Arrays, Files, and Menu-Driven Programs

Introduction to Programming : Arrays, Files, and Menu-Driven Programs

Lesson 3 Overview

In this lesson, you'll learn about arrays and menu-driven programs. First, you'll learn about arrays and the various types of arrays you can create. You'll learn how to use algorithms to sort and search arrays. Then, you'll learn about file input and output and how to process files. And finally, you'll learn about menu-driven programs and how you can use them in your programs.

3.1 Describe the use of arrays in programming languages

Arrays

READING ASSIGNMENT

Read this assignment. Then, read Chapter 8 of your textbook.

Arrays are lists of variables in the computer's memory that have the same name. They're distinguished from each other by subscripts or numbers that indicate the variable's position in the array. An example of an array is a set of file folders in a file cabinet that's alphabetized from A through Z.

Storage locations within an array are known as **elements.** Each element in an array is assigned a unique number known as a **subscript** or **index**. Using the subscripts is how you can access the individual elements in an array.

You can input and output array contents of an array element just as you would a regular variable. It's also possible to use a loop to step through an entire array and perform the same operation on each variable. Your textbook explains how to do so as well as how to do the following:

- Process elements of an array
- Initialize an array with values when you declare it
- Perform **array bounds checking**, which means the programming language won't allow a program to use an invalid array subscript
- Watch for **off-by-one errors**, which result when a loop iterates one time too many or one time too few
- Create **partially filled arrays**, which may be useful if you need to store a series of items in an array but you don't know the exact number of items in the series

You'll also learn how to search for data stored in an array. When searching for an array to find a value, you must do the following:

1. Create a variable.
2. Use a loop to test each array element.
3. Set a flag for when a match is found.

There are many operations you can perform on an array using a loop:

- Totaling or averaging the values in an array
- Finding the highest or lowest value in an array
- Passing an array as an argument to a module or function

Some special types of arrays include the following:

- **Parallel arrays** involve two arrays in which each element in the first array is associated with the element in the same position in the second array.
- **Two-dimensional arrays** store multiple data sets in several identical arrays.
- **Multidimensional arrays** model data in multiple sets.

Make sure to answer the checkpoints in the textbook as you read the chapter. Check your answers in Appendix E at the back of your textbook.

Key Points and Links

READING ASSIGNMENT

Key Points

- Arrays are lists of variables in the computer's memory that have the same name.
- Storage locations within an array are known as elements, and each element is assigned a unique number known as a subscript or index.
- You can input and output array contents of an array element just as you would a regular variable, or use a loop to step through an entire array and perform the same operation on each variable.
- When searching for an array to find a value, you must create a variable, use a loop to test each array element, and set a flag for when a match is found.
- Some special types of arrays include parallel arrays, two-dimensional arrays, and multidimensional arrays.

Exercise: Arrays

Answer the Review Questions and the following exercises at the end of Chapter 8 in your textbook:

- **Algorithm Workbench 1, 3, and 8**
- **Debugging Exercises 1 and 2**
- **Programming Exercise 3**

Exercise Answer Key:

Exercise: Arrays

Review Questions

Multiple Choice

1. B
2. D
3. A
4. B
5. C
6. D
7. A
8. B
9. C
10. C

True or False

1. False
2. True
3. True
4. False
5. False

Short Answer

1. This is an error that occurs when a loop iterates one too many times or one too few.
2. (a) 10

   (b) 0

   (c) 9

3. (a) 3

   (b) 1
4. It would be stored in balances[187].
5. (a) 8

   (b) 10

   (c) 80

   (d) Set sales[7][9] = 0

Answers for the Algorithm Workbenches, Debugging Exercises, and Programming Exercise are in the CSC105 Lesson 3 Additional Exercise Answers supplement.

3.2 Explain the process of sorting arrays

Sorting and Searching Arrays

READING ASSIGNMENT

Read this assignment. Then, read Chapter 9 of your textbook.

Many programming tasks require that data in an array be sorted in some order. The order you choose may be **ascending** (for example, A to Z) or **descending** (for example, Z to A).

You can use a **sorting algorithm** to step through an array and rearrange the contents:

- **Bubble sort** makes passes through and compares elements, and certain values "bubble" toward the end of the array, one at a time, with each pass.

- **Selection sort** performs fewer swaps than the bubble sort because it moves items immediately to their final position in the array.
- **Insertion sort** sorts using pairs of elements in an array until all the elements have been inserted in their proper position.

For more information about sorting algorithms, watch the following videos:

- HackerRank "Algorithms: Bubble Sort" (www.youtube.com/embed /6Gv8vg0kcHc)
- CS50 "Selection Sort" (www.youtube.com/embed/3hH8kTHFw2A)
- CS50 "Insertion Sort" (www.youtube.com/embed/O0VbBkUvriI)

Sometimes when you sort records, you may have to swap two values, in which case you can create a temporary variable to hold one of the variables while you swap them.

The **binary search algorithm** finds items in an array by dividing the array in half multiple times. Through each division, it removes half of the array that doesn't contain the item for which you're searching. It's more efficient than a sequential search.

Make sure to answer the checkpoints in the textbook as you read the chapter. Check your answers in Appendix E at the back of your textbook.

Key Points and Links

READING ASSIGNMENT

Key Points

- Many programming tasks require that data in an array be sorted in some order.
- You can use a sorting algorithm to step through an array and rearrange the contents.
- Sorting algorithms include bubble sort, selection sort, and insertion sort.
- Sometimes, when you sort records, you may have to swap two values, in which case you can create a temporary variable to hold one of the variables while you swap them.
- The binary search algorithm finds items in an array by dividing the array in half multiple times.

Links

- HackerRank "Algorithms: Bubble Sort" (www.youtube.com/embed/6Gv8vg0kcHc)
- CS50 "Selection Sort" (www.youtube.com/embed/3hH8kTHFw2A)
- CS50 "Insertion Sort" (www.youtube.com/embed/O0VbBkUvriI)

Exercise: Sorting and Searching Arrays

Answer the Review Questions and the following exercises at the end of Chapter 9 in your textbook:

- **Algorithm Workbench 2, 3, and 4**
- **Debugging Exercise 1**
- **Programming Exercise 3**

Exercise Answer Key:

Exercise: Sorting and Searching Arrays

Review Questions

Multiple Choice

1. B
2. C
3. D
4. A
5. B
6. C
7. A
8. B
9. A
10. B

True or False

1. True
2. False

3. False
4. True
5. False

Short Answer

1. 10,000
2. *n*/2 times
3. One time
4. 10
5. Because values move by only one element at a time toward their final destination in an array
6. It usually performs fewer swaps because it moves items immediately to their final position in the array.
7. (a) Move 1 to element zero. The array now looks like this:
   1, 4, 3, 2
   (b) Move 2 to element one. The array now looks like this:
   1, 2, 3, 4
   (c) The values are now in order. No further swaps are made.
8. (a) Swap the 4 and the 1. The array now looks like this:
   1, 4, 3, 2
   (b) Move the 3 so that it appears before the 4. The array now looks like this:
   1, 3, 4, 2
   (c) Move the 2 so that it appears before the 3. The array now looks like this:
   1, 2, 3, 4

Answers for the Algorithm Workbenches, Debugging Exercise, and Programming Exercise are in the CSC105 Lesson 3 Additional Exercise Answers supplement.

3.3 Identify the use of files and records

Files

READING ASSIGNMENT

Read this assignment. Then, read Chapter 10 of your textbook.

Program data can be saved to files so you can use them at a later time. Files are saved on disks, such as a computer's hard disk. Programmers refer to the process of saving data to files as **writing** data to the file (**output file**); when they retrieve it, they **read** data from the file (**input file**).

To use a file, a program must do the following:

1. Open the file.
2. Process the file.
3. Close the file.

The two types of files are text and binary. **Text files** contain data that has been encoded as text, and **binary files** contain data that hasn't been converted to text. There are two different ways to access data stored in a file:

1. **Sequential access files** must be read from the beginning to the end; you can't jump to the desired data.
2. **Direct access files** or **random access files** can be read from any point.

Files are identified by a filename followed by a **filename extension**, which identifies the type of data stored in the file (such as ***.jpg**, ***.txt**, and so on). When writing a program that performs an operation on a file, you'll work with the filename that identifies the file on the computer's disk as well as an internal name that's similar to a variable name. Your textbook shows you how to do the following using pseudocode:

- Open a file
- Write data to a file
- Close an output and input file
- Read data from a file
- **Append** (add) data to the end of the data that already exists in a file

A **delimiter** is a predefined character or set of characters that's written to a file after each item to indicate the end of a piece of data. An **end-of-file (EOF) marker** indicates where the file's contents end.

Since files can contain a lot of data, programs often use loops to process the data. Your textbook explains how to read and detect an end of a file with a loop. You'll also learn how to write a loop that saves the contents of an array to a file.

File data is often stored in **records**, which contain a set of data for an item. Each piece of data in the item record is known as a **field**. For example, a university file might contain student information in records. Each record is for an individual student; each item about a student, such as his or her first name or address, is a field.

There's usually a **file specification document** for each data file that describes the fields and data types that are stored in a particular file. A programmer can review this document to understand how data is organized inside the file. An example of a file specification document is in Figure 10-20 of your textbook.

Your textbook also explains how to write and read records using pseudocode. You'll learn how to add, display, search, modify, and delete records as well.

In **control break logic**, a program's processing is paused to perform a different action. The pause occurs when a control variable is changed. A control break occurs when a program's logic takes a temporary detour.

For example, if you create an inventory for a bookstore with books arranged by genres and you have a subtotal for the retail amount of each category, you've created a **control break report**. Another example of a control break is placing page breaks in your report after each group of data, such as a book genre. Control break reports must be arranged in sequential order.

If you're comparing data from one record to see if it matches data from another, you can create a **control break field**, which is a special variable that "remembers" the data to which a variable is being compared.

Print spacing charts are paper sheets with grids on them. The paper is divided up in grid form, and each square holds a character. Programmers often use **X**s to represent characters in variables and **9**s to represent digits in variables.

Make sure to answer the checkpoints in the textbook as you read the chapter. Check your answers in Appendix E at the back of your textbook.

Key Points and Links
READING ASSIGNMENT

Key Points

- Programmers refer to the process of saving data to files as writing data to the file (output file); when they retrieve it, they read data from the file (input file).
- To use a file, a program must open the file, process the file, and close the file.
- Text files contain data that have been encoded as text, and binary files contain data that haven't been converted to text.
- Sequential access files must be read from the beginning to the end, while direct access files or random access files can be read

from any point.

- Since files can contain a lot of data, programs often use loops to process the data.
- File data is often stored in records, which contain a set of data for an item; each piece of data in the item record is known as a field.
- A programmer can review a file specification document to understand how data is organized inside the file.
- In control break logic, a control variable is changed, which pauses a program's processing and makes a different action occur.

Exercise: Files

Answer the Review Questions and the following exercises at the end of Chapter 10 in your textbook:

- **Algorithm Workbench 1, 3, and 8**
- **Debugging Exercise 1**
- **Programming Exercise 1**

Exercise Answer Key:
Exercise: Files

Review Questions

Multiple Choice

1. B

2. A
3. D
4. C
5. A
6. B
7. D
8. C
9. A
10. B
11. D
12. D
13. B
14. C
15. A

True or False

1. False
2. True
3. False
4. False
5. True
6. False
7. False

Short Answer

1. (1) Open the file—Opening a file creates a connection between

the file and the program. Opening an output file usually creates the file on the disk and allows the program to write data to it. Opening an input file allows the program to read data from the file.

(2) Process the file—In this step, data is either written to the file (if it's an output file) or read from the file (if it's an input file).

(3) Close the file—When the program is finished using the file, the file must be closed. Closing a file disconnects the file from the program.

2. Closing a file disconnects the program from the file. In some systems, failure to close an output file can cause a loss of data. This is because the data that's written to a file is first written to a **buffer**, which is a small "holding section" in the memory. When the buffer is full, the computer's operating system writes the buffer's contents to the file. This technique increases the system’s performance because writing data to memory is faster than writing it to a disk. The process of closing an output file forces any unsaved data that remains in the buffer to be written to the file.
3. A file’s read position marks the location of the next item that will be read from the file. When a file is opened, its read position is initially set to the first item in the file.
4. The existing contents are preserved.
5. The file is created.
6. It returns True if the end of a file has been reached, or False.
7. In control break logic, the program performs some ongoing task

(such as processing the items in a file) but temporarily interrupts the task when a control variable reaches a specific value or changes its value. When this happens, some other action is performed, and then the program resumes its ongoing task.

Answers for the Algorithm Workbenches, Debugging Exercise, and Programming Exercise are in the CSC105 Lesson 3 Additional Exercise Answers supplement.

3.4 State the process of creating a menu-driven program

Menu-Driven Programs

READING ASSIGNMENT

Read this assignment. Then, read Chapter 11 of your textbook.

In a **menu-driven program**, a **menu** (list of operations) is displayed on the screen. Users can then select which program they wish to run.

Programmers should make sure the user's selection is valid. For instance, if a program asks a user to select one of four options by entering **1, 2, 3**, or **4**, it should reject a user input of, for example,**5** or **11.**

The program uses a **decision structure** to perform whichever action the user selects. A **case structure** is often used in menu-driven programs since it's the simplest approach. Other options, such as an

If-Then-Else statement, can also be used. A menu-driven program must also validate the user's selection.

Since many tasks can be written into a menu-driven program, programmers should modularize the programs. **Modularization** refers to breaking down each task into a separate module. Additionally, programmers should include a loop to repeat the menu since the program may need to be run several times.

If a program has only one menu, it's called a **single-level menu**. If a menu has a main menu and additional submenus, it's called a **multiple-level menu**, which is useful for programs that have multiple list items.

Make sure to answer the checkpoints in the textbook as you read the chapter. Check your answers in Appendix E at the back of your textbook.

Key Points and Links

READING ASSIGNMENT

Key Points

- In a menu-driven program, a menu (list of operations) is displayed on the screen. Users can then select which program they wish to run.
- Programmers should make sure the user's menu selection is valid.
- The program uses a decision structure to perform whichever

action the user selects.
- Since many tasks can be written into a menu-driven program, programmers should modularize the programs (divide each task into a separate module).
- Programmers should include a loop to repeat the menu, since the program may need to be run several times.
- If a program has only one menu, it’s called a single-level menu; if a menu has a main menu and additional submenus, it’s called a multiple-level menu.

Exercise: Menu-Driven Programs

Answer the Review Questions and the following exercises at the end of Chapter 11 in your textbook:

- **Algorithm Workbench 1, 2, 3, and 4**
- **Programming Exercise 1**

Exercise Answer Key:
Exercise: Menu-Driven Programs

Review Questions

Multiple Choice

1. C
2. D
3. A

4. B
5. C
6. B
7. A

True or False

1. False
2. False
3. True
4. True
5. False

Short Answer

1. You use a decision structure of some type. You might choose a case structure, nested **If-Then-Else** statements, or an **If-Then-Else-If** statement.
2. If you're using a case structure to perform the selected menu actions, you can have a Default section that handles invalid menu selections. If you use nested **If-Then-Else** statements, you can have an **Else** clause that handles invalid menu selections.
3. You use a loop that redisplays the menu after the user's selected action has been performed.
4. In a program that uses a single-level menu, all of the menu selections fit nicely in a single menu. When the user selects an operation from a single-level menu, the program immediately performs that operation, and then the program redisplays the

menu (or the program ends if it doesn't use a loop to redisplay the menu). A program that uses a multiple-level menu typically displays a main menu when the program starts, showing only a few items, and then displays smaller submenus when the user makes a selection.

5. Users often have trouble sorting through the items in a menu when given too many choices.

Answers for the Algorithm Workbenches and Programming Exercise are in the CSC105 Lesson 3 Additional Exercise Answers supplement.

3.5 Create a basic menu-driven program

CSC105 Graded Project 3

READING ASSIGNMENT

Your project must be submitted as a zipped/compressed (*.zip) file that includes the following files:

- Text file (*.txt) of your pseudocode
- Screenshot(s) in JPEG format (*.jpg) of your flowchart
- A Rich Text Format (*.rtf) or Microsoft Word (*.doc) file that lists the following information:
    - Your name
    - Your student ID number
    - The exam number
    - Your email address

For information on how to take and save a screenshot on your computer or to zip files, review the instructions for CSC105 Graded Project 1. Save your compressed file as **[Your Name]_CSC105_GradedProject3.** Your project will be individually graded by your instructor and therefore may take up to five to seven days to grade. To submit your graded project, follow these steps:

- Log into your student portal.
- Click **Take Exam** next to the lesson you're working on.
- Find the exam number for your project at the top of the Project Upload page.
- Follow the instructions provided to complete your exam.

Be sure to keep a backup copy of any files you submit to the school!

Introduction

You'll apply the concepts of Lesson 3 to create a basic menu-driven program.

Instructions

You'll create both pseudocode and a flowchart to design an application that displays the following menu:

```
Select a Planet

1. Mercury
2. Venus
3. Earth
4. Mars
5. Exit the program

Enter your selection.
```

When the user selects a planet from the menu, the program should display the following information about the planet:

Mercury	Average distance from the sun	57.9 million kilometers
	Mass	3.31 × 10^23kg
	Surface temperature	–173 to 430 degrees Celsius
Venus	Average distance from the sun	108.2 million kilometers
	Mass	4.87 × 10^24kg
	Surface temperature	472 degrees Celsius
Earth	Average distance from the sun	149.6 million kilometers
	Mass	5.967 × 10^24kg
	Surface temperature	–50 to 50 degrees Celsius
Mars	Average distance from the sun	227.9 million kilometers
	Mass	0.6424 × 10^24kg
	Surface temperature	–140 to 20 degrees Celsius

Review Appendices B and C in your textbook for guidance when working on your project. Use free trials of any of the programs listed in CSC105 Graded Project 1 to create the flowchart. Write your pseudocode in a plain-text editor such as Notepad or TextEdit and save as a text file (*.txt). Also, save a screenshot of your flowchart as a JPEG file (*.jpg).

Grading Criteria

Your instructor will use the following guidelines to grade your project.

The menu-driven program is designed to effectively show a menu from which the user can make a selection and see the correct display of the corresponding data.	34 points
The pseudocode is accurate and in the correct format.	33 points
The flowchart is accurate and uses the appropriate shapes.	33 points
TOTAL	**100 points**

Lesson 3 Review

Self-Check

1. An individual element in an array is identified by its

a. integer.

b. value.

c. size.

d. subscript.

2. Two-dimensional arrays can be thought of as containing

a. lines and boxes.

b. elements and columns.

c. rows and pages.

d. rows and columns.

3. What's the term used for the number inside the brackets of an array that specifies how many values the array can hold?

a. Subscript declarator

b. Size initialization

c. Size declarator

d. Number declarator

4. Which of the following array declarations would be *best* suited for storing prices of items sold in a store?

a. **Declare String itemPrice[SIZE]**

b. **Declare itemPrice[Real]**

c. **Declare Real itemPrice[SIZE]**

d. **Declare Integer itemPrice[SIZE]**

5. How many subscripts do you need to access one element in a two-dimensional array?

a. Three

b. One

c. Two

d. None

6. When elements in an array are sorted from lowest to highest, the array is sorted in _______ order.

a. numeric

b. descending

c. ascending

d. alphabetic

7. The _______ search algorithm uses three variables to mark positions within the array as it searches for the desired value.

a. selection

b. bubble

c. insertion

d. binary

8. In the pseudocode of the textbook, which of the following variables in a selection sort holds the subscript of the element with the smallest value found in the scanned area of the array?

a. **startScan**

b. **minValue**

c. **SIZE**

d. **minIndex**

9. Which of the following statements is *true* after the following statements are executed? **Set x = y**
Set y = x

a. **x** and **y** contain their original values.

b. **y** contains the value in **x** and **x** stayed the same.

c. **x** and **y** have swapped their values.

d. **x** contains the value in **y** and **y** stayed the same.

10. In the following bubble sort that's sorting an array named scores in ascending order, what would be the index of the element with the value **93** at the end of the first outer pass? **Declare Integer scores[6] = 89, 52, 93, 78, 86**

a. **0**

b. **5**

c. **4**

d. **3**

11. When a web page is visited, the browser may store a small file on the user's computer. This file is known as a

a. temp file.

b. cookie.

c. web file.

d. biscuit.

12. To process the data in a file, you must first _______ the file and then read the data into memory.

a. declare

b. loop

c. open

d. input

13. Which of the following statements about the **eof** function is *true?*

a. It accepts a file's internal name as an argument and returns **False** if the end of the file has been reached.

b. It accepts the name of the file on disk as an argument and returns **False** if the end of the file has been reached.

c. It accepts the name of the file on disk as an argument and returns **True** if the end of the file has been reached.

d. It accepts a file's internal name as an argument and returns **True** if the end of the file has been reached.

14. Given the following pseudocode, what's the name of the file on disk? **Declare String item**

```
Declare Integer numOrdered
Declare InputFile stuffBought
Open stuffBought "inventory.dat"
Display "Your orders:"
```

```
While NOT eof(stuffBought)
    Read stuffBought item, numOrdered
    Display item, ": " , numOrdered
End While
Close stuffBought
```

a. **item**

b. **stuffBought**

c. **numOrdered**

d. **inventory.dat**

15. A character or set of characters that marks the end of a piece of data is known as a/an

a. delimiter.

b. end marker.

c. boundary.

d. median.

16. A menu-driven program that has a main menu and several submenus is known as a/an _______ menu.

a. parent-child

b. standard

c. inheritance

d. multiple-level

17. When a user makes a selection from the main menu in a multiple-level menu, a _______ would probably be displayed next.

a. new main menu

b. submenu

c. number

d. grand menu

18. A list of operations that's displayed on the screen, allowing the user to select one of these options, is known as a/an

a. selector.

b. list.

c. menu.

d. operator.

19. A/An _______ statement provides a default section to validate a user's menu selection.

a. **If-Then-Else**

b. **Case**

c. **While**

d. **If-Then**

20. In a _______ loop, the menu will be displayed at least once.

a. **Case**

b. **For**

c. **Do-While**

d. **While**

Self-Check Answer Key

1. subscript.
   Explanation: An individual element in an array is identified by its subscript.
   Reference: Section 3.1

2. rows and columns.
   Explanation: Two-dimensional arrays can be thought of as containing rows and columns.
   Reference: Section 3.1

3. Size declarator
   Explanation: *Size declarator* is the term used for the number inside the brackets of an array that specifies how many values the array can hold.
   Reference: Section 3.1

4. **Declare Real itemPrice[SIZE]**
   Explanation: The array declaration **Declare Real itemPrice[SIZE]** would be best suited for storing prices of items sold in a store.
   Reference: Section 3.1

5. Two
   Explanation: You need two subscripts to access one element in a two-dimensional array.
   Reference: Section 3.1

6. ascending
   Explanation: When elements in an array are sorted from lowest to highest, the array is sorted in ascending order.
   Reference: Section 3.2

7. binary

   Explanation: The binary search algorithm uses three variables to mark positions within the array as it searches for the desired value.

   Reference: Section 3.2

8. **minIndex**

   Explanation: In the pseudocode of the textbook, the variable **minIndex** in a selection sort holds the subscript of the element with the smallest value found in the scanned area of the array.

   Reference: Section 3.2

9. **x** contains the value in **y** and **y** stayed the same.

   Explanation: After the following statements are executed, **x** contains the value in **y** and **y** stayed the same: **Set x = y**
   Set y = x

   Reference: Section 3.2

10. **4**

    Explanation: In the following bubble sort that's sorting an array named scores in ascending order, **4** would be the index of the element with the value **93** at the end of the first outer pass:

    Declare Integer scores[6] = 89, 52, 93, 78, 86

    Reference: Section 3.2

11. cookie.

    Explanation: When a web page is visited, the browser may store a small file on the user's computer. This file is known as a *cookie.*

    Reference: Section 3.3

12. open

    Explanation: To process the data in a file, you must first open the file and then read the data into memory.

    Reference: Section 3.3

13. It accepts a file's internal name as an argument and returns **True** if the end of the file has been reached.

    Explanation: An **eof** function accepts a file's internal name as an argument and returns **True** if the end of the file has been reached.

    Reference: Section 3.3

14. **inventory.dat**

    Explanation: Given the following pseudocode, the name of the file on disk is **inventory.dat**:

```
Declare String item
Declare Integer numOrdered
Declare InputFile stuffBought
Open stuffBought "inventory.dat"
Display "Your orders:"
While NOT eof(stuffBought)
    Read stuffBought item, numOrdered
    Display item, ": " , numOrdered
End While
Close stuffBought
```

Reference: Section 3.3

15. delimiter.

    Explanation: A character or set of characters that marks the end of a piece of data is known as a delimiter.

    Reference: Section 3.3

16. multiple-level

    Explanation: A menu-driven program that has a main menu and several submenus is known as a multiple-level menu.

    Reference: Section 3.4

17. submenu

    Explanation: When a user makes a selection from the main menu in a multiple-level menu, a submenu would probably be displayed next.

    Reference: Section 3.4

18. menu.

    Explanation: A list of operations that's displayed on the screen, allowing the user to select one of these options, is known as a menu.

    Reference: Section 3.4

19. **Case**

    Explanation: A **Case** statement provides a default section to validate a user's menu selection.

    Reference: Section 3.4

20. **Do-While**

    Explanation: In a **Do-While** loop, the menu will be displayed at least once.

    Reference: Section 3.4

Flash Cards

1. Term: Parallel Arrays

Definition: Involve two arrays in which each element in the first array is associated with the element in the same position in the second array

2. Term: Two-Dimensional Arrays

Definition: Store multiple data sets in several identical arrays

3. Term: Subscript

Definition: A unique element assigned to each element in an array

4. Term: Elements

Definition: Storage locations within an array

5. Term: Bubble Sort

Definition: Makes passes through and compares elements; certain values move toward the end of the array, one at a time, with each pass

6. Term: Insertion Sort

Definition: Sorts using pairs of elements in an array until all the elements have been inserted in their proper position

7. Term: Binary Search Algorithm

Definition: Finds items in an array by dividing the array in half multiple times and removing half of the array that doesn’t contain the item for which you’re searching

8. Term: Selection Sort

Definition: Moves items immediately to their final position in the array

9. Term: Sequential Access Files

Definition: Saved data that must be read from the beginning to the end

10. Term: Control Break Logic

Definition: The interruption of a program's regular processing to perform a different action when a control variable's value changes or the variable acquires a specific value

11. Term: Delimiter

Definition: A predefined character or set of characters that's written to a file after each item to indicate the end of a piece of data

12. Term: File Specification Document

Definition: Describes the fields and data types that are stored in a particular file

13. Term: Single-Level Menu

Definition: A menu system with one list of operations

14. Term: Multiple-Level Menu

Definition: A menu with a main list of operations that display smaller sublists when a user makes a selection

15. Term: Menu-Driven Program

Definition: A program that displays a list of operations that the user can make selections from