#ifdef` directives permit_____in that the statements immediately following these directives, up to either the `#else` or `#endif` directives, are compiled only if the condition is true, whereas the statements following the `#else` are compiled only if the condition is false.

2 Are there any ternary operators in C? If so, give an example of a ternary operator.

3 The_____statement creates a new name for an existing data type.

4 How can you create enumerated lists in C?

Bit Operations

Topic Tip	The bit operators are also referred to as bitwise operators.

Topic Tip	You can use `~0` to learn what the largest possible integer in a computer system is. See: **www.cprogramming.com/tutorial/bitwise_operators.html**.

Topic Tip	Bitwise operations are usually faster than other types of operations in a computer (**http://en.wikipedia.org/wiki/Bitwise_operation**). For this reason, if efficiency is crucial, you may consider using bit operations for division and multiplication (when possible) instead of using `/` or `*`.

Quick Quiz 2

1. What are bit operators?

2. In an operation like `op1 & op2`, the 0s in `op2` effectively mask, or eliminate, the respective bits in `op1`, while the ones in `op2`_____the respective bits in `op1` through with no change in their values.

3. In a(n)_____right shift (using the >> operator), each single shift to the right corresponds to a division by 2.

4. How does the ^ operator work?

Macros

Topic Tip	For more information on how the C preprocessor works, see **http://en.wikipedia.org/wiki/C_preprocessor**.

Command-Line Arguments

Topic Tip	Note that if the full path name of the program is stored, `pgm14.3` in Figure 14.11 should be replaced by its full path name.

Topic Tip	Note that if the full path name of the program is stored, the first character displayed is the disk drive designation, which is usually C.

Quick Quiz 3

1. What is a macro?

2. What are command-line arguments?

3. The advantage of using a(n)_____instead of a function is an increase in execution speed.

4. Any argument typed on a command line is considered to be a(n) _____.

Additional Resources

1. C Preprocessor:
   http://en.wikipedia.org/wiki/C_preprocessor

2. C Tutorial - Constants in C/C++:
   https://www.geeksforgeeks.org/constants-in-c-cpp/

3. Bitwise Operators in C and C++:
   www.cprogramming.com/tutorial/bitwise_operators.html

4. C Tutorial - Command line arguments in C/C++:
https://www.geeksforgeeks.org/command-line-arguments-in-c-cpp/

Key Terms

- The `typedef` declaration statement permits constructing alternate names for an existing C data type name. These alternate names are known as **aliases别名**.
- In an **arithmetic right shift 算术右移**(using the >> operator), each single shift to the right corresponds to a division by 2.
- The operators that are used to perform bit operations in C are known as **bit operators位运算符**.
- **Command-line arguments命令行实参** are arguments that are typed on the command line.
- Both the `#ifndef` and `#ifdef` directives permit **conditional compilation条件编译** in that the statements immediately following these directives, up to either the `#else` or `#endif` directives, are compiled only if the condition is true, whereas the statements following the `#else` are compiled only if the condition is false.
- In an operation like `op1 & op2`, the 0s in `op2` effectively mask, or eliminate, the respective bits in `op1`, while the ones in `op2` **filter过滤器**, or pass, the respective bits in `op1` through with no change in their values.
- For positive signed numbers, where the leftmost bit is 0, both arithmetic and **logical right shifts逻辑右移** produce the same result.
- When the equivalence created using a `#define` statement consists of more than a single value, operator or variable, the symbolic name is referred to as a **macro宏命令**, and the substitution of the text in place of the symbolic name is called a **macro expansion宏扩展** or **macro substitution宏替换**.
- In an operation like `op1 & op2`, the variable `op2` is called a **mask掩码**.
- `AND` operations are extremely useful in **masking掩蔽**, or eliminating, selected bits from an operand.
- The conditional operator, `?:`, is unique in C in that it is a **ternary operator三元运算符**.