School of Computing and Information Technologies

(56)

PROGCON - CHAPTER 2

c/by: Deither Sunio

CLASS NUMBER: # 31 SECTION: BSTM-HRO 191

NAME: Torres, Icaiah DATE: 8/11/19

**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 collesion 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 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.

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.

right and
it to left associative
numeric
left to right associativity
overhead
Rules of
Precedence
in scope
garbage
local
global
Rules of
Precedence
external documentation
internal documentation
real numbers
end-of-job tasks
housekeeping
tasks
details loop tasks
module header
lower camel casing
kebob case
module return
statement
numeric
variable
main program
named constant
modules
initializing a
variable
functional decomposition
echoing input
assignment
operator
reusability

21. Describe operators that evaluate the expression to the right first.
22. Describes data that consists of numbers.
23. Describes operators that evaluate the expression to the left first.
24. Describes the extra resources a task requires.
25. Describes the rules of precedence.

26. Describes the state of data that is visible.
27. Describes the unknown value stored in an unassigned variable.
28. Describes variables that are declared within the module that uses them.
29. Describes variables that are known to an entire program.
30. Dictate the order in which operations in the same statement are carried out.

31. Documentation that is outside a coded program.
32. Documentation within a coded program.
33. Floating-point numbers.
34. Hold the steps you take at the end of the program to finish the application.
35. Include steps you must perform at the beginning of a program to get ready for the rest of the program.
36. Include the steps that are repeated for each set of input data.
37. Includes the module identifier and possibly other necessary identifying information.
38. Is another name for the camel casing naming convention.
39. Is sometimes used as the name for the style that uses dashes to separate parts of a name.
40. Marks the end of the module and identifies the point at which control returns to the program or module that called the module.

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.
42. Runs from start to stop and calls other modules.
43. Similar to a variable, except that its value cannot change after the first assignment.
44. Small program units that you can use together to make a program; programmers also refer to modules as subroutines, procedures, functions, or methods.
45. The act of assigning its first value, often at the same time the variable is created.

46. The act of containing a task's instructions in a module.
47. The act of reducing a large program into more manageable modules.
48. The act of repeating input back to a user either in a subsequent prompt or in output.
49. The equal sign; it is used to assign a value to the variable or constant on its left.
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.

kebob case 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 case 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

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

#31

School of Computing and Information Technologies

35

PROGCON - CHAPTER 2

CLASS NUMBER: 31

SECTION: BSTM - HRO 191

NAME: Torres, Isaiah

DATE:

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

a) Age
Yes

b) age_*
no because programs do not read asterisk and underscore as variable names

c) +age
no because the "+" symbol is not considered as part of a name in programming languages

d) age_
~~no~~ yes

e) _age
~~no~~ yes

f) Age
yes

g) 1age
Yes

h) Age 1
no because of the presence of the space