#09

60

Checked by: [illegible] (Ritchie L)

School of Computing and Information Technologies

PROGCON - CHAPTER 2

CLASS NUMBER: 09 SECTION: TM181

NAME: DELA FUENTE, Marie Merose N. 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.

Numeric constants 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 name 16. Can contain alphabetic characters, numbers, and punctuation.

keywords 17. Constitute the limited word set that is reserved in a language.

module 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.

50

Right associativity and right to left 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.

Detailed 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.

Kebob case
39. Is sometimes used as the name for the style that uses dashes to separate parts of a name.

Module 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 the 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.

Lvalue 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

aka selective ignorance ↔ paying attention to impr. properties while ignoring

1. Abstraction
2. Alphanumeric values
3. Annotation symbol
4. Assignment operator — right to left — binary equal sign
5. Assignment statement
6. Binary operator
7. Call a module
8. Camel casing — hump in middle
9. Data dictionary — variables list
10. Data type — classification — < numeric string
11. Declaration — provides a data type & identifier for a variable
12. Detail loop tasks — core work
13. Echoing input
14. Encapsulation — one program in module
15. End-of-job tasks
16. External documentation
17. Floating-point (integer/real no.)
18. Functional cohesion — statement = same UOO
19. Functional decomposition — aka #39
20. Garbage — variables unknown value
21. Global — declared at the program level
22. Hierarchy chart
23. Housekeeping tasks
24. Hungarian notation
25. Identifier — variable name
26. In scope
27. Initializing the variable — start value
28. Integer
29. Internal documentation
30. Kebob case
31. Keywords
32. Left-to-right associativity
33. Local
34. Lower camel casing
35. Lvalue — result left of #4
36. Magic number — unnamed constant
37. Main program — aka basic steps of the program
38. Mainline logic
39. Modularization — large program into modules
40. Module body
41. Module header
42. Module return statement
43. Modules — subunit of program problem
44. Named constant — value only once
45. Numeric (numbers)
46. Numeric constant (literal numeric constant) — data form
47. Numeric variable — holds digits, main operations
48. Order of operations
49. Overhead
50. Pascal casing
51. Portable — self contained units that are transported
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 — meaningful names

aka order of operations arithmetic from left to right

School of Computing and Information Technologies

PROGCON - CHAPTER 2

29

CLASS NUMBER: 09 SECTION: TM181

NAME: DELA FUENTE, Marie Therese N. DATE: 11/8/19

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

a) Age 3pts

valid

b) age_*

invalid – the "*" is used to declare a pointer variable (holds the address of a variable), this should be placed before the variable and used with the data type 'int'. 5pts

c) +age 5pts

invalid – "+" is a reserved symbol for addition operations, it would be confusing for the program if that is placed with your variable

d) age_ 3pts

valid

e) _age 3pts

valid

f) Age

invalid – you cannot declare another variable with the same name (see letter A) in the function, this will confuse the program and have a difficulty of differentiating the two variables from each other

g) 1age 5pts

invalid – variables should not begin with a number – rather letters because then a string of digits would be a valid identifier as well as a valid number ex. int 17 = 497, int 42 = 6*9, etc.

h) Age 1 5pts

invalid – while spaces breaks up a programming language and will be difficult for the program to decipher – simply, it is not allowed