



School of Computing and Information Technologies

PROGCON - CHAPTER 2

score: 59/60 = (116/120)
corrected by: gail oandatarian

CLASS NUMBER: #14

NAME: NEFALAK, ALAN PAUL D.

SECTION: AC102

DATE: 11/8/19

PART 1: Identify the following.

20

- Data Type
- Hierarchy chart
- Data Dictionary
- Functional cohesion
- Prompt
- portable
- Floating point
- Identifier
- unren constant
- Declaration
- gation Notation
- Integer
- iron operator
- Magin Number
- ignment Statement
- numeric values
- keywords
- idic body
- ation Symbol
- documenting
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.
 2. A diagram that illustrates modules' relationships to each other.
 3. A list of every variable name used in a program, along with its type, size, and description.
 4. A measure of the degree to which all the module statements contribute to the same task.
 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.
 6. A module that can more easily be reused in multiple programs.
 7. A number with decimal places.
 8. A program component's name.
 9. A specific numeric value.
 10. A statement that provides a data type and an identifier for a variable.
 11. A variable-naming convention in which a variable's data type or other information is stored as part of its name.
 12. A whole number.
 13. An operator that requires two operands—one on each side.
 14. An unnamed constant whose purpose is not immediately apparent.
 15. Assigns a value from the right of an assignment operator to the variable or constant on the left of the assignment operator.
 16. Can contain alphabetic characters, numbers, and punctuation.
 17. Constitute the limited word set that is reserved in a language.
 18. Contains all the statements in the module.
 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.
 20. Contains meaningful data and module names that describe the program's purpose.

- 29
21. Describe operators that evaluate the expression to the right first. *right-to-left associativity*
 22. Describes data that consists of numbers. *numeric*
 23. Describes operators that evaluate the expression to the left first. *left-to-right associativity*
 24. Describes the extra resources a task requires. *overhead*
 25. Describes the rules of precedence. *order of operations*
 26. Describes the state of data that is visible. *in scope*
 27. Describes the unknown value stored in an unassigned variable. *garbage*
 28. Describes variables that are declared within the module that uses them. *local*
 29. Describes variables that are known to an entire program. *global*
 30. Dictate the order in which operations in the same statement are carried out. *rules of precedence*
 31. Documentation that is outside a coded program. *External documentation*
 32. Documentation within a coded program. *Internal documentation*
 33. Floating-point numbers. *Real numbers*
 34. Hold the steps you take at the end of the program to finish the application. *end-of-job tasks*
 35. Include steps you must perform at the beginning of a program to get ready for the rest of the program. *housekeeping tasks*
 36. Include the steps that are repeated for each set of input data. *detail loop tasks*
 37. Includes the module identifier and possibly other necessary identifying information. *module header*
 38. Is another name for the camel casing naming convention. *lower camel casing*
 39. Is sometimes used as the name for the style that uses dashes to separate parts of a name. *kebab case*
 40. Marks the end of the module and identifies the point at which control returns to the program or module that called the module. *module return statement*
 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. *numeric variable*
 42. Runs from start to stop and calls other modules. *main program*
 43. Similar to a variable, except that its value cannot change after the first assignment. *named constant*
 44. Small program units that you can use together to make a program; programmers also refer to modules as subroutines, procedures, functions, or methods. *modules*
 45. The act of assigning its first value, often at the same time the variable is created. *initializing the variable*
 46. The act of containing a task's instructions in a module. *Encapsulation*
 47. The act of reducing a large program into more manageable modules. *functional decomposition*
 48. The act of repeating input back to a user either in a subsequent prompt or in output. *Echoing Input*
 49. The equal sign; it is used to assign a value to the variable or constant on its left. *assignment operator*
 50. The feature of modular programs that allows individual modules to be used in a variety of applications. *reusability*

- (10)
51. The feature of modular programs that assures you a module has been tested and proven to function correctly. *reliability*
 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. *camel 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. *Pascal casing*
 54. The logic that appears in a program's main module; it calls other modules. *mainline logic*
 55. The memory address identifier to the left of an assignment operator. *Lvalue*
 56. The process of breaking down a program into modules. *modularization*
 57. The process of paying attention to important properties while ignoring nonessential details. *abstraction*
 58. To use the module's name to invoke it, causing it to execute. *call a module*
 59. Where global variables are declared. *Program level*
 60. Written explanations that are not part of the program logic but that serve as documentation for those reading the program. *program comments*

Choose from the following

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

#16



School of Computing and Information Technologies
PROGCON - CHAPTER 2

32

checked by:
Hannarexy ♥

CLASS NUMBER: #16

SECTION:

NAME: NEPALAN, ALAN PAUL D.

DATE:

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

3 pts. a) Age ~~valid~~

5 pts. b) age_* Invalid
- no special characters allowed

5 pts. c) +age Invalid
- no special characters allowed

3 pts. d) age_ valid

3 pts. e) _age valid

3 pts. f) Age valid

5 pts. g) 1age Invalid
- a variable name can be constructed with digits and letters

5 pts. h) Age1 Invalid
- should be indicated with an underscore