120

60/60

School of Computing and Information Technologies

PROGCON - CHAPTER 2

CB: 29_Teves

CLASS NUMBER: 09 SECTION: AC 192

NAME: Gregorio, Bryan James C. DATE: 11/8/19

20

**PART 1: Identify the following.**

Data Type 1. A classification that describes what values can be assigned, how the variable is stored, and what types of operations can be performed with the variable.

Hierarchy chart 2. A diagram that illustrates modules' relationships to each other.

Data Dictionary 3. A list of every variable name used in a program, along with its type, size, and description.

Functional Cohesion 4. A measure of the degree to which all the module statements contribute to the same task.

Prompt 5. A message that is displayed on a monitor to ask the user for a response and perhaps explain how that response should be formatted.

Portable 6. A module that can more easily be reused in multiple programs.

Floating-Point 7. A number with decimal places.

• Identifier 8. A program component's name.

(Literal) Numeric Constant 9. A specific numeric value.

Declaration 10. A statement that provides a data type and an identifier for a variable.

Hungarian notation 11. A variable-naming convention in which a variable's data type or other information is stored as part of its name.

Integer 12. A whole number.

Binary Operator 13. An operator that requires two operands—one on each side.

- Magic Number 14. An unnamed constant whose purpose is not immediately apparent.

Assignment Statement 15. Assigns a value from the right of an assignment operator to the variable or constant on the left of the assignment operator.

Alphanumeric Values 16. Can contain alphabetic characters, numbers, and punctuation.

- Key words 17. Constitute the limited word set that is reserved in a language.

Module's Body 18. Contains all the statements in the module.

Annotation Symbol 19. Contains information that expands on what appears in another flowchart symbol; it is most often represented by a three-sided box that is connected to the step it references by a dashed line.

Self-Documenting 20. Contains meaningful data and module names that describe the program's purpose.

20

[illegible]-associativity and right associativity 21. Describe operators that evaluate the expression to the right first.

Numeric 22. Describes data that consists of numbers.

Left-to-right Associativity 23. Describes operators that evaluate the expression to the left first.

Overhead 24. Describes the extra resources a task requires.

Order of operations 25. Describes the rules of precedence.

In scope 26. Describes the state of data that is visible.

Garbage 27. Describes the unknown value stored in an unassigned variable.

Local 28. Describes variables that are declared within the module that uses them.

Global 29. Describes variables that are known to an entire program.

Rules of Precedence 30. Dictate the order in which operations in the same statement are carried out.

External Documentation 31. Documentation that is outside a coded program.

Internal Documentation 32. Documentation within a coded program.

Real Numbers 33. Floating-point numbers.

end-of-job tasks 34. Hold the steps you take at the end of the program to finish the application.

Housekeeping Tasks 35. Include steps you must perform at the beginning of a program to get ready for the rest of the program.

Detail loop tasks 36. Include the steps that are repeated for each set of input data.

Module header 37. Includes the module identifier and possibly other necessary identifying information.

-Lower camel casing 38. Is another name for the camel casing naming convention.

-Kebab Case 39. Is sometimes used as the name for the style that uses dashes to separate parts of a name.

Modules return statement 40. Marks the end of the module and identifies the point at which control returns to the program or module that called the module.

Numeric Variable 41. One that can hold digits, have mathematical operations performed on it, and usually can hold a decimal point and a sign indicating positive or negative.

Main Program 42. Runs from start to stop and calls other modules.

Named constant 43. Similar to a variable, except that its value cannot change after the first assignment.

Modules 44. Small program units that you can use together to make a program; programmers also refer to modules as subroutines, procedures, functions, or methods.

Initializing a variable 45. The act of assigning its first value, often at the same time the variable is created.

Encapsulation 46. The act of containing a task's instructions in a module.

Functional Decomposition 47. The act of reducing a large program into more manageable modules.

Echoing input 48. The act of repeating input back to a user either in a subsequent prompt or in output.

Assignment operator 49. The equal sign; it is used to assign a value to the variable or constant on its left.

Reusability 50. The feature of modular programs that allows individual modules to be used in a variety of applications.

reliability 51. The feature of modular programs that assures you a module has been tested and proven to function correctly

camel casing 52. The format for naming variables in which the initial letter is lowercase, multiple-word variable names are run together, and each new word within the variable name begins with an uppercase letter.

Pascal casing 53. The format for naming variables in which the initial letter is uppercase, multiple-word variable names are run together, and each new word within the variable name begins with an uppercase letter.

Mainline Logic 54. The logic that appears in a program's main module; it calls other modules.

L value 55. The memory address identifier to the left of an assignment operator.

Modularization 56. The process of breaking down a program into modules.

Abstraction 57. The process of paying attention to important properties while ignoring nonessential details.

call a module 58. To use the module's name to invoke it, causing it to execute.

Program level 59. Where global variables are declared.

Program Comments 60. Written explanations that are not part of the program logic but that serve as documentation for those reading the program.

Choose from the following

1. Abstraction
2. Alphanumeric values
3. Annotation symbol
4. Assignment operator
5. Assignment statement
6. Binary operator
7. Call a module
8. Camel casing
9. Data dictionary
10. Data type
11. Declaration
12. Detail loop tasks
13. Echoing input
14. Encapsulation
15. End-of-job tasks
16. External documentation
17. Floating-point
18. Functional cohesion
19. Functional decomposition
20. Garbage
21. Global
22. Hierarchy chart
23. Housekeeping tasks
24. Hungarian notation
25. Identifier
26. In scope
27. Initializing the variable
28. Integer
29. Internal documentation
30. Kebob case
31. Keywords
32. Left-to-right associativity
33. Local
34. Lower camel casing
35. Lvalue
36. Magic number
37. Main program
38. Mainline logic
39. Modularization
40. Module body
41. Module header
42. Module return statement
43. Modules
44. Named constant
45. Numeric
46. Numeric constant (literal numeric constant)
47. Numeric variable
48. Order of operations
49. Overhead
50. Pascal casing
51. Portable
52. Program comments
53. Program level
54. Prompt
55. Real numbers
56. Reliability
57. Reusability
58. Right-associativity and right-to-left associativity
59. Rules of precedence
60. Self-documenting

School of Computing and Information Technologies

PROGCON - CHAPTER 2

30/32

CLASS NUMBER: 09

NAME: Gregorio, Bryan James C.

SECTION: AC192

DATE: 11/8/19

Checked by: Dan Bacarro III

**PART 2: Identify whether each variable name is valid, and if not explain why.**

5 a) Age Valid

Sorry? b) age_* Invalid. In a variable, no special character are allowed beside underscore (_).

5 c) +age invalid. This is considered a typo since + is considered an operator so putting + assumes that you're putting an invalid equation

5 d) age_ Valid

9 e) _age Valid

9 f) Age Valid

Sorry g) 1age Invalid. Variable names should start with A-Z or (_)

Sorry h) Age 1 Invalid. Spaces are not allowed as you are putting a single variable