CLEVELAND INSTITUTE OF ELECTRONICS

A FIRST BOOK OF ANSI C, FOURTH EDITION
LESSONS 9201C TO 9210C

Study Guide to accompany *A First Book of ANSI C, Fourth Edition*

By Gary J. Bronson
ISBN: 1418835560
ISBN 13: 9781418835569

This study guide contains excerpts from the A First Book of ANSI C Instructor's resources and the book A First Book of ANSI C by Gary Bronson

FIRST EDITION / First Printing / March 2012

Contents

Chat with Your Instructor

The Study Guide for *A First Book of ANSI C, Fourth Edition*, is created to provide you with concepts, ideas and pointers for learning about computer programming in C. Chapter outlines are included followed by topic ideas presented in a chapter-by-chapter format, which include the following:

- Quick Quizzes
- Additional Resources
- Key Terms definitions

The topics follow the section-by-section format of the book and are intended to give you a range of ideas for your understanding.

Students often learn from investigation of programming topics. For this reason, there are many topic tips throughout this guide. These topic tips may require you to solve a problem, work through a new direction in technology or even attempt to predict the future. All of the topic tips are meant to be thought provoking and to help you apply what you are learning.

Each chapter in the Study Guide also includes a set of interesting Additional Resources, which are Web links to topics of interest. Finally, a list of definitions of Key Terms is included for each chapter.

A First Book of ANSI C, Fourth Edition, covers many exciting topics, and the staff believes that you will share in the excitement.

If you have a technical problem, we recommend the following:

- First, check the textbook that accompanies the software.
- Many software products include on-line help. If the answer is not available in the printed materials, try using the Help feature of your software.
- Feel free to call the instruction department during business hours (8:30 AM to 6 PM Eastern time), Monday through Friday, and Saturday during the weekend hours (8:30 AM to 5 PM Eastern time). Be prepared to describe which lesson you're working on and the problem you're having.

Instructional Support Addresses and Phone Numbers
Main Support Help Line: (800) 243-6446 or (216) 781-9400
E-mail address: faculty@cie-wc.edu
Instructional Support is available business hours (Eastern time) Monday through Saturday.
Mailing address: Cleveland Institute of Electronics
1776 East 17th Street
Cleveland, OH 44114

Chapter 1 - Introduction to Computer Programming

LESSON 9201C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 1 provides an introduction to computer programming. You will learn about the history of computers and about computer hardware. You will also learn about computer languages, how to classify them, and how executable programs are generated. Students will learn about algorithms and about the software development process. Through a case study, you will put to practice some of the concepts learned in the chapter. Finally, you learn about common programming errors and how to avoid them.

Objectives

- History and hardware
- Programming languages
- Algorithms
- The software development process
- Case study: Design and development
- Common programming errors

Topic Tips

History and Hardware

Topic Tip	The historical note on page 8 introduces Turing Machine. You may note that Turing's contributions to the field of computer science were so important that the highest award in the field of computing was named after him. For more information, see: **http://en.wikipedia.org/wiki/Turing_Award**.

Quick Quiz 1

1. The smallest and most basic data item in a computer is a _____; it is really a switch that can be either open (0) or closed (1).

2. What is the ALU of a computer?

3. What is the control unit of a computer?

4. A(n) _____ allows a computer to read or write any one file or program independent of its position on the storage medium.

Programming Languages

Topic Tip	For an example of the instruction set of an assembly language, see **http://home.comcast.net/~fbui/intel.html**.

Topic Tip	More on the history of algorithms can be found at: **http://en.wikipedia.org/wiki/Algorithm**.

Quick Quiz 2

1. What is an assembly language?

2. The program that translates a high-level source program as a complete unit before any individual statement is executed is called a(n) _____.

3. What is a linker?

4. When English phrases are used to describe an algorithm (the processing steps), the description is called _____.

The Software Development Process

Topic Tip	See the footnote on page 29 for a very interesting explanation on why the term *bug* is used to refer to program errors.

Quick Quiz 3

1. What is the software development process?

2. When writing a program, a(n) _____ structure provides the capability to make a choice between different instructions, depending on the result of some condition.

3. When writing a program, a(n) _____ structure involves summoning into action specific sections of code as they are needed.

4. What is a repetition structure?

Additional Resources

1. History of Computing Hardware:
   http://en.wikipedia.org/wiki/History_of_computing_hardware

2. C Programming Language:
   http://en.wikipedia.org/wiki/C_programming_language

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

4. Software Development Process:
   http://en.wikipedia.org/wiki/Software_development_process

Key Terms

- **Application software** consists of programs written to perform particular tasks required by users.
- The **Arithmetic and Logic Unit (ALU)** of a computer performs all of the computations, such as addition, subtraction, comparisons, and so on, that a computer provides.
- Translator programs that translate assembly language programs into machine language programs are known as **assemblers**.
- Programming languages that use the substitution of word-like symbols, such as ADD, SUB, MUL, for the binary opcodes, and both decimal numbers and labels for memory addresses are referred to as **assembly languages**.
- The smallest and most basic data item in a computer is a **bit**; it is really a switch that can be either open (0) or closed (1).

- The **bootstrap loader** is internally contained in ROM and is a permanent, automatically executed component of the computer's system software.
- The grouping of 8 bits to form a larger unit is an almost universal computer standard and is referred to as a **byte**.
- The collections of patterns consisting of 0s and 1s used to represent letters, single digits, and other single characters are called **character codes**.
- Converting an algorithm into a computer program, using a language such as C, is called **coding the algorithm**.
- When all of the statements in a high-level source program are translated as a complete unit before any individual statement is executed, the programming language is called a **compiled language**.
- The program that translates a high-level source program as a complete unit before any individual statement is executed is called a **compiler**.
- A **computer program** is a structured combination of data and instructions that is used to operate a computer and produce a specific result.
- The **control unit** of a computer directs and monitors the overall operation of the computer.
- A **direct access storage device (DASD)** allows a computer to read or write any one file or program independent of its position on the storage medium.
- An **executable program** is a program that can operate a computer.
- A **first-level structure diagram** for an algorithm represents the first attempt at an initial, but not yet sufficiently detailed, structure for a solution algorithm.
- Initially, the most common magnetic disk storage device was the removable **floppy disk**.
- A **flowchart** provides a pictorial representation of an algorithm using specifically defined shapes.
- When mathematical equations are used to describe an algorithm, the description is called a **formula**.
- In C, a procedure is referred to as a **function**.
- Collectively, the components used to make a computer are referred to as **hardware**.
- When each statement in a high-level source program is translated individually and executed immediately upon translation, the programming language is called an **interpreted language**.
- The program that translates each statement in a high-level source program and executes it immediately upon translation is called an **interpreter**.
- When writing a program, an **invocation** structure involves invoking, or summoning into action, specific sections of code as they are needed.
- A **linker** combines additional machine language code with the object program to create a final executable program.
- Both machine and assembly languages are classified as **low-level languages**; this is because both of these language types use instructions that are directly tied to one type of computer.
- Executable programs are always written as a sequence of binary numbers, which is a computer's internal language, and are also referred to as **machine language programs**.
- A **magnetic hard disk** consists of either a single rigid platter or several platters that spin together on a common spindle.
- In Java, a procedure is referred to as a **method**.
- CPUs are constructed as a single microchip, which is referred to as a **microprocessor**.

- Operating systems that permit each user to run multiple programs are referred to as both **multiprogrammed** and **multitasking** systems.
- **Multiuser systems** are able to handle multiple users concurrently.
- The output produced by the compiler is called an **object program**, which is a machine language version of the source code.
- Languages with object orientation like C++, Java, Visual Basic, and C#, are known as **object-oriented languages**.
- **Opcode** is short for operation code.
- Collectively, the set of system programs used to operate and control a computer is called the **operating system**.
- The three tasks of an overall solution algorithm are the primary responsibility of almost every problem, and we refer to this algorithm as the **Problem-Solver Algorithm**.
- In a **procedural language**, the available instructions are used to create self-contained units, referred to as procedures.
- The purpose of a **procedure** is to accept data as input and transform the data in some manner to produce a specific result as an output.
- The program instructions resulting from coding an algorithm are referred to as **program code**, or simply **code**, for short.
- In an automobile, control is provided by the driver, who sits inside of and directs the car; in a computer, the driver is called a **program**.
- A statement of a problem, or a specific request for a program, is referred to as a **program requirement**.
- The set of instructions that can be used to construct a program is called a **programming language**.
- **Programming** is the process of writing instructions in a language that the computer can respond to and that other programmers can understand.
- When English phrases are used to describe an algorithm (the processing steps), the description is called **pseudocode**.
- When writing a program, a **repetition** structure, which is also referred to as **looping** and **iteration**, provides the ability for the same operation to be repeated based on the value of a condition.
- Each field of study has its own name for the systematic method used to design solutions to problems. In science this method is referred to as the **scientific method**, while in engineering disciplines the method is referred to as the **systems approach**.
- When writing a program, a **selection** structure provides the capability to make a choice between different instructions, depending on the result of some condition.
- When writing a program, a s**equence** structure defines the order in which instructions are executed by the program.
- Another term for a program or set of programs is **software**.
- The technique used by professional software developers for understanding the problem that is being solved and for creating an effective and appropriate software solution is called the **software development process**.
- Programs written in a computer language (high or low level) are referred to interchangeably as both **source programs** and **source code**.
- A **structured language** is a high-level procedural language, such as C, that enforces structured procedures.
- Procedures conforming to structure guidelines are known as **structured procedures**.
- **System software** is the collection of programs that must be readily available to any computer system to enable the computer to operate.

- A **top-down algorithm development** starts at the topmost level and proceeds to develop more and more detailed algorithms as it proceeds to the final set of algorithms.
- Main memory is **volatile**; whatever is stored in it is lost when the computer's power is turned off.
- Main memories combine 1 or more bytes into a single unit, referred to as a **word**.

Chapter 2 - Getting Started in C Programming

At a Glance

Lesson Contents

- Overview
- Objectives
- Quick Quizzes
- Additional Resources
- Key Terms
- Lesson Assignment

Chapter Notes

Overview

Chapter 2 provides an introduction to C programming. You learn about good programming style and about the data types available in C. You also learn about arithmetic operations, and how to declare and initialize variables. You put to practice the concepts learned through a case study on temperature conversion. Finally, you learn about some common programming and compiler errors, and how to avoid them.

Objectives

- Introduction to C programming
- Programming style
- Data types
- Arithmetic operations
- Variables and declarations
- Case Study: Temperature conversion
- Common programming and compiler errors

Topic Tips

Introduction to C Programming

Topic Tip	In a computer language, a *token* is the smallest unit of the language that has a unique meaning. Thus, the reserved words, programmer-defined identifiers, and all special mathematical symbols, such as + and -, are considered tokens of the C language.

Topic Tip	What if you want to include the backslash character in a string? Just precede it with another backslash, as "\\".

Programming Style

Topic Tip	For a brief overview of different indentation styles, see **http://en.wikipedia.org/wiki/Indent_style**.

Quick Quiz 1

1. What is an identifier?
2. What is a function header line?
3. The two characters \ and n, when used together, are called a(n) _____.
4. A(n) _____ is a word that is predefined by the programming language for a special purpose and can only be used in a specified manner for its intended purpose.

Data Types

Topic Tip	For a discussion on the meaning of the term *precision*, see the box on page 64.

Quick Quiz 2

1. What is a data type?

2. In numerical theory, the term _____ typically refers to numerical accuracy.

3. What is an expression?

4. _____ is the order in which operators of the same precedence are evaluated.

Variables and Declarations

Topic Tip	Several tips on how to select variable names have been provided throughout this chapter. Try to describe the ones you can remember and write the list on a sheet of paper. Next, add any other tips missing from the list (see the Programming Note on page 82).

Quick Quiz 3

1. _____ are simply names given by programmers to computer storage locations.

2. What is an assignment statement?

3. What is a definition statement?

4. When a declaration statement provides an initial value, the variable is said to be _____.

Additional Resources

1. Indent Style:
   http://en.wikipedia.org/wiki/Indent_style

2. C/C++ Data Types:
   http://www.cppreference.com/data_types.html

3. C Tutorial - Lesson 2: Variables:
   http://cplus.about.com/od/beginnerctutoria1/l/aa030302a.htm

4. Operator Precedence:
   http://computer.howstuffworks.com/c37.htm

Key Terms

- Items passed to a function are always placed within the function name parentheses and are called **arguments**.
- Data transmitted into a function at run time are referred to as **arguments of the function**.
- The operators used for arithmetic operations are called **arithmetic operators**.
- An **assignment statement** tells the computer to assign a value to (that is, store a value in) a variable.
- **Associativity** is the order in which operators of the same precedence are evaluated.
- **Binary operators** require two operands to produce a result.
- A **built-in data type** is one that is provided as an integral part of the language.
- A **comment** is a note about the code that the programmer includes so that he (or other programmers) can keep track of what the various parts of the program do.
- A control string is referred to as a **control specifier**.
- A string that also includes a **conversion control sequence**, such as `%d`, is termed a **control string**.
- Conversion control sequences are also referred to as **conversion specifications** and **format specifiers**.
- A **data type** is defined as a set of values *and* a set of operations that can be applied to these values.
- **Definition statements** define or tell the compiler how much memory is needed for data storage.
- A `double` value is sometimes referred to as a **double-precision** number.
- The `main()` function is sometimes referred to as a **driver function**, because it tells the other functions the sequence in which they are to operate.
- The backslash character, \, is also known as the **escape character**.
- The combination of a backslash and one of several specific characters is called an **escape sequence**.
- All statements that cause some specific action to be performed by the computer when the function is executed must end with a semicolon (;); such statements are known as **executable statements**.
- An **expression** is any combination of operators and operands that can be evaluated to yield a value.
- An expression containing only floating-point values (single and double precision) as operands is called a **floating-point expression** (the term **real expression** is also used), and the result of such an expression is a double-precision value.
- A **floating-point value**, which is also called a **real number**, can be the number zero or any positive or negative number that contains a decimal point.
- A **function header line**, which is always the first line of a function, contains three pieces of information: (1) what type of data, if any, is returned by the function, (2) the name of the function, and (3) what type of data, if any, is sent into the function.
- A **header file** is placed at the top, or head, of a C program using the `#include` command.
- The names of functions, as well as all of the words permitted in a program that have special meaning to the compiler, such as `radius` and `circumference`, are collectively referred to as **identifiers**.
- Declaration statements can also be used to store an initial value into declared variables; this value is referred to as an **initial value**.

- When a declaration statement provides an initial value, the variable is said to be **initialized**.
- **Invoking** a function is more commonly referred to as **calling the function**.
- Reserved words are also referred to as **keywords** in C.
- A **literal** is an acceptable value for a data type.
- Another name for a literal is a **literal value**, or **constant**.
- An expression containing both integer and floating-point values is called a **mixed-mode expression**.
- The % operator, called both the **modulus** and **remainder operator**, captures the remainder when an integer number is divided by an integer.
- Under no circumstances may comments be **nested**—one comment containing another comment.
- The two characters \ and n, when used together, are called a **newline escape sequence**.
- An **operand** can be either a literal value or an identifier that has a value associated with it.
- Inputting data or messages to a function is called **passing data to the function**.
- In numerical theory, the term **precision** typically refers to numerical accuracy.
- Built-in types are also known as **primitive types**.
- A large number of the identifiers used in a C program are selected by the programmer, and are known as **programmer-created identifiers** or **programmer-created names**.
- The keywords `short`, `long`, and `unsigned` are known as **qualifiers**, because they qualify the meaning of the keyword `int`.
- A **reserved word** is a word that is predefined by the programming language for a special purpose and can only be used in a specified manner for its intended purpose.
- `short int`, `int`, and `long int` data types are formally referred to as **signed data types**.
- A **simple binary arithmetic expression** consists of a binary arithmetic operator connecting two literal values in the form: `literalValue` *`operator`* `literalValue`.
- A `float` value is sometimes referred to as a **single-precision** number.
- **Standard identifiers** are words that are predefined in C.
- Messages are known as **strings** in C, because they consist of a string of characters made up of letters, numbers, and special characters.
- A programming language's **syntax** is the set of rules for formulating statements that are "grammatically correct" for the language.
- A **unary operator** is one that operates on a single operand.
- An **unsigned data type** provides only for nonnegative (that is, zero and positive) values.
- The address of the first memory byte used for storing a variable is known as the **variable's address**.
- **Variables** are simply names given by programmers to computer storage locations.
- In C, **white space** refers to any combination of one or more blank spaces, tabs, or new lines.

ANSI C Lesson 9201C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

1. A(n) ____ is a word that is predefined by the programming language for a special purpose and can only be used in a specified manner for its intended purpose.
   1) variable
   2) identifier
   3) reserved word
   4) data type

2. Main memories combine 1 or more bytes into a single unit, referred to as a(n) ____.
   1) bit
   2) character
   3) opcode
   4) word

3. ____ identifiers are words that are predefined in C.
   1) Standard
   2) Programmer-created
   3) Reserved
   4) Primitive

4. The collections of patterns consisting of 0s and 1s used to represent letters, single digits, and other single characters are called ____.
   1) bytes
   2) character codes
   3) words
   4) opcodes

5. The names of functions, as well as all of the words that are permitted in a program, that have special meaning to the compiler are collectively referred to as ____.
   1) variables
   2) identifiers
   3) reserved words
   4) keywords

6. Messages are known as ____ in C.
   1) characters
   2) text
   3) banners
   4) strings

7. A ____ is placed at the top of a C program using the `#include` command.
   1) header file
   2) `main()` function
   3) return statement
   4) data type

8. A repetition structure is also known as a(n) ____ structure.
   1) sequence
   2) selection
   3) looping
   4) invocation

9. An expression containing only floating-point values as operands is called a floating-point expression, and the result of such an expression is a(n) ____ value.
   1) single-precision
   2) double-precision
   3) integer
   4) long integer

10. Built-in types are also known as ____.
    1) data types
    2) primitive types
    3) literals
    4) basic

11. A(n) ____ is any combination of operators and operands that can be evaluated to yield a value.
    1) expression
    2) statement
    3) operation
    4) argument

12. The grouping of 8 bits to form a larger unit is an almost universal computer standard and is referred to as a ____.
    1) byte
    2) character
    3) word
    4) opcode

13. A(n) ____ is an acceptable value for a data type.
    1) identifier
    2) variable
    3) escape sequence
    4) literal

14. A ____ value is sometimes referred to as a single-precision number.
    1) `float`
    2) `double`
    3) `int`
    4) `short int`

15. The program that translates a high-level source program as a complete unit before any individual statement is executed is called a(n) ____.
    1) interpreter
    2) assembler
    3) compiler
    4) linker

16. ____ is the order in which operators of the same precedence are evaluated.
    1) Associativity
    2) Priority
    3) Syntax
    4) Precision

17. A(n) ____ value can be the number zero or any positive or negative number that contains a decimal point.
    1) integer
    2) floating-point
    3) boolean
    4) character

18. Translator programs that translate assembly language programs into machine language programs are known as ____.
    1) assemblers
    2) linkers
    3) compilers
    4) interpreters

19. When writing a program, a(n) ____ structure defines the order in which instructions are executed by the program.
    1) sequence
    2) selection
    3) iteration
    4) invocation

20. A(n) ____ is defined as a set of values and a set of operations that can be applied to these values.
    1) variable
    2) identifier
    3) data type
    4) literal

END OF EXAM

Chapter 3 - Processing and Interactive Input

LESSON 9202C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 3 provides an introduction to processing and interactive input in C. You learn how assignment statements work and how to use mathematical library functions. You learn to use the `scanf()` function for interactive input, and how to accomplish complex formatted output. In this chapter, you also learn how to create and use symbolic constants. The chapter case study has you practice creating a program with interactive input. Finally, the common programming and compiler errors related to this chapter are reviewed.

Objectives

- Assignment
- Mathematical library functions
- Interactive input
- Formatted output

- Symbolic constants
- Case study: Interactive input
- Common programming and compiler errors

Topic Tips

Assignment

Topic Tip	Explore the meaning of the terms *lvalue* and *rvalue*, as described in the Programming Note on page 107.

Mathematical Library Functions

Topic Tip	Besides `math.h`, ANSI C provides other standard library header files. For more information, see **http://en.wikipedia.org/wiki/C_standard_library**.

Quick Quiz 1

1. In C, the _____ symbol is called the assignment operator.

2. The automatic conversion across an assignment operator is referred to as a(n) _____ type conversion.

3. What is a garbage value?

4. What is the prefix increment operator?

Interactive Input

Topic Tip	Note that there are two other solutions for the problem described above: 1) Replace the last `scanf()` call in Program 3.11 with the statement `scanf("\n%c",&skey);`, 2) Place the statement `fflush(stdin);` after accepting a one-character input. The `fflush()` function flushes the input buffer of any remaining characters.

Quick Quiz 2

1. A(n) _____ is a message that tells the person at the screen what should be typed.

2. On most computer systems, characters read by the keyboard are stored in a temporary holding area called a(n) _____ immediately after they are pressed.

3. What are robust programs?

4. What is user-input validation?

Formatted Output

Topic Tip	The `#define` preprocessor directive can also be used to create useful macros. For more information, see: **http://en.wikipedia.org/wiki/C_preprocessor**.

Quick Quiz 3

1. The format of numbers displayed by `printf()` can be controlled by _____ included as part of each conversion control sequence.

2. What are magic numbers?

3. `#define` statements are also called _____ statements.

4. What does the term "literal data" mean?

Additional Resources

1. `math.h`:
   www.cplusplus.com/ref/cmath/

2. `scanf`:
   www.cplusplus.com/ref/cstdio/scanf.html

3. C Tutorial - Lesson 3: Constants:
   http://cplus.about.com/od/beginnerctutoria1/l/aa031002a.htm

4. C Macros:
   http://en.wikipedia.org/wiki/C_preprocessor

Key Terms

- User-defined data types are formally referred to as **abstract data types**.
- In C, the = symbol is called the **assignment operator**.
- On most computer systems, characters read by the keyboard are stored in a temporary holding area called a **buffer** immediately after they are pressed.
- The operator used to force the conversion of a value to another type is the **cast** operator.
- A special type of assignment statement that is very similar to the accumulating statement is the **counting statement**.
- `#define` statements are also called **equivalence** statements.
- The format of numbers displayed by `printf()` can be controlled by **field width specifiers** included as part of each conversion control sequence.
- A previously stored number, if it has not been initialized to a specific and known value, is frequently referred to as a **garbage value**.
- The automatic conversion across an assignment operator is referred to as an **implicit type conversion**.
- Using the **increment operator**, ++, the expression `variable = variable + 1` can be replaced by the either the expression `variable++` or `++variable`.
- **Literal data** refers to any data within a program that explicitly identifies itself.
- The term **lvalue** refers to any quantity that is valid on the left side of an assignment operator.
- Literal values that appear many times in the same program are referred to by programmers as **magic numbers**.
- When the -- operator appears after a variable, it is called a **postfix decrement operator**.
- When the ++ operator appears after a variable, it is called a **postfix increment operator**.
- When the -- operator appears before a variable, it is called a **prefix decrement operator**.
- When the ++ operator appears before a variable, it is called a **prefix increment operator**.
- The assigning of a name to a function or procedure in such a way that the function is invoked by simply using a name with appropriate arguments is formally referred to as **procedural abstraction**.
- A **prompt** is a message that tells the person at the screen what should be typed.
- Programs that detect and respond effectively to unexpected user input are formally referred to as **robust** programs and informally as "bullet-proof" programs.
- An **rvalue** refers to any quantity that is valid on the right side of an assignment operator.
- Other terms for symbolic names are **symbolic constants** and **named constants**.
- C provides the programmer with the capability to define a value (that will be used throughout a program) once by equating the number to a **symbolic name**.
- The basic approach to handling invalid data input is referred to as **user-input validation**, which means validating the entered data either during or immediately after the data have been entered, and then providing the user with a way of reentering any invalid data.
- The term **validate** means checking that the entered value matches the data type of the variable that the value is assigned to within a `scanf()` function call, and that the value is within an acceptable range of values appropriate to the application.

ANSI C Lesson 9202C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice

Identify the choice that best completes the statement or answers the question.

1. ____ statements are also called equivalence statements.
   1) Assignment
   2) Prompt
   3) `#include`
   4) `#define`
2. Programs that detect and respond effectively to unexpected user input are formally referred to as ____ programs.
   1) standard
   2) robust
   3) cost-effective
   4) abstract
3. If you are using a Unix or Linux operating system, you must include the ____ option when compiling a C program that uses the mathematical functions like `log` and `sqrt`.
   1) `-lm`
   2) `-l math.h`
   3) `-math`
   4) `-lib = math`
4. The ____ sign is a signal to a C preprocessor.
   1) `!`
   2) `&`
   3) `;`
   4) `#`
5. To access the mathematical functions such as `log` and `sqrt`, you must include the following preprocessor statement in your program: ____.
   1) `#define <mathematical.h>`
   2) `#include <mathematical.h>`
   3) `#define <math.h>`
   4) `#include <math.h>`
6. The term ____ refers to any quantity that is valid on the left side of an assignment operator.
   1) leftv
   2) lval
   3) variable
   4) lvalue
7. The C function ____ yields the result of a value raised to the power of another value.
   1) `pow`
   2) `exp`
   3) `abs`
   4) `sqrt`
8. On most computer systems characters read by the keyboard are stored in a temporary holding area called a ____ immediately after they are pressed.
   1) register
   2) buffer
   3) stack
   4) RAM
9. The statement ____ shows an implicit conversion.
   1) `int total = (int) sum;`
   2) `double avg = 0.0;`
   3) `float price = 9.90f;`
   4) `int answer = 2.745;`
10. Which of the following statements about rvalues and lvalues is NOT true?
    1) Any expression that yields a value can be an rvalue.
    2) A variable declared for an array cannot be an rvalue, but individual array variables can be.
    3) A variable declared for an array can be an lvalue.

4) Individual numbers can only be an rvalue.

11. The expression `sum = sum + 10` can be written as ____.
    1) `sum =+ 10`
    2) `sum += 10`
    3) `sum = sum ++ 10`
    4) `sum ++ 10`
12. ____ is a valid statement in C.
    1) `a = 10 = c = 25;`
    2) `a = b = c = 25;`
    3) `2 = b;`
    4) `a - 1 = c;`
13. The operator used to force the conversion of a value to another type is the ____ operator.
    1) conversion
    2) cast
    3) assignment
    4) increment
14. The ____ function requires a control string as the first argument inside the function name parentheses.
    1) `sqrt()`
    2) `pow()`
    3) `scanf()`
    4) `log()`
15. Literal values that appear many times in the same program are referred to by programmers as ____ numbers.
    1) symbolic
    2) magic
    3) constant
    4) literal
16. The increment operator is ____.
    1) +=
    2) =+
    3) ++
    4) --
17. In C, the ____ symbol is called the assignment operator.
    1) =
    2) ++
    3) --
    4) ()
18. The cast operator has the syntax ____.
    1) `(dataType expression)`
    2) `(expression dataType)`
    3) `(dataType) expression`
    4) `expression (dataType)`
19. The conversion control sequence ____ would cause an integer number to both display its sign and be left-justified in a field width of 10 spaces.
    1) `%-+10d`
    2) `%-10d`
    3) `%+10d`
    4) `%*10d`
20. A previously stored number, if it has not been initialized to a specific and known value, is frequently referred to as a ____.
    1) garbage value
    2) literal
    3) bogus value
    4) buffer
21. A(n) ____ is a message that tells the person at the screen what should be typed.
    1) prompt
    2) input statement
    3) `scanf`
    4) `printf`
22. Format modifiers, if used, must always be placed immediately after the ____ symbol.
    1) !
    2) =
    3) %
    4) .
23. When the ++ operator appears before a variable, it is called a ____ increment operator.
    1) basic
    2) standard
    3) postfix
    4) prefix
24. The expression `price *= rate + 1` is equivalent to the expression ____.
    1) `price = price * (rate + 1)`
    3) `price = (price * rate) + 1`

2) `price = price * rate + 1` 4) `price = price ^ (rate + 1)`

25. Formatted floating-point numbers require ____ field width specifier(s).
    1) one
    2) two
    3) three
    4) four

END OF EXAM

Chapter 4 - Selection

LESSON 9203C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 4 introduces the selection structures available in C. You learn to create and evaluate relational expressions and to use `if` and `if-else` statements. You also learn to use the `switch` statement. In the case study you learn how to use selection structures for data validation. Finally, you learn to identify and avoid common programming and compiler errors.

Objectives

- Relational expressions
- The `if` and `if-else` statements
- The `if-else` chain
- The `switch` statement
- Case study: Data validation
- Common programming and compiler errors

Relational Expressions

Topic Tip	Read the section about De Morgan's laws, as they are useful when writing relational expressions. For more information, see the Historical Note on page 161.

Quick Quiz 1

1. What is "flow of control"?

2. A(n) _____ expression consists of a relational operator that compares two operands.

3. Relational expressions are also known as _____.

4. How does the NOT (!) operator work?

The `if` and `if-else` Statements

Topic Tip	Beginner programmers tend to write a single = instead of == in an `if` condition. This leads to unexpected errors that are not detected by the compiler, because the expression is valid in C. For more information, see the Programming Note on page 169.

Topic Tip	A way to avoid the common problem of accidentally using = instead of == in an `if` condition is to code the expression with the constant to the left of the relational operator (see the Programming Note on page175).

Topic Tip	C also provides a shortcut to the `if-else` operator: the ternary conditional operator (`?:`). This operator is very useful, as it allows you to write code like: `int max = (a > b) ? a : b`.

Quick Quiz 2

1. The simplest C selection statement is the _____ `if` statement.

2. A(n) _____ statement is one or more statements contained between braces.

3. What is a nested `if` statement?

4. Is indentation important in the evaluation of `if-else` statements?

The `switch` Statement

Topic Tip	Do you have previous programming experience with Pascal? If so, note that the equivalent to the `switch` statement is the `case` statement in Pascal.

Quick Quiz 3

1. What is a `switch` statement?

2. Internal to the `switch` statement, the keyword _____ identifies the values that will be compared to the value of the `switch` expression.

3. What is the role of the `default` statement in a `switch`?

4. Once an entry point has been located by the `switch` statement, no further `case` evaluations are done; this means that unless a(n) _____ statement is encountered, all statements that follow, until the closing brace of the `switch` statement, will be executed.

Additional Resources

5. C Tutorial - Lesson 5: Conditional Processing, Part 1: If/Else Statements and Relational Operators:
   http://cplus.about.com/od/beginnerctutoria1/l/aa040202a.htm

6. The Use of Braces in C/C++:
   http://cplus.about.com/od/cprogrammingtip1/l/aa010102a.htm

7. C Ternary Operator:
   http://cplus.about.com/od/cprogrammin1/l/bldef_ternaryop.htm

8. C Tutorial - Lesson 6: Conditional Processing, Part 2: Switch Statements and Logical Operators:
   http://cplus.about.com/od/beginnerctutoria1/l/aa040302a.htm

Key Terms

- A **compound statement** is one or more statements contained between braces.
- Relational expressions are also known as **conditions**.
- A **debugger** program controls the execution of a C program, can interrupt the C program at any point in its execution, and can display the values of all variables at the point of interruption.
- In computer jargon, a program error is referred to as a **bug**, and the process of isolating, correcting, and verifying the correction is called **debugging**.
- **Defensive programming** is a technique where the program includes code to check for improper data before an attempt is made to process it further.
- **Diagnostic `printf()` statements** can be a considerable help in debugging.
- **Echo printing** is the technique to add temporary code that displays the values of all input data.
- The term **flow of control** refers to the order in which a program's statements are executed.
- A nested `if` construction, in which each nested `if` is written in the same line as the previous `else`, is called an **`if-else` chain**, and is used extensively in many programming problems.
- The defensive programming technique of checking user input data for erroneous or unreasonable data is referred to as **input data validation**.
- Including one or more `if-else` statements within an `if` or `if-else` statement is referred to as a **nested `if` statement**.
- The simplest C selection statement is the **one-way `if` statement**.
- **Program tracing** is the technique to imitate the computer and execute each statement by hand, as the computer would.
- A **relational expression** consists of a relational operator that compares two operands.
- **Short-circuit evaluation** is the feature for the && and || operators that makes the evaluation of an expression stop as soon as it is determined that an expression is false.

Chapter 5 - Repetition

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 5 introduces the use of repetition statements in C. You learn about the basic loop structures and how to use `while` and `for` statements. Computing sums and averages using a `while` loop is studied in detail. Some case studies are explored that introduce you to several loop programming techniques. You also learn about nested loops and how to use the `do-while` statement. Finally, some common programming and compiler errors are reviewed.

Objectives

- Basic loop structures
- The `while` statement
- Computing sums and averages using a `while` loop
- The `for` statement
- Case studies: Loop programming techniques
- Nested loops
- The `do-while` statement

- Common programming and compiler errors

Basic Loop Structures

Topic Tip	The Programming Note in page 220 explains how to control a loop with a symbolic constant.

Quick Quiz 1

1. What is a loop?

2. In a(n) _____ loop, which is also known as a fixed-count loop, the condition is used to keep track of the number of repetitions that have occurred.

3. What is a program loop?

4. A(n) _____ loop is a condition-controlled loop where one specific value is required to terminate the loop.

Computing Sums and Averages Using a `while` Loop

Topic Tip	Note that sentinels are sometimes called flags or flag values. For more information, see: **http://en.wikipedia.org/wiki/Sentinel_value**.

The `for` Statement

Topic Tip	The Programming Note on page 242 discusses whether a programmer should use a `for` loop or a `while` loop.

Topic Tip	Remember that there are different styles of writing braces in C programs. Stress that which style is chosen is not that important, as long as the style is consistent throughout a program (or group of programs). For more information, see the Programming Note on page 244.

Quick Quiz 2

1. In computer programming, data values used to signal either the start or end of a data series are called _____.

2. On IBM-compatible computers, the EOF mark is generated whenever the _____ keys are pressed simultaneously.

3. How does a `break` statement work?

4. What happens if you omit the tested expression in a `for` loop?

The `do-while` Statement

Quick Quiz 3

1. What is a nested loop?

2. The second loop of a nested loop is called the _____ loop.

3. A(n) _____ statement always creates a posttest loop.

4. What type of application is ideally suited for a posttest loop?

Additional Resources

1. C Tutorial - Lesson 7: Looping:
   http://cplus.about.com/od/beginnerctutoria1/l/aa040402a.htm

2. While Loop:
   http://en.wikipedia.org/wiki/While_loop

3. For Loop:
   http://en.wikipedia.org/wiki/For_loop

4. Do While Loop:
   http://en.wikipedia.org/wiki/Do_while_loop

5. How C Programming Works: Branching and Looping:
   http://computer.howstuffworks.com/c8.htm

Key Terms

- Lists in C, where commas are required to separate individual expressions in the list, are referred to as **comma-separated** lists.
- In a **condition-controlled loop**, the tested condition does not depend on a count being achieved, but rather on a specific value being encountered.

- In a **counter-controlled loop**, which is also known as a **fixed-count loop**, the condition is used to keep track of the number of repetitions that have occurred.
- Pretest loops are also referred to as **entrance-controlled loops**.
- The second loop of a nested loop is called the **inner loop**.
- The **input data validation application** is ideally suited for a posttest loop.
- Each repetition in a loop is referred to as an **iteration** or **pass through the loop**.
- A section of code that is repeated is referred to as a **loop**, because after the last statement in the code is executed, the program branches, or loops, back to the first statement and starts another repetition through the code.
- **Nested loops** have a loop contained within another loop.
- A **null statement** is a do-nothing statement that is used where a statement is syntactically required, but no action is called for.
- The first loop of a nested loop is called the **outer loop**.
- A loop that evaluates a condition at the end of the repeating section of code is referred to as a **posttest loop** or **exit-controlled loop**.
- This type of loop is referred to as a **pretest loop** because the condition is tested before any statements within the loop are executed.
- In computer programming, data values used to signal either the start or end of a data series are called **sentinels**.

ANSI C Lesson 9203C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the letter of the choice that best completes the statement or answers the question.

1. The ____ operator is used to change an expression to its opposite state.
1) ||
2) &&
3) !
4) %%

2. Which of the following operators has the lowest precedence?
1) !
2) *
3) &&
4) ||

3. The logical OR operator is ____.
1) ||
2) &&
3) !
4) %%

4. A(n) ____ loop is a condition-controlled loop that terminates when a value within a valid range is entered.
1) input-validation
2) sentinel-controlled
3) condition-controlled
4) counter-controlled

5. It is a good practice to terminate the last `case` in a `switch` statement with a ____.
1) `switch`
2) `break`
3) `default`
4) `case`

6. The ____ statement literally loops back on itself to recheck the expression until it evaluates to 0 (becomes false).
1) `for`
2) `switch`
3) `do-while`
4) `while`

7. Omitting the ____ expression in a `for` statement results in an infinite loop.
1) initializing
2) altering
3) tested
4) break

8. What will the following program print on screen?

```
int age = 0;
if (age = 40)
   printf("Happy Birthday!");
else
   printf("Sorry");
```

1) `Happy Birthday!`
2) `Sorry`
3) `Runtime error.`
4) Nothing; the program will not compile.

9. In Unix operating systems, the EOF mark is generated whenever the ____ keys are pressed simultaneously.

1) Ctrl and D
2) Ctrl and E
3) Ctrl and F
4) Ctrl and Z

10. In computer programming, data values used to signal either the start or end of a data series are called ____.

1) input values
2) limits
3) sentinels
4) iterators

11. In IBM-compatible computers, the EOF mark is generated whenever the ____ keys are pressed simultaneously.

1) Ctrl and D
2) Ctrl and E
3) Ctrl and F
4) Ctrl and Z

12. The use of ____ in a C program will result in a compiler error.

1) `if (age == 40)`
2) `if (40 == age)`
3) `if (age = 40)`
4) `if (40 = age)`

13. The logical AND operator is ____.

1) ||
2) &&
3) !
4) %%

14. In a `switch` statement, the word ____ is optional and operates the same as the last `else` in an `if-else` chain.

1) `if`
2) `break`
3) `case`
4) `default`

15. A ____ statement is a specialized selection statement that can be used in place of an `if-else` chain where exact equality to one or more integer constants is required.

1) `case`
2) `break`
3) `switch`
4) nested `if`

16. Which of the following operators has right to left associativity?

1) !
2) *
3) &&
4) ||

17. What will the following program print on screen?

```
int tenure = -5;
if (tenure + 5)
   printf("Congratulations!");
else
   printf("Sorry");
```

1) `Congratulations!`
2) `Sorry`
3) `Runtime error.`
4) Nothing; the program will not compile.

18. ____ is an accumulating statement.

1) `total += num;`
2) `total++;`
3) `++total;`
4) `total *= num;`

19. The second loop of a nested loop is called the ____ loop.

1) inner
2) outer
3) slave
4) conditioned

20. A(n) ____ is a condition-controlled loop where one specific value is required to terminate the loop.

1) input-validation
2) sentinel-controlled
3) condition-controlled
4) counter-controlled

END OF EXAM

Chapter 6 - Modularity Using Functions: Part I

LESSON 9204C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 6 introduces the role of C functions. You learn about functions and parameter declarations, and also how to return a value. In the case study, you put to practice the concepts learned to the particular case of calculating age norms. You also learn about several useful standard library functions. Finally, you learn to identify and avoid common programming and compiler errors.

Objectives

- Function and parameter declarations
- Returning a value
- Case study: Calculating age norms
- Standard library functions
- Common programming and compiler errors

Function and Parameter Declarations

Topic Tip	The Programming Note on page 279 provides a concise explanation of the difference between a function prototype, the calling statement and a function header line.

Topic Tip	It is important to note that in earlier versions of C, function prototypes were not required. Additionally, if a function header line omitted a return data type, the return value was, by default, implicitly declared as being of type `int`. For more information, see the Programming Note on page 280.

Topic Tip	Note the importance of providing preconditions and postconditions (see the Programming Note on page 284).

Quick Quiz 1

1. What is the difference between a called function and a calling function?
2. The items enclosed within the parentheses in a function call statement are called _____ of the function.
3. What is a pass by value?
4. The argument names in the header line of a function are known as _____.

Returning a Value

Topic Tip	A basic rule of testing states that each function should only be tested in a program in which all other functions are known to be correct (see the Programming Note on page 295).

Quick Quiz 2

1. A(n) _____ is the beginning of a final function that is used as a placeholder for the final function until the function is completed.
2. Pass by value is also referred to as _____.
3. How can you return a value from a function?
4. How do you write the prototype of a function with an empty parameter list?

Standard Library Functions

Quick Quiz 3

1. What are random numbers?

2. What are pseudorandom numbers?

3. The method for adjusting the random numbers produced by a random-number generator to reside within a specified range is called _____.

4. The standard library consists of _____ header files.

Additional Resources

1. C Tutorial - Lesson 14: Functions:
   http://cplus.about.com/od/beginnerctutoria1/l/aa051002a.htm

2. C Programming Tutorial: Lesson 4: Functions:
   www.cprogramming.com/tutorial/c/lesson4.html

3. C Standard Library:
   http://en.wikipedia.org/wiki/C_standard_library

4. C Programming Tips: Random Numbers: Using Pseudorandom Numbers:
   http://cplus.about.com/od/cprogrammingtips/l/aa041403a.htm

Key Terms

- Other terms used as synonyms for arguments are **actual arguments** and **actual parameters**.
- A fairly common procedure in child development is to establish normal ranges for height and weight as they relate to a child's age; these normal ranges are frequently referred to as **age norms**.
- The items enclosed within the parentheses in a function call statement are called **arguments** of the function.
- A function that is called or summoned into action by its reference in another function is a **called function**.
- A function that calls another function is referred to as the **calling function**.
- The purpose of a **function body** is to operate on the passed data and return, at most, one value directly back to the calling function.

- The portion of the function header that contains the function name and parameters is known as a **function declarator**, which should not be confused with a function declaration (prototype).
- The purpose of a **function header** is to identify the data type of the value returned by the function, if any, provide the function with a name, and specify the number, order, and type of values expected by the function.
- A **function prototype** declares the function to the compiler—it tells the compiler the name of the function, the data type of the value that the function will return (the keyword `void` indicates that the function will not be returning any value), and the data types of each argument that the function expects to receive when it is called.
- The argument names in the header line of a function are known as **parameters** or **formal parameters** and **formal arguments**.
- When a function simply receives copies of the values of each of the arguments and must determine where to store these values before it does anything else, this is known as a **pass by value** (or a **call by value**).
- **Pseudorandom numbers** are numbers which are not really random, but are sufficiently random for the task at hand.
- **Random numbers** are a series of numbers whose order cannot be predicted.
- The method for adjusting the random numbers produced by a random-number generator to reside within a specified range is called **scaling**.
- A **stub** is the beginning of a final function that is used as a placeholder for the final function until the function is completed.

ANSI C Lesson 9204C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the choice that best completes the statement or answers the question.

1. The purpose of a ____ is to operate on the passed data and return, at most, one value directly back to the calling function.
1) function declarator
2) prototype
3) function body
4) function header

2. Scaling a random number as an integer value between 1 and N is accomplished using the expression ____.
1) `1 + (int)rand() / N`
2) `1 + (int)rand() % N`
3) `(int)rand() / N`
4) `(int)rand() % N`

3. ____ is a prototype of a function that returns no value.
1) `void funcA();`
2) `funcA();`
3) `int funcA();`
4) `null funcA();`

4. The function ____ converts an ASCII string to an integer.
1) `string itoa(int)`
2) `double atof(string)`
3) `int atoi(string)`
4) `int toupper(int)`

5. The portion of the function header that contains the function name and parameters is known as a ____.
1) function body
2) prototype
3) function declarator
4) stub

6. A ____ is the beginning of a final function that is used as a placeholder for the final function until the function is completed.
1) function header
2) function declarator
3) prototype
4) stub

7. ___ is an example of a calling statement.

1) `float roi(int, double);`
2) `printf("%f", roi(3, amt));`
3) `float roi( int yrs, double rate);`
4) `float roi( int yrs, double rate)`

8. ____ reads the computer's internal clock time, in seconds.

1) `stime()`
2) `time(SECONDS)`
3) `time()`
4) `time(NULL)`

9. ___ is an example of a function prototype.

1) `float roi(int, double);`
2) `printf("%f", roi(3, amt));`
3) `roi(3, amt);`
4) `float roi( int yrs, double rate)`

10. A function that is called or summoned into action by its reference in another function is a ____.

1) function prototype
2) called function
3) calling function
4) function declarator

11. The function ____ returns a non-0 number if the argument is a letter or a digit; otherwise it returns a 0.

1) `int isalnum(int)`
2) `int isalpha(int)`
3) `int isdigit(int)`
4) `int isxdigit(int)`

12. To return a value, a function must use a(n) ____ statement.

1) `exit`
2) `throw`
3) `break`
4) `return`

13. When a function simply receives copies of the values of the arguments and must determine where to store these values before it does anything else, this is known as a ____.

1) pass by value
2) pass by reference
3) stub
4) function declarator

14. ___ is an example of a function header line.

1) `float roi(int, double);`
2) `printf("%f", roi(3, amt));`
3) `float roi( int yrs, double rate);`
4) `float roi( int yrs, double rate)`

15. The method for adjusting the random numbers produced by a random-number generator to reside within a specified range is called ____.

1) scaling
2) stubbing
3) prototyping
4) converting

16. The ____ function can be used to generate a random number in C.

1) `rand()`
2) `srand()`
3) `random()`
4) `rnd()`

17. All C compilers provide ____ function(s) for creating random numbers, defined in the `stdlib.h` header file.

1) one
2) two
3) three
4) four

18. A function that calls another function is referred to as the ____.

1) function prototype
2) called function
3) calling function
4) function declarator

19. The purpose of a ____ is to identify the data type of the value returned by the function, if any, provide the function with a name, and specify the number, order, and type of values expected by the function.

1) function declarator
2) prototype
3) function body
4) function header

20. ___ is the correct way to include a header file in your program.

1) `#include <header-file-name>`
2) `#include <header-file-name>;`
3) `#include header-file-name`
4) `#include header-file-name;`

21. The items enclosed within the parentheses in a function call statement are called ____ of the function.

1) parameters
2) formal parameters
3) arguments
4) formal arguments

22. The function ____ returns the absolute value of its double-precision argument.

1) `double ceil(double)`
2) `double fmod(double)`
3) `double fabs(double)`
4) `double abs(double)`

23. The argument names in the header line of a function are known as ____.

1) arguments
2) parameters
3) actual arguments
4) actual parameters

24. The minimum requirement of a ____ is that it compile and link with its calling module.

1) function body
2) function prototype
3) stub function
4) function declarator

25. The function ____ returns the common logarithm of its argument.

1) `double exp(double)`
2) `double log(double)`
3) `double log10(double)`
4) `double fmod(double)`

END OF EXAM

Chapter 7 - Modularity Using Functions: Part II

LESSON 9205C

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	You can find a short quiz about the `auto` storage class in: **www.cs.utah.edu/~hamlet/lib/lessons/c26/c26/node1.shtml**.

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	You can find a short quiz about the `static` storage class in: **www.cs.utah.edu/~hamlet/lib/lessons/c26/c26/node2.shtml**.

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. C Tutorial - Lesson 15: Scope and Program Structure:
   http://cplus.about.com/od/beginnerctutoria1/l/aa060402d.htm

2. Pass by Value vs. Pass by Reference:
   www.cs.princeton.edu/~lworthin/126/precepts/pass_val_ref.html

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**.
- When a function invokes itself, the process is called **direct recursion**.
- A variable with **global scope** 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**.
- 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**.
- To use a stored address, C provides us with an **indirection operator**, *.
- A variable with a **local scope** 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**.
- A function can invoke a second function, which in turn invokes the first function; this type of recursion is referred to as **indirect** or **mutual recursion**.
- Passing an address is referred to as a function **pass by reference**, because the called function can reference, or access, the variable using the passed address.
- In **pass by value**, 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**.
- Pointer variables are also **pointers**.
- **Registers** are high-speed storage areas physically located in the computer's processing unit.
- In **run-time initialization**, initialization occurs each time the declaration statement is encountered.
- **Scope** 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** or **recursive** 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**.
- Where and how long a variable's storage locations are kept before they are released can be determined by the **storage class** of the variable.

ANSI C Lesson 9205C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the choice that best completes the statement or answers the question.

1. In ____ initialization, initialization occurs each time the declaration statement is encountered.
1) dynamic
2) static
3) compile-time
4) run-time

2. Coding a function prototype as ____ makes sense when the function is used by a number of other functions in a source code file.
1) private
2) global
3) local
4) `void`

3. If `numAddr` is a pointer, ____ means the variable whose address is stored in `numAddr`.
1) `*numAddr`
2) `numAddr*`
3) `&numAddr`
4) `*&numAddr`

4. To use a stored address, C provides us with an indirection operator, ____.
1) %
2) ^
3) &
4) *

5. A local variable that is declared as ____ causes the program to keep the variable and its latest value even when the function that declared it is through executing.
1) `auto`
2) `static`
3) `extern`
4) `register`

6. Variables created inside a function are ____ variables.
1) recursive
2) private
3) local
4) global

7. ____ is a high-speed storage area physically located in the computer's processing unit.
1) A reserved variable
2) RAM
3) A register
4) A stack

8. The variable `secnum` is ____.
```
int main()
{
  int secnum;
  . . .
}
```
1) local to `main()`
3) local to the program

2) global to `main()`
4) global to the program

9. A declaration statement that specifically contains the word ____ is different from every other declaration statement in that it does not cause the creation of a new variable by reserving new storage for the variable.

1) `auto`
3) `extern`
2) `static`
4) `register`

10. A variable that can store an address is known as a(n) ____ variable.

1) register
3) static
2) pointer
4) extern

11. The ____ of a variable defines the location within a program where that variable can be used.

1) storage class
3) scope
2) time dimension
4) data type

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

1) storage class
3) scope
2) time-dimension
4) data type

13. ____ can only be members of the `auto`, `static`, or `register` storage classes.

1) Constants
3) Local variables
2) `int` variables
4) Global variables

14. The declaration statement ____ declares `milesAddr` to be a pointer variable that can store the address of (that is, will point to) an integer variable.

1) `int milesAddr&;`
3) `int *milesAddr;`
2) `int milesAddr*;`
4) `int &milesAddr;`

15. When a function invokes itself, the process is called ____ recursion.

1) direct
3) self-referential
2) mutual
4) indirect

16. The purpose of the ____ storage class is to extend the scope of a global variable declared in one source code file into another source code file.

1) `auto`
3) `extern`
2) `static`
4) `register`

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

1) local
3) module
2) global
4) function

18. ____ variables have the same time duration as automatic variables.

1) Static
3) Extern
2) Register
4) Global

19. The four available storage classes are called `auto`, `static`, `extern`, and ____.

1) `stack`
3) `void`
2) `intern`
4) `register`

20. When the function returns control to its calling function, its ____ variables "die".

1) local `static`
2) `extern`
3) `local extern`
4) local `auto`

21. ____ variables allow the programmer to "jump around" the normal safeguards provided by functions.

1) Global
2) Local
3) Static
4) `void`

22. Functions that call themselves are referred to as ____ functions.

1) nested
2) recursive
3) loop-back
4) rolling

23. A function can invoke a second function, which in turn invokes the first function; this type of recursion is referred to as ____ recursion.

1) direct
2) mutual
3) self-referential
4) tail

24. A variable with a ____ scope is simply one that has had storage locations set aside for it by a declaration statement made within a function body.

1) function
2) module
3) local
4) global

25. ____ is defined as the section of the program where the variable is valid or "known."

1) Scope
2) Resolution
3) Domain
4) Reach

END OF EXAM

Chapter 8 - Arrays

LESSON 9206C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 8 provides an introduction to arrays. You first learn about one-dimensional arrays: how to create them, use them, initialize them, and pass them as function arguments. In a case study, you use a one-dimensional array and learn to compute averages and standard deviations. You also learn about two-dimensional arrays. Several programming examples are used to show you how to use these types of arrays. You are also briefly introduced to larger dimensional arrays. Finally, common programming and compiler errors are reviewed.

Objectives

- One-dimensional arrays
- Array initialization
- Arrays as function arguments
- Case study: Computing averages and standard deviations
- Two-dimensional arrays
- Common programming and compiler errors

One-Dimensional Arrays

Topic Tip	One-dimensional arrays are sometimes called vectors. For more information, see: **http://en.wikipedia.org/wiki/Array**.

Topic Tip	In C, the starting index value for all arrays is always 0. This starting index value is fixed by the compiler and cannot be altered. Although other high-level languages, such as Visual Basic, allow the programmer to change this starting value (even permitting negative values), C does not. In C, the first array element is always 0, and negative index values are not permitted. For more information, see the Programming Note on page 377.

Topic Tip	Note that some compilers permit double-precision variables as subscripts; in these cases, the double-precision value is truncated to an integer value.

Topic Tip	To practice this topic's concept, write a short program in which you try to access a non-existent array element. What happens? Understand that answers will vary depending on the compiler and operating system being used.

Quick Quiz 1

1. What is an atomic variable?

2. What is a data structure?

3. A(n) _____ array is a list of values of the same data type that is stored using a single group name.

4. Each item in an array is called a(n) _____ or component of the array.

Array Initialization

Topic Tip	Be aware that if the number of initializers is less than the declared number of elements listed in square brackets, the initializers are applied starting with array element 0. After providing this information, ask students to think of an easy way to initialize all elements to zero. They should come up with something like `int myArray[100] = {0};`. Note that this does not work for local `auto` arrays.

Quick Quiz 2

1. The NULL character (_____) is automatically appended to all strings by the C compiler.

2. The individual elements of all global and static arrays (local or global) are, by default, set to _____ at compilation time.

3. Are `auto` local arrays initialized at compilation time? If so, to which value?

4. True/False: It is generally advisable to omit the size of the array in the function header line.

Case Study: Computing Averages and Standard Deviations

Topic Tip	For more information on boundary tests, see **http://en.wikipedia.org/wiki/Black_box_testing#Boundary_value_analysis**.

Two-Dimensional Arrays

Quick Quiz 3

1. What is a two-dimensional array?

2. When initializing two-dimensional arrays, the _____ braces can be omitted.

3. Initialization in a two-dimensional array is done in _____ order.

4. How can you view or interpret arrays of three, four, five, six or more dimensions?

Additional Resources

1. C Tutorial - Lesson 9: Arrays:
   http://cplus.about.com/od/beginnerctutoria1/l/aa040802a.htm

2. C++ Tutorial - Lesson 10: Arrays and Vectors:
   http://cplus.about.com/od/beginnerctutorial/l/aa050102a.htm

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

4. How C Programming Works: Arrays:
   http://computer.howstuffworks.com/c10.htm

5. Black Box Testing: Boundary Value Analysis:
   http://en.wikipedia.org/wiki/Black_box_testing#Boundary_value_analysis

Key Terms

- One of the simplest data structures, called an **array**, is used to store and process a set of values, all of the same data type that forms a logical group.
- An **atomic variable**, which is also referred to as a **scalar variable**, is a variable whose value cannot be further subdivided or separated into a built-in data type.
- C does not check the value of the index being used (called a **bounds check**).
- In **bubble sort**, successive values in the list are compared, beginning with the first two elements.
- A **data structure**, which is also known as an **aggregate data type**, is a data type with two main characteristics. First, its values can be decomposed into individual data elements, each of which is either atomic or another data structure. Secondly, it provides an access scheme for locating individual data elements within the data structure.
- Each item in an array is called an **element** or **component** of the array.
- **External sorts** are used for much larger data sets that are stored in large external disk or tape files, and cannot be accommodated within the computer's memory as a complete unit.
- Each individual element is referred to as an **indexed variable** or a **subscripted variable** because both a variable name and an index or subscript value must be used to reference the element.
- **Internal sorts** are used when the data list is not too large and the complete list can be stored within the computer's memory, usually in an array.
- The two most common methods of performing such searches are the **linear** and **binary search algorithms**.
- In 1958, John McCarthy developed a language at the MIT specifically for manipulating lists; this language was named **LISP**, the acronym for **List Processing**.
- The **NULL** character ('\0') is automatically appended to all strings by the C compiler.
- A **one-dimensional array**, which is also known as both a **single-dimensional array** and a **single-subscript array**, is a list of values of the same data type that is stored using a single group name.
- The **Quicksort algorithm**, which is also called a **"partition" sort**, divides a list into two smaller sublists and sorts each sublist by portioning into smaller sublists, and so on.
- The third subscript in a three-dimensional array is often called the **rank**.
- In a **selection sort**, the smallest value is initially selected from the complete list of data and exchanged with the first element in the list.
- In a linear search, which is also known as a **sequential search**, each item in the list is examined in the order it occurs until the desired item is found or the end of the list is reached.

ANSI C Lesson 9206C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the choice that best completes the statement or answers the question.

1. ____ refers to the first grade stored in the `grades` array.
1) `grades[0]`
2) `grades[1]`
3) `grades(0)`
4) `grades{1}`

2. In C, the array name and index of the desired element are combined by listing the index in ____ after the array name.
1) parentheses
2) square braces
3) curly braces
4) dashes

3. The individual elements of all global and `static` arrays are, by default, set to ____ at compilation time.
1) NULL
2) -1
3) 0
4) 1

4. The ____ character is automatically appended to all strings by the C compiler.
1) `'\NULL'`
2) `'\1'`
3) `'\n'`
4) `'\0'`

5. A(n) ____ is a data type with two main characteristics: (1) its values can be decomposed into individual data elements, and (2) it provides an access scheme for locating individual data elements.
1) data structure
2) scalar data type
3) array
4) atomic data type

6. The initialization of a two-dimensional array is done in ____ order.
1) ascending
2) descending
3) row
4) column

7. ____ declares an array of three rows and four columns.
1) `int val[3,4];`
2) `int val[4,3];`
3) `int val[3][4];`
4) `int val[4][3];`

8. In a one-dimensional array in C, the first element has an index of ____.
1) NULL
2) -1
3) 0
4) 1

9. All ____ arrays are created and destroyed each time the function they are local to is called and completes its execution.
1) global
3) `auto`

2) `static` 4) `extern`

10. `char codes[] = "sample";` sets aside ____ elements in the `codes` array.
1) 5 3) 7
2) 6 4) 8

11. A(n) ____, is used to store and process a set of values, all of the same data type, that forms a logical group.
1) data structure 3) array
2) scalar variable 4) atomic variable

12. A two-dimensional array is sometimes referred to as a ____.
1) list 3) queue
2) vector 4) table

13. Any individual element in an array can be accessed by giving the name of the array and the element's position; this position is called the element's ____ value.
1) component 3) index
2) variable 4) element

14. A(n) ____ variable, is a variable whose value cannot be further subdivided or separated into a built-in data type.
1) data structure 3) array
2) scalar 4) class

15. A ____ is a list of values of the same data type that is stored using a single group name.
1) one-dimensional array 3) three-dimensional array
2) two-dimensional array 4) matrix

16. The term ____ uniquely identifies the element in row 1, column 3.
1) `val[3][1]` 3) `val[3,1]`
2) `val[1][3]` 4) `val[1,3]`

17. ____ shows a correct array initialization statement.
1) `char codes[6] = ['s', 'a', 'm', 'p', 'l', 'e'];`
2) `char codes[] = ('s', 'a', 'm', 'p', 'l', 'e');`
3) `char codes[] = "sample";`
4) `char codes[*] = {'s', 'a', 'm', 'p', 'l', 'e'};`

18. In a function prototype that has a two-dimensional argument, the ____ size is optional.
1) column 3) array
2) row 4) subscript

19. A ____ loop is very convenient for cycling through array elements.
1) `while` 3) `switch`
2) `do-while` 4) `for`

20. ____ is a correct statement.
1) `int grades[5] = {98, 87, 92, 79, 85};`
2) `int grades[5] = 98, 87, 92, 79, 85;`
3) `int grades[5] = (98, 87, 92, 79, 85);`
4) `int grades[5] = [98, 87, 92, 79, 85];`

21. For one-dimensional arrays, the offset to the element with index `i` is calculated as ____.
1) Offset = i * the size of the array
2) Offset = i * the size of the subscript
3) Offset = i * the size of a component + 1
4) Offset = i * the size of an individual element

22. A ____-dimensional array can be viewed as a book of data tables.
1) one
2) two
3) three
4) four

23. Any expression that evaluates a(n) ____ may be used as a subscript.
1) character
2) `double`
3) boolean
4) integer

24. ____ shows a correct array initialization statement.
1) `char codes[4] = {'s', 'a', 'm', 'p', 'l', 'e'};`
2) `char codes[] = {'s', 'a', 'm', 'p', 'l', 'e'};`
3) `char codes = {'s', 'a', 'm', 'p', 'l', 'e'};`
4) `char codes[*] = {'s', 'a', 'm', 'p', 'l', 'e'};`

25. Each item in an array is called a(n) ____ of the array.
1) subscript
2) variable
3) index
4) element

END OF EXAM

Chapter 9 - Character Strings

LESSON 9207C

At a Glance

Lesson Contents

Chapter Notes

Overview

In Chapter 9, you learn how to create, use and manipulate strings in C. You learn how to use several useful library functions for string and character manipulation. You also learn how to use strings for input data validation. An optional section covers how to format strings using `printf()`, `scanf()`, `sprintf()` and `sscanf()`. The case study shows how to process strings character-by-character to be able to count characters and words in a string. Finally, several common programming and compiler errors are reviewed.

Objectives

- String fundamentals
- Library functions
- Input data validation
- Formatting strings (optional)
- Case study: Character and word counting
- Common programming and compiler errors

String Fundamentals

Topic Tip	Some languages have a special `String` or `string` data type. You can find more information about strings in several languages in: **http://en.wikipedia.org/wiki/String_%28computer_science%29**.

Topic Tip	Refer to the Programming Note on page 449 to help explain the difference between `'\n'` and `"\n"`.

Topic Tip	You may be wondering why the `char` data type uses integer values. The Programming Note on page 451 provides a good explanation on this issue.

Quick Quiz 1

1. What is a string literal?
2. What other terms are used to refer to a string literal?
3. The NULL character is _____.
4. The newline character is _____.

Library Functions

Topic Tip	You learned how to initialize strings in Chapter 8, but it is possible you may have forgotten by now. The Programming Note on page 456 summarizes the issues involved in initializing strings.

Quick Quiz 2

1. How does `strcpy(str1, str2)` work?
2. How does `int toupper(char)` work?
3. `isalpha()` is included in the _____ header file.
4. `atoi()` is included in the _____ header file.

Formatting Strings

Topic Tip	A very easy way to convert from character data to numerical data in C is to correctly use the `sscanf()` function. The Programming Note on page 472 explains how to use `sscanf()` to extract the month, day, and year from the string 07/01/94.

Quick Quiz 3

1. What is the difference between the angle brackets and the double quotes in a `#include` statement?

2. The statement `printf("|%25s|","Have a Happy Day");` displays the message `Have a Happy Day`, _____-justified, in a field of 25 characters.

3. What is the result of using the statement `printf("|%-25.12s|","Have a Happy Day");`?

4. True/False: When you use any of the four functions, `printf()`, `scanf()`, `sprintf()`, or `sscanf()`, the control string containing the conversion control sequences need not be explicitly contained within the function.

Additional Resources

1. String (computer science):
   http://en.wikipedia.org/wiki/String_%28computer_science%29

2. C Tutorial - Lesson 10: Strings:
   http://cplus.about.com/od/beginnerctutoria1/l/aa041302a.htm

3. C Programming Tips: Character Strings: Avoiding Common Mistakes Using Character Strings in C:
   http://cplus.about.com/od/cprogrammingtips/l/aa070203a.htm

4. How C Programming Works: Strings:
   http://computer.howstuffworks.com/c35.htm

Key Terms

- A string literal is also referred to as a **string constant** and **string value**, and more conventionally as a **string**.

- A **string literal** is any sequence of characters enclosed in double quotes.

Chapter 10 - Data Files

At a Glance

Lesson Contents

- Overview
- Objectives
- Quick Quizzes
- Additional Resources
- Key Terms
- Lesson Assignment

Chapter Notes

Overview

Chapter 10 provides an overview of the use of data files in C. You learn to declare, open and close file streams, as well as read and write to text files. You also learn how to work with random access files and how to pass and return filenames from and to functions. In the case study, you develop a program that creates and uses a table of constants. You also learn to write and read binary files. Finally, several common programming and compiler errors are reviewed.

Chapter Objectives

- Declaring, opening, and closing file streams
- Reading from and writing to text files
- Random file access
- Passing and returning filenames
- Case study: Creating and using a table of constants
- Writing and reading binary files (optional)
- Common programming and compiler errors

Declaring, Opening, and Closing File Streams

Topic Tip	When opening a file, the filename must not only exist, but you should also have permission to read and/or write to the file.

Topic Tip	The Programming Note on pages 488 and 489 describe in detail the role of each of the mode indicators available when opening a file. It is important to be aware that you can open a file for both input and output at the same time.

Topic Tip	Note the importance of checking `fopen()`'s return value (for more information see the Programming Note on page 492).

Topic Tip	Remember that when using strings for filenames, you should make sure that the string is long enough to include the end-of-string marker.

Quick Quiz 1

1. What is a file?
2. What is a file stream?
3. Each file stream name, when it is declared, is preceded by a(n) _____.
4. If a file opened for reading does not exist, the `fopen()` function returns the _____ address value.

Reading from and Writing to Text Files

Topic Tip	The Programming Note on page 499 describes the issues involved in using full path names when opening a file.

Topic Tip	Practice by writing a short program that writes some output to `stderr`. Where is the output displayed?

Random File Access

Topic Tip	It may be a good idea if you try to write a short program in which you test reading and writing to a file using random file access. It is important that you get a good understanding of how these functions work.

Quick Quiz 2

1. The _____ function resets the current position to the start of the file.
2. What is the role of the `fseek()` function?
3. The function prototype for `ftell()` is contained in _____.
4. What happens if a file is opened for output and the file already exists?

Writing and Reading Binary Files

Topic Tip	You may wish to explore how to use `fwrite()` to write all the elements of an array to a file at once. For an example, see **www.cprogramming.com/tutorial/cfileio.html**.

Quick Quiz 3

1. What are binary files?
2. What is a disadvantage of using binary files (instead of text files)?
3. The specification for explicitly creating and writing to a binary file is made by appending a(n) _____ to the mode indicator when the file is opened.
4. When using `fwrite()` to write to a binary file, the first method argument is always the _____ operator and a variable name.

Additional Resources

1. C Tutorial - Lesson 13: File I/0 and Command Line Arguments:
   http://cplus.about.com/od/beginnerctutoria1/l/aa042902a.htm
2. How C Programming Works: Text Files:
   http://computer.howstuffworks.com/c17.htm
3. How C Programming Works: Binary Files:
   http://computer.howstuffworks.com/c39.htm

4. C File I/O and Binary File I/O:
   www.cprogramming.com/tutorial/cfileio.html

Key Terms

- **Binary files** use the same code as your computer processor uses internally for C's primitive data types.
- Text files are also known as **character-based files**.
- Data that is stored together under a common name on a storage medium other than the computer's main memory is called a **data file**.
- Each file has a unique filename referred to as the file's **external name**.
- A **file** is a collection of data that is stored together under a common name, usually on a disk, magnetic tape, or CD-ROM.
- A **holiday table** consists of legal holiday dates that have been previously stored in a file.
- A file stream that receives (that is, reads) data from a file into a program is referred to as an **input file stream**.
- A file stream that sends (that is, writes) data to a file is referred to as an **output file stream**.
- **Text files** store each individual character, such as a letter, digit, dollar sign, decimal point and so on, using an individual character code (typically ASCII).

ANSI C Lesson 9207C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the letter of the choice that best completes the statement or answers the question.

1. The maximum allowable filename in the DOS operating system is ____.
1) 8 characters plus an optional period and 3-character extension
2) 14 characters
3) 155 characters
4) 255 characters

2. Line ____ in the following section of code checks for the end-of-string character.

```
 1 void strcopy (char string1[], char string2[])
 2 {
 3   int i = 0;
 4
 5   while (string2[i] != '\0')
 6   {
 7     string1[i] = string2[i];
 8     i++;
 9   }
10   string1[i] = '\0';
11 }
```

1) 3
2) 5
3) 7
4) 10

3. ____ causes the same display as the statement `printf("Hello World!");`.
1) `fprintf(stdout,"Hello World!");`
2) `fprintf(stdin,"Hello World!");`
3) `fprintf(stderr,"Hello World!");`
4) `fprintf(NULL,"Hello World!");`

4. To write to a binary file you use the ____ function.
1) `fput()`
2) `fputb()`
3) `fwrite()`
4) `write()`

5. The actual declaration of the `FILE` structure is contained in the ____ standard header file.
1) `stdio.h`
2) `stdlib.h`
3) `file.h`
4) `stream.h`

6. A ____ is a one-way transmission path that is used to connect a file stored on a physical device, such as a disk or CD-ROM, to a program.

1) data file
2) text file
3) binary file
4) file stream

7. The statement ____ displays the message `Have a Happy Day`, right-justified, in a field of 25 characters.

1) `printf("%s25","Have a Happy Day");`
2) `printf("%s-25","Have a Happy Day");`
3) `printf("%25s","Have a Happy Day");`
4) `printf("%-25s","Have a Happy Day");`

8. The array `char message[81];` can be used to store a string of up to ____ characters.

1) 79
2) 80
3) 81
4) 82

9. The string `"Good Morning!"` is stored in memory using a character array of size ____.

1) 13
2) 14
3) 15
4) 16

10. Programs that use the `gets()` routine must include the ____ header file.

1) `stdio.h`
2) `stdlib.h`
3) `string.h`
4) `ctype.h`

11. Typically, the ____ function is used to "assemble" a string from smaller pieces until a complete line of characters is ready to be written, either to the standard output device or to a file.

1) `strcpy()`
2) `strcat()`
3) `sscanf()`
4) `sprintf()`

12. Programs that use the `atoi()` routine must include the ____ header file.

1) `stdio.h`
2) `stdlib.h`
3) `string.h`
4) `ctype.h`

13. The value assigned to the NULL constant is ____.

1) `'\n'`
2) `'\0'`
3) `'\NULL'`
4) `"NULL"`

14. Data that is stored together under a common name on a storage medium other than the computer's main memory is called a ____.

1) database
2) data file
3) text file
4) binary file

15. Notice that each file stream name, when it is declared, is preceded by a(n) ____.

1) pipe
2) underscore
3) ampersand
4) asterisk

16. ____ reads values for the listed arguments from the file, according to the format.

1) `fgetc()`
2) `fgets()`
3) `fprintf()`
4) `fscanf()`

17. When using `#include`, the characters _____, tell the compiler to start looking in the default directory where the program file is located.

1) `""`
2) `<>`
3) `//`
4) `\\`

18. ____ files store each individual character, such as a letter, digit, dollar sign, decimal point, and so on, using an individual character code.

1) Data
2) Text
3) Binary
4) ASCII

19. A file stream is closed using the ____ function.

1) `exit()`
2) `osclose()`
3) `fclose()`
4) `close()`

20. `fputc()` is the general form of ____.

1) `fput()`
2) `putc()`
3) `putchar()`
4) `fputchar()`

END OF EXAM

Chapter 11 - Arrays, Addresses, and Pointers

LESSON 9208C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 11 thoroughly covers how pointers work in C. You learn the relationship between arrays and pointers. You also learn how to manipulate pointers and how to pass and use array addresses. You practice processing strings using pointers and learn how to create strings using pointers. Finally, several common programming and compiler errors are reviewed.

Objectives

- Array names as pointers
- Manipulating pointers
- Passing and using array addresses
- Processing strings using pointers
- Creating strings using pointers
- Common programming and compiler errors

Array Names as Pointers

Topic Tip	In C, `5["abcdef"]` is actually valid. For more information, see: **http://c-faq.com/aryptr/joke.html**.

Quick Quiz 1

1. Pointers, both as variables and function parameters, are used to store _____.
2. With respect to pointers, what is an offset?
3. When an array is created, the compiler automatically creates an internal pointer _____.
4. Can a pointer access be replaced using subscript notation? If so, under which circumstances?

Manipulating Pointers

Topic Tip	For a good explanation on C pointer arithmetic, see **www.cs.umd.edu/class/spring2003/cmsc311/Notes/BitOp/pointer.html**.

Quick Quiz 2

1. What is the purpose of adding or subtracting numbers from pointers?
2. When adding or subtracting numbers to pointers, the computer automatically adjusts the number to ensure that the result still "points to" a value of the original _____.
3. How are pointer operations scaled automatically?
4. When initializing pointers you must be careful to set a(n) _____ in the pointer.

Passing and Using Array Addresses

Topic Tip	For more information of pointers to functions, see **www.newty.de/fpt/fpt.html**.

Processing Strings Using Pointers

Topic Tip	It may be useful to see a real compiler implementation of `strcpy()`. For an example, see: **http://freebsd.active-venture.com/FreeBSD-srctree/newsrc/libkern/strcpy.c.html**.

Creating Strings Using Pointers

Topic Tip	It is important that you understand how memory is allocated when a string (character array) is declared versus when a `char` pointer is declared. For a good explanation of the subject, see the Programming Note on page 565.

Quick Quiz 3

1. If `nums` is a two-dimensional integer array, `*(*(nums + 1) + 2)` refers to element _____.

2. What is the main difference between the following declarations?
```
char message1[81] = "this is a string";
char *message2 = "this is a string";
```

3. The header line _____ declares `calc` to be a pointer to a function that returns an integer.

4. What does the declaration `char *seasons[4];` create?

Additional Resources

1. FAQ: Arrays and Pointers:
   http://c-faq.com/~scs/cgi-bin/faqcat.cgi?sec=aryptr

2. Tutorial: Pointers in C and C++:
   http://augustcouncil.com/~tgibson/tutorial/ptr.html

3. The Function Pointer Tutorials:
   www.newty.de/fpt/fpt.html

4. C Tutorial - Lesson 8: An Introduction To Pointers:
   http://cplus.about.com/od/beginnerctutoria1/l/aa040702a.htm

5. How C Programming Works: Pointers:
   http://computer.howstuffworks.com/c20.htm

Key Terms

- **Anagram** is a rearrangement of the letters in a word or phrase that takes another word or phrase.
- One unique feature of pointers is that **offsets** may be included in expressions using pointers.
- A word, phrase, or sentence that reads the same forward and backward, such as *top spot* is a **palindrome**.
- When an array is created, the compiler automatically creates an internal **pointer constant** for it and stores the base address of the array in this pointer.

ANSI C Lesson 9208C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the choice that best completes the statement or answers the question.

1. The expression ____ adds 3 to "the variable pointed to by `gPtr`."
1) `*(gPtr + 3)`
2) `*gPtr + 3`
3) `gPtr + 3`
4) `&gPtr + 3`

2. The ____ in the expression `*(gPtr + 1)` is an offset.
1) `*`
2) `gPtr`
3) `+`
4) `1`

3. Assuming `grade` is an array of ten integers, the statement ____ is invalid.
1) `grade = &grade[2];`
2) `*grade = *(grade + 2);`
3) `*grade = *grade + 2;`
4) `*grade = *(&grade[2]) + 2;`

4. The indirection operator in C is ____.
1) `&`
2) `*`
3) `->`
4) `.`

5. After creating two variables as follows:
```
char message1[81] = "this is a string";
char *message2 = "this is a string";
```
The statement ____ is not valid in C.
1) `message1 = "A new message";`
2) `message2 = "A new message";`
3) `message2 = message1;`
4) `message2[0] = 'T';`

6. The header line ____ declares `calc` to be a pointer to a function that returns an integer.
1) `int *calc()`
2) `int (*calc)()`
3) `int &calc()`
4) `int calc(*)`

7. A suitable equivalent to the function header `calc(int pt[2][3])` is ____.
1) `calc(int *(*pt))`
2) `calc(int (*pt)[])`
3) `calc(int (*pt)[2])`
4) `calc(int (*pt)[3])`

8. When working with pointers, the ____ tells the number of variables that are to be skipped over.
1) indirection operator
2) address operator
3) offset
4) address

9. You can replace lines 5 and 6 in the following function with ____.

```
1 /* copy string2 to string1 */
2 void strcopy(char string1[], char string2[])
3 {
4   int i = 0;
5   while (string1[i] = string2[i])
6     i++;
7 }
```

1) `while (*string1 = *string2) ;`
2) `while (*string1 = string2) ;`
3) `while (*string1++ = *string2++) ;`
4) `while (*++string1 = *++string2) ;`

10. When an array is created, the compiler automatically creates an internal ____ for it and stores the base address of the array in it.

1) pointer constant
2) pointer
3) symbolic constant
4) location

11. Consider the declarations

```
int nums[100];
int *nPtr;
```

The statement ____ produces the same result as `nPtr = nums;`.

1) `nPtr = &nums[0];`
2) `nPtr = nums[0];`
3) `nPtr = *nums[0];`
4) `nPtr = &nums;`

12. `&grade[3]` is equivalent to ____; assume that `grade` is an array of integers, and each integer requires 4 bytes of storage..

1) `&grade[0] + 3`
2) `&grade[0] + 4`
3) `&grade[0] + (3 * 4)`
4) `&grade[0] + (3 / 4)`

13. The address operator in C is ____.

1) `&`
2) `*`
3) `->`
4) `.`

14. If `nums` is a two-dimensional integer array, ____ refers to element `nums[0][0]`.

1) `*nums`
2) `*(*nums)`
3) `*(&nums)`
4) `&(*nums)`

15. ____ uses the pointer and then increments it.

1) `*ptNum--`
2) `*--ptNum`
3) `*ptNum++`
4) `*++ptNum`

16. If numPtr is declared as a pointer variable, the expression ____ can also be written as numPtr[i].

1) *numPtr + i
2) (numPtr + i)
3) *numPtr
4) *(numPtr + i)

17. Consider the following declarations of a function that receives an array of integers and finds the element with the maximum value:

(i) `findMax(int *vals, int numEls)`
(ii) `findMax(int vals[], int numEls)`

The address in `vals` may be modified ____.
1) only if the function is declared as in (i)
2) only if the function is declared as in (ii)
3) if either (i) or (ii) is used
4) in neither case because an array variable cannot be modified (it is a pointer constant)

18. Of the following expressions, ____ is the most commonly used. This is because such an expression allows each element in an array to be accessed as the address is "marched along" from the starting address of the array to the address of the last array element.

1) `*ptNum--`
2) `*--ptNum`
3) `*ptNum++`
4) `*++ptNum`

19. If `nums` is a two-dimensional integer array, ____ refers to element `nums[1][0]`.

1) `*nums[1]`
2) `*nums[0]`
3) `*nums + 1`
4) `*nums++`

20. `int *ptNum = &miles;` is ____.
1) always valid
2) never valid
3) only valid if `miles` is declared as an integer variable before `ptNum` is declared
4) only valid if `miles` is declared as an array of integers before `ptNum` is declared

21. Pointers ____ be initialized when they are declared.

1) must
2) must not
3) can
4) cannot

22. If `gPtr` is a pointer that points to the first element of an integer array (and each integer requires four bytes of storage), ____ references the variable that is three integers beyond the variable pointed to by `gPtr`.

1) `*gPtr + 3`
2) `*(gPtr + 3)`
3) `*(gPtr + 3 * 4)`
4) `*(gPtr + 3 / 4)`

23. If we store the address of `grade[0]` in a pointer named `gPtr` (using the assignment statement `gPtr = &grade[0];`), then, the expression ____ references `grade[0]`.

1) `gPtr(0)`
2) `gPtr`
3) `&gPtr`
4) `*gPtr`

24. Adding ____ to a pointer causes the pointer to point to the next element of the original data type being pointed to.

1) `1`
2) `1 * sizeof(data type being pointed to)`
3) `2`
4) `2 * sizeof(data type being pointed to)`

25. In performing ____ on pointers, we must be careful to produce addresses that point to something meaningful.

1) comparisons
2) arithmetic
3) subscript operations
4) duplication

END OF EXAM

Chapter 12 - Structures

LESSON 9209C

At a Glance

Lesson Contents

Chapter Notes

Overview

Chapter 12 covers the use of structures and unions in C. You learn to create and use structures and arrays of structures. You also learn how to pass and return structures to and from functions. You learn how to create and use unions in C. Finally, several common programming and compiler errors are reviewed.

Objectives

- Single structures
- Arrays of structures
- Passing and returning structures
- Unions (optional)
- Common programming and compiler errors

Single Structures

Topic Tip	The Programming Note on page 581 introduces the terms *homogeneous* and *heterogeneous* data structure.

Quick Quiz 1

1. What is the difference between a structure's form and the structure's contents?

2. Assigning actual data values to the data items of a structure is called _____ the structure.

3. What is the difference between a homogeneous and a heterogeneous data structure?

4. For non-ANSI C compilers, the keyword _____ must be placed before the keyword `struct` for initialization within a local declaration statement.

Arrays of Structures

Topic Tip	Using `typedef` statements with structure declarations is very common. The Programming Note on page 585 provides a good explanation on the use of `typedef` statements.

Quick Quiz 2

1. A _____ statement provides a simple method for creating a new and typically shorter name for an existing structure type.

2. What are parallel arrays?

3. What is the problem with parallel arrays?

4. When initializing an array of structures, the _____ braces are not necessary.

Unions

Topic Tip	Usually we may have a hard time thinking of uses for unions. You can find a detailed example of a situation that benefits from using a union at **www.cs.utk.edu/~plank/plank/classes/cs140/Spring-1999/notes/Unions/**.

Quick Quiz 3

1. Individual structure members may be passed to a function in the same manner as any _____ variable.

2. An alternative to passing a copy of a structure is to pass the _____ of the structure.

3. What is a union?

4. How much memory space does a union reserve?

Additional Resources

1. C Tutorial – Lesson 11: Structures:
   http://cplus.about.com/od/beginnerctutorial1/l/aa041602a.htm

2. How C Programming Works: Pointers to Structures:
   http://computer.howstuffworks.com/c31.htm

3. C FAQ: Structures, Unions, and Enumerations:
   http://c-faq.com/struct/

4. C Tutorial: Unions:
   www.sysprog.net/cunions.html

Key Terms

- The structure's **contents** consist of the actual data stored in the symbolic names.
- Each of the individual data items in a "structure" (single unit) is an entity by itself that is referred to as a **data field**.
- A structure's **form** consists of the symbolic names, data types, and arrangement of individual data fields in the record.
- A record is a **heterogeneous data structure**, which means that each of its components can be of different data types.
- An array is a **homogeneous data structure**, which means that each of its components must be of the same type.
- The data items of a structure are called **members of the structure**.
- **Parallel arrays** are two or more arrays, where each array has the same number of elements and the elements in each array are directly related by their position in the arrays.
- Assigning actual data values to the data items of a structure is called **populating the structure**.
- Taken together, all the data fields form a single unit that is referred to as a **record**.
- In C, a record is referred to as a **structure**, and we use these terms interchangeably.

- When defining structures, if the form of the structure is not followed by any variable names, the list of structure members must be preceded by a user-selected **structure type name**.
- A **union** is a data type that reserves the same area in memory for two or more variables, each of which can be a different data type.

Chapter 13 - Dynamic Data Structures

At a Glance

Lesson Contents

Chapoter Notes

Overview

Chapter 13 provides an introduction to linked lists and dynamic memory allocation in C. You learn how to use and create linked lists, stacks and queues. You also learn how to create dynamically linked lists. Finally, you learn about some common programming and compiler errors, and how to avoid them.

Objectives

- Introduction to linked lists
- Dynamic memory allocation
- Stacks
- Queues
- Dynamically linked lists
- Common programming and compiler errors

Introduction to Linked Lists

Topic Tip	You may want to use an animation to help visualize how a linked list works. For example, see **www.cs.stir.ac.uk/~mew/dissertation/simulation.htm**.

Topic Tip	Variants of linked lists include doubly-linked lists and circularly-linked lists. For more information, see **http://en.wikipedia.org/wiki/Linked_list**.

Quick Quiz 1

1. What is a linked list?

2. What is a self-referencing structure?

3. All programming languages that support pointers provide a special pointer value, known as both NULL and _____, which acts as a sentinel or flag to indicate when the last structure has been processed.

4. The expression `t1.nextaddr->name` can, of course, be replaced by the equivalent expression _____, which explicitly uses the indirection operator.

Dynamic Memory Allocation

Topic Tip	Make sure you read and understand why it is very important to check return values when making `malloc()` and `realloc()` function calls (see the Programming Note on page 616).

Stacks

Topic Tip	You may use an animation to helps visualize how a stack works. For example, see **www.cs.usask.ca/resources/tutorials/csconcepts/1998_5/stacks/java/** or **www.cs.hope.edu/~alganim/jvall/applet/stack.html**.

Topic Tip	Reverse Polish Notation (RPN) (postfix algebra) can be easily implemented using stacks. For more information, read the Historical Note on page 611.

Quick Quiz 2

1. What functions are available in C for the dynamic allocation and release of memory space?

2. How does `malloc()` work?

3. A(n) _____ is a special type of linked list in which objects can only be added to and removed from the top of the list.

4. The operation of placing a new structure on the top of a stack is called a PUSH, and removing a structure from a stack is called a(n) _____.

Queues

Topic Tip	Stacks and queues are two special forms of a more general data object called a deque (pronounced "deck"). The term "deque" stands for "double-ended queue." For more information, see the Historical Note on page 620.

Topic Tip	You may use an animation to help visualize how a stack works. For example, see **www.cs.odu.edu/~zeil/cs361/Demos/replays/queuelist.html**.

Quick Quiz 3

1. What is a queue?

2. What are the names of the operations used to add and remove items to/from a queue?

3. In a(n) _____, elements can be added and removed from anywhere within the list.

4. The operation of adding a new structure to a dynamically linked list is called a(n) _____.

Additional Resources

1. Linked List:
   http://en.wikipedia.org/wiki/Linked_list

2. Stack:
   http://en.wikipedia.org/wiki/Stack_%28data_structure%29

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

4. Deque:

http://en.wikipedia.org/wiki/deque

Key Terms

- **Dynamic memory allocation** makes it unnecessary to reserve a fixed amount of memory for a scalar, array or structure variable in advance.
- Placing a new item on top of the queue is formally referred to as **enqueueing**.
- The **heap** consists of unallocated memory that can be allocated to a program as requested, while the program is executing.
- The field on which a list is ordered is referred to as the **key field**, and insertions and deletions are always made to preserve the ordering of this field.
- A **linked list** is a set of structures in which each structure contains at least one member whose value is the address of the next logically ordered structure in the list.
- Items are removed from a **queue** in the order in which they were entered.
- Dynamic memory allocation is also known as **run-time allocation**.
- Structures that are "linked" together by including the address of the next structure in the structure immediately preceding it are known as **self-referencing structures**.
- The operation of removing an item from a queue is formally referred to as **serving**.

ANSI C Lesson 9209C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the letter of the choice that best completes the statement or answers the question.

1. Structures that are "linked" together by including the address of the next structure in the structure immediately preceding it are known as ____ structures.
1) linked
2) self-referencing
3) dynamic
4) sequential

2. ____ arrays are two or more arrays, where each array has the same number of elements and the elements in each array are directly related by their position in the arrays.
1) Two-dimensional
2) Multi-dimensional
3) Parallel
4) Complex

3. If `pt` is declared as a pointer to a structure of type `Employee`, ____ refers to the variable whose address is in the `pt.idNum` variable.
1) `(*pt).idNum`
2) `*pt.idNum`
3) `pt->idNum`
4) `(*pt.)idNum`

4. ____ is not a valid C statement.
1) `struct {int month; int day; int year;} birth;`
2) `struct {int month; int day; int year;} birth, current;`
3) `struct Date {int month; int day; int year;};`
4) `struct {int month, int day, int year} birth;`

5. Stacks and queues are two special forms of a more general data object called a(n) ____.
1) array
2) table
3) deque
4) heap

6. ____ memory allocation makes it unnecessary to reserve a fixed amount of memory for a scalar, array, or structure variable in advance.
1) Dynamic
2) Static
3) Partial
4) Advanced

7. The operation of removing a structure from a stack is called a ____.
1) PUSH
2) POP
3) DELETE
4) REMOVE

8. A ____ is a special type of linked list in which objects can only be added to and removed from the top of the list.
1) heap
2) stack
3) queue
4) set

9. The function call ____ passes a copy of the complete `emp` structure to `calcNet()`.

1) `calcNet(struct emp);`
2) `calcNet(*emp);`
3) `calcNet(&emp);`
4) `calcNet(emp);`

10. The following function cycles through a linked list and displays its contents. Line 3 can be replaced with ____.

```
1 void display(struct myStruct *contents)
2 {
3   while (contents != NULL)
4   {
5     printf("%-30s\n",contents->name, contents->phoneNum);
6     contents = contents->nextaddr;
7   }
8 }
```

1) `while (isValid(contents))`
2) `while (contents != EOF)`
3) `while (!contents)`
4) `while (contents != NIL)`

11. ____ is equivalent to `(*pointer).member`.

1) `*pointer.member`
2) `pointer>member`
3) `pointer->member`
4) `pointer@member`

12. Each member of a structure is accessed by giving both the structure name and individual data item name, separated by a ____.

1) `@`
2) `->`
3) `:`
4) `.`

13. If you have declared a structure named `Date`, you can then make the name `DATE` a synonym for the terms struct `Date`, by using the statement ____.

1) `typedef struct Date DATE;`
2) `typedef DATE struct Date;`
3) `#define struct Date DATE`
4) `#define DATE struct Date`

14. The expression `t1.nextaddr->name` can be replaced by the equivalent expression ____.

1) `(*t1.nextaddr).name`
2) `(&t1.nextaddr).name`
3) `*(*t1.nextaddr).name`
4) `*((*t1.nextaddr).name)`

15. ____ reserves space for an array of n elements of the specified size.

1) `malloc()`
2) `calloc()`
3) `realloc()`
4) `nalloc()`

16. A(n) ____ is a set of structures in which each structure contains at least one member whose value is the address of the next logically ordered structure in the list.

1) array
2) stack
3) queue
4) linked list

17. The operation of removing a structure from a dynamically linked list is called a(n) ____.

1) POP
2) SERVE
3) REMOVE
4) DELETE

18. In C, a record is referred to as a(n) ____.

1) data field
2) union
3) structure
4) tuple

19. A union reserves sufficient memory locations to accommodate ____.

1) its smallest member's data type
2) its largest member's data type
3) all of its members' data types
4) none of its members' data types

20. ____ reserves the number of bytes requested by the argument passed to the function.

1) `malloc()`
2) `calloc()`
3) `realloc()`
4) `balloc()`

END OF EXAM

Chapter 14 - Additional Capabilities

LESSON 9210C

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 - Lesson 3: Constants:
   http://cplus.about.com/od/beginnerctutoria1/l/aa031002b.htm

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

4. C Tutorial - Lesson 13: File I/0 and Command Line Arguments: **http://cplus.about.com/od/beginnerctutoria1/l/aa042902d.htm**

Key Terms

- The `typedef` declaration statement permits constructing alternate names for an existing C data type name. These alternate names are known as **aliases**.
- In an **arithmetic right shift** (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**.
- **Command-line arguments** are arguments that are typed on the command line.
- Both the `#ifndef` and `#ifdef` directives permit **conditional compilation** 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**, 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** 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**, and the substitution of the text in place of the symbolic name is called a **macro expansion** or **macro substitution**.
- In an operation like `op1 & op2`, the variable `op2` is called a **mask**.
- `AND` operations are extremely useful in **masking**, or eliminating, selected bits from an operand.
- The conditional operator, `?:`, is unique in C in that it is a **ternary operator**.

ANSI C Lesson 9210C Exam

Please complete the following exam. You may use the electronic grading system for quicker response. Simply log on to **www.study-electronics.com** and enter your credentials. Once the exam has been submitted, your results will be returned within 72 hours. You may also e-mail your answers to **faculty@cie-wc.edu**, or fax them to us at 1-216-781-0331. If you have any questions, please contact the Instruction Department.

Multiple Choice
Identify the choice that best completes the statement or answers the question.

1. The conditional preprocessor directive ____ means "if not defined".
   1) `#ifdef`
   2) `#ifndef`
   3) `#ifnotdef`
   4) `#ifnotdefined`

2. The operator ____ is a ternary operator.
   1) `?:`
   2) `->`
   3) `&`
   4) `[]`

3. `ARRAY first, second;` is equivalent to the two definitions `int first[100];` and `int second[100];` if ____.
   1) you are using a pre-ANSI C compiler
   2) you are using a C/C++ compiler
   3) the statement `typedef int ARRAY[100];` is used before
   4) the statement `#define ARRAY int[100];` is used before

4. For unsigned integers, each left shift (using the << operator) corresponds to ____.
   1) multiplication by 2
   2) division by 2
   3) multiplication by 4
   4) division by 4

5. Enumerated lists are identified by the reserved word ____ followed by an optional, user-selected name and a required list of one or more constants.
   1) `list`
   2) `typedef`
   3) `enumerate`
   4) `enum`

6. A conditional expression uses the conditional operator, ____, and provides an alternate way of expressing a simple `if-else` statement.
   1) `->`
   2) `?:`
   3) `?`
   4) `:`

7. The equivalence produced by a `typedef` statement can frequently be produced equally well by a ____ statement.
   1) `enum`
   2) `#define`
   3) `struct`
   4) `alias`

8. ____ bit operations are extremely useful in masking, or eliminating, selected bits from an operand.
   1) `&`
   2) `|`
   3) `>>`
   4) `<<`

9. Explicit values can be assigned to each enumerated constant, with unspecified values automatically continuing the integer sequence from the last specified value. For example, ____.

1) `enum {Mon: 1, Tue, Wed, Thr, Fri, Sat, Sun};`
2) `enum {Mon, Tue, Wed, Thr, Fri, Sat, Sun};`
   `Mon = 1;`
3) `enum {Mon = 1, Tue, Wed, Thr, Fri, Sat, Sun};`
4) `enum {Mon 1, Tue, Wed, Thr, Fri, Sat, Sun};`

10. ____ is the most frequently used conditional preprocessor directive.

1) `#define`
2) `#else`
3) `#ifdef`
4) `#ifndef`

11. In an arithmetic right shift (using the >> operator), each right shift corresponds to ____.

1) multiplication by 2
2) division by 2
3) multiplication by 4
4) division by 4

12. `1 0 1 1 0 0 1 1 ____ 1 1 0 1 0 1 0 1` results in `1 0 0 1 0 0 0 1`.

1) `&`
2) `|`
3) `>>`
4) `<<`

13. `1 0 1 1 0 0 1 1 ____ 1 1 0 1 0 1 0 1` results in `1 1 1 1 0 1 1 1`.

1) `&`
2) `|`
3) `>>`
4) `<<`

14. The definition `REAL val;` is ____.

1) not valid in C
2) will generate a compiler warning
3) is equivalent to `double val;` if it comes after `typedef double REAL;`
4) defines a macro instance if it comes after `#define REAL double`

15. The ____ statement provides an unconditional transfer of control to some other statement in a program.

1) `jump`
2) `goto`
3) `label`
4) `transfer`

16.
```
#define SQUARE(x) x * x
val = SQUARE(num1 + num2);
```

results in the equivalent statement ____.

1) `val = num1 + (num2 * num1 + num2);`
2) `val = (num1 + num2 * num1) + num2;`
3) `val = (num1 + num2) * (num1 + num2);`
4) `val = num1 + num2 * num1 + num2;`

17. ____ is the exclusive `OR` operator.

1) `&`
2) `|`
3) `^`
4) `~`

18. The conditional preprocessor directive ____ means “if defined”.

1) `#ifdef`
2) `#ifndef`
3) `#ifdefined`
4) `#if_def`

19. Upon encountering the command line `pgm14.3 three blind mice`, the operating system stores it as a sequence of ____ strings.

1) one
2) three
3) four
4) five

20. The ____ operator causes a bit-by-bit AND comparison between its two operands.

1) `~`
2) `^`
3) `&&`
4) `&`

21. `1 0 1 1 0 0 1 1` ____ `1 1 0 1 0 1 0 1` results in `0 1 1 0 0 1 1 0`.

1) `&`
2) `|`
3) `^`
4) `~`

22. `typedef` can be used to create ____.

1) structures
2) variables
3) aliases
4) macros

23. Using even one ____ statement in a program is almost always a sign of bad programming structure.

1) `enum`
2) `typedef`
3) `goto`
4) `#define`

24. The statement ____ makes the name `REAL` a synonym for `double`.

1) `typedef double REAL;`
2) `#define double REAL`
3) `enum REAL double`
4) `typedef REAL double;`

25. In the equivalence statement `#define SQUARE(x) x * x`, `x` is ____.

1) fixed
2) an error
3) a variable
4) an argument

END OF EXAM

Appendix - Quick Quiz Answers

Chapter 1

Quiz 1

1. bit
2. The Arithmetic and Logic Unit (ALU) of a computer performs all of the computations, such as addition, subtraction, comparisons, and so on, that a computer provides.
3. The control unit of a computer directs and monitors the overall operation of the computer.
4. direct access storage device (DASD)
   direct access storage device
   DASD

Quiz 2

1. Programming languages that use the substitution of word-like symbols, such as ADD, SUB, MUL, for the binary opcodes, and both decimal numbers and labels for memory addresses are referred to as assembly languages.
2. compiler
3. A linker combines additional machine language code with the object program to create a final executable program.
4. pseudocode

Quiz 3

1. The technique used by professional software developers for understanding the problem that is being solved and for creating an effective and appropriate software solution is called the software development process.
2. selection
3. invocation
4. When writing a program, a repetition structure, which is also referred to as looping and iteration, provides the ability for the same operation to be repeated based on the value of a condition.

Chapter 2

Quiz 1

1. The names of functions, as well as all of the words permitted in a program that have special meaning to the compiler, are collectively referred to as identifiers.
2. A function header line, which is always the first line of a function, contains three pieces of information (1) what type of data, if any, is returned by the function, (2) the name of the function, and (3) what type of data, if any, is sent into the function.
3. newline escape sequence
4. reserved word
   keyword
   reserved or keyword

Quiz 2

1. A data type is defined as a set of values and a set of operations that can be applied to these values.

2. precision
3. An expression is any combination of operators and operands that can be evaluated to yield a value.
4. Associativity

Quiz 3

1. Variables
2. An assignment statement tells the computer to assign a value to (that is, store a value in) a variable.
3. Definition statements define or tell the compiler how much memory is needed for data storage.
4. initialized

Chapter 3

Quiz 1

1. =
2. implicit
3. A previously stored number, if it has not been initialized to a specific and known value, is frequently referred to as a garbage value.
4. Using the increment operator, ++, the expression variable = variable + 1 can be replaced by the either the expression variable++ or ++variable. When the ++ operator appears before a variable, it is called a prefix increment operator.

Quiz 2

1. prompt
2. buffer
3. Programs that detect and respond effectively to unexpected user input are formally referred to as robust programs and informally as "bullet-proof" programs.
4. The basic approach to handling invalid data input is referred to as user-input validation, which means validating the entered data either during or immediately after the data have been entered, and then providing the user with a way of reentering any invalid data.

Quiz 3

1. field width specifiers
2. Literal values that appear many times in the same program are referred to by programmers as magic numbers.
3. equivalence
4. Literal data refers to any data within a program that explicitly identifies itself.

Chapter 4

Quiz 1

1. The term flow of control refers to the order in which a program's statements are executed.
2. relational

3. conditions
4. If an `expression` has any non-0 value (true), `!expression` produces a 0 value (false). If an `expression` is false to begin with (has a 0 value), `!expression` is true and evaluates to 1.

Quiz 2

1. one-way
2. compound
3. Including one or more `if-else` statements within an `if` or `if-else` statement is referred to as a nested `if` statement.
4. Indentation used within an `if-else` is always irrelevant as far as the compiler is concerned. Whether the indentation exists or not, the compiler will, by default, associate an `else` with the closest previous unpaired `if`, unless braces are used to alter this default pairing.

Quiz 3

1. A `switch` statement is a specialized selection statement that can be used in place of an `if-else` chain where exact equality to one or more integer constants is required.
2. `case`
3. Any number of `case` labels may be contained within a `switch` statement, in any order. However, if the value of the expression does not match any of the `case` values, no statement is executed unless the keyword `default` is encountered. The word `default` is optional and operates the same as the last `else` in an `if-else` chain. If the value of the expression does not match any of the `case` values, program execution begins with the statement following the word `default`.
4. `break`

Chapter 5

Quiz 1

1. A section of code that is repeated is referred to as a loop, because after the last statement in the code is executed, the program branches, or loops, back to the first statement and starts another repetition through the code.
2. counter-controlled
3. The transfer of control back to the start of a `while` statement to reevaluate the expression is known as a program loop.
4. sentinel-controlled

Quiz 2

1. sentinels
2. Ctrl and Z
3. A `break` statement, as its name implies, forces an immediate break, or exit, from `switch`, `while`, `for`, and `do-while` statements only.

4. Although the initializing and altering lists can be omitted from a `for` statement, omitting the tested expression results in an infinite loop.

Quiz 3

1. There are many situations in which it is very convenient to have a loop contained within another loop. Such loops are called nested loops.
2. inner
3. `do-while`
4. There is one type of application that is ideally suited for a posttest loop, which is the input data validation application.

Chapter 6

Quiz 1

1. A function that is called or summoned into action by its reference in another function is a called function. A function that calls another function is referred to as the calling function.
2. Arguments
   actual arguments
   actual parameters
3. When a function simply receives copies of the values of each of the arguments and must determine where to store these values before it does anything else, this is known as a pass by value (or a call by value).
4. Parameters
   formal parameters
   formal arguments

Quiz 2

1. stub
2. call by value
3. To return a value you use a `return` statement. For example, `return` *`expression;`* or `return (`*`expression`*`);`
4. The prototype for a function with an empty parameter list requires either writing the keyword `void` or nothing at all between the parentheses following the function's name.

Quiz 3

1. Random numbers are a series of numbers whose order cannot be predicted.
2. Pseudorandom numbers are numbers which are not really random, but are sufficiently random for the task at hand.
3. caling
4. 15

Chapter 7

Quiz 1

1. Variables that are created inside a function and available only to the function are said to be local to the function, or local variables.

2. Scope is defined as the section of the program where the variable is valid or "known."
3. local
4. global

Quiz 2

1. storage class
2. Registers are high-speed storage areas physically located in the computer's processing unit.
3. Passing an address is referred to as a function pass by reference, because the called function can reference, or access, the variable using the passed address.
4. pointer variable
   pointer
   pointer or pointer variable

Quiz 3

1. recursive
2. direct
3. A function can invoke a second function, which in turn invokes the first function; this type of recursion is referred to as indirect or mutual recursion.
4. 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.

Chapter 8

Quiz 1

1. An atomic variable, which is also referred to as a scalar variable, is a variable whose value cannot be further subdivided or separated into a built-in data type.
2. A data structure, which is also known as an aggregate data type, is a data type with two main characteristics. First, its values can be decomposed into individual data elements, each of which is either atomic or another data structure. Second, it provides an access scheme for locating individual data elements within the data structure.
3. one-dimensional
   single-dimensional
   single-subscript
4. element

Quiz 2

1. '\0'
2. zero
   0
   zero (0)
3. No. The values within `auto` local arrays are undefined.
4. True

Quiz 3

1. A two-dimensional array, or table, consists of both rows and columns of elements.

2. inner
3. row
4. Arrays of three, four, five, six, or more dimensions can be viewed as mathematical *n*-tuples.

Chapter 9

Quiz 1

1. A string literal is any sequence of characters enclosed in double quotes.
2. String constant, string value and string.
3. `'\0'`
4. `'\n'`

Quiz 2

1. It copies `str2` to `str1`, including the `'\0'`.
2. It returns the uppercase equivalent if the character is lowercase; otherwise, it returns the character unchanged.
3. `ctype.h`
4. `stdlib.h`

Quiz 3

1. The angle brackets, <>, tell the compiler to begin searching for the included file in the C compiler system library directory, while the double quotes, "", tell the compiler to start looking in the default directory where the program file is located.
2. right
3. The statement `printf("|%-25.12s|","Have a Happy Day");` causes 12 characters to be left justified in a field of 25 characters.
4. True

Chapter 10

Quiz 1

1. A file is a collection of data that is stored together under a common name, usually on a disk, magnetic tape, or CD-ROM.
2. A file stream is a one-way transmission path that is used to connect a file stored on a physical device, such as a disk or CD-ROM, to a program.
3. Asterisk
   *
   asterisk (*)
4. NULL

Quiz 2

1. `rewind()`
2. The `fseek()` function allows the programmer to move to any position in the file.
3. `stdio.h`
4. When opening a file for output, if the name of an existing data file is used with `fopen()`, the file will be destroyed when it is opened in write mode.

Quiz 3

1. Files that are referred to as binary files store numerical values using the computer's internal numerical code.
2. A disadvantage is that the file can no longer be inspected using either a word processing or text editing program, which means that the ability to see the numerical values as textual information is lost.
3. `'b'`
4. address
   `&`

Chapter 11

Quiz 1

1. addresses
2. One unique feature of pointers is that offsets may be included in expressions using pointers. For example, the `1` in the expression `*(gPtr + 1)` is an offset. The offset is the number of variables to skip over.
3. constant
4. A pointer access can always be replaced using subscript notation. For example, if `numPtr` is declared as a pointer variable, the expression `*(numPtr + i)` can also be written as `numPtr[i]`. This is true even though `numPtr` is not created as an array.

Quiz 2

1. A pointer, constructed either as a variable or function parameter, contains a value. With pointers, however, the stored value is an address. Thus, by adding numbers to and subtracting numbers from pointers, we can obtain different addresses.
2. data type
3. When numbers are added to pointers, a correct scaling is automatically accomplished because the compiler converts the arithmetic operation `pointer + number` to `pointer + number * sizeof(data type being pointed to).`
4. address

Quiz 3

1. `nums[1][2]`
2. The main difference in the definitions of `message1` as an array and `message2` as a pointer is the way the pointer is created. Defining `message1` using the declaration `static char message1[81]` explicitly calls for a fixed amount of storage for the array. This causes the compiler to create a pointer constant. Defining `message2` using the declaration `char *message2` explicitly creates a pointer variable first. This pointer is then used to hold the address of a string when the string is actually specified. This difference in definitions has both storage and programming consequences (see Figure 11.12).
3. `int (*calc)()`

4. The declaration `char *seasons[4];` creates an array of four elements, where each element is a pointer to a character.

Chapter 12

Quiz 1

1. A structure's form consists of the symbolic names, data types, and arrangement of individual data fields in the record. The structure's contents consist of the actual data stored in the symbolic names.
2. populating
3. The difference is best explained with an example. An array is a homogeneous data structure, which means that each of its components must be of the same type. A record is a heterogeneous data structure, which means that each of its components can be of different data types.
4. `static`

Quiz 2

1. `typedef`
2. Parallel arrays are two or more arrays, where each array has the same number of elements and the elements in each array are directly related by their position in the arrays.
3. The problem with parallel arrays is that the correspondence between data items is easily lost if only one of the arrays is reordered.
4. inner

Quiz 3

1. scalar
2. address
3. A union is a data type that reserves the same area in memory for two or more variables, each of which can be a different data type.
4. A union reserves sufficient memory locations to accommodate its largest member's data type.

Chapter 13

Quiz 1

1. A linked list is a set of structures in which each structure contains at least one member whose value is the address of the next logically ordered structure in the list.
2. In a linked list, rather than requiring each structure to be physically stored in the proper order, each new structure is physically added either to the end of the existing list, or wherever the computer has free space in its storage area. The structures are "linked" together by including the address of the next structure in the structure immediately preceding it. From a programming standpoint, the current structure being processed contains the address of the next structure, no matter where the next structure is actually stored. Such structures are also known as self-referencing structures.
3. NIL
4. `(*t1.nextaddr).name`

Quiz 2

1. C provides the four functions, `malloc()`, `calloc()`, `realloc()` and `free()`, to control the dynamic allocation and release of memory space.
2. It reserves the number of bytes requested by the argument passed to the function. It returns the address of the first reserved location as an address of a `void` data type, or NULL if sufficient memory is not available.
3. stack
4. POP

Quiz 3

1. Items are removed from a queue in the order in which they were entered. Thus, a queue is a first in, first out (FIFO) structure.
2. Placing a new item on top of the queue is formally referred to as enqueueing, and the operation of removing an item from a queue is formally referred to as serving.
3. dynamically linked list
4. INSERT

Chapter 14

Quiz 1

1. conditional compilation
2. The conditional operator, `?:`, is unique in C in that it is a ternary operator.
3. `typedef`
4. Enumerated lists are identified by the reserved word `enum` followed by an optional, user-selected name and a required list of one or more constants.

Quiz 2

1. The operators that are used to perform bit operations in C are known as bit operators. The bit operators in C are `&`, `|`, `^`, `~`, << and >>.
2. filter
   pass
3. arithmetic
4. The exclusive `OR` operator, `^`, performs a bit-by-bit comparison of its two operands. The result of the comparison is determined by the following rule: The result of an exclusive OR comparison is 1 if one and only one of the bits being compared is a 1; otherwise the result is 0.

Quiz 3

1. 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, and the substitution of the text in place of the symbolic name is called a macro expansion or macro substitution.
2. They are arguments that are typed on the command line.
3. macro
4. string