

#22



(120)

day: Allan

School of Computing and Information Technologies

PROGCON - CHAPTER 2

CLASS NUMBER: 22

SECTION: AC192

NAME: SARTE, CHARLIZE MAY

DATE: NOV. 8, 2019

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. 20
 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 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 value 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-associativity and Left-to-left associativity

21. Describe operators that evaluate the expression to the right first.

Numeric

22. Describes data that consists of numbers.

29

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.

Keobob 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.

Modularization

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

- | | | |
|---|--|--|
| 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 | 28. Internal documentation | 50. Pascal casing |
| 9. Data dictionary | 30. Kebob 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 | |

22



School of Computing and Information Technologies

PROGCON - CHAPTER 2

#22

Score 29

Checked by Niel

CLASS NUMBER: 22

SECTION: AC192

NAME: SARTI, CHARLIZE MAY

DATE: NOV. 8, 2019

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

a) Age

3 / Valid

b) age_*

5 / not valid

→ In declaring variables, operations should not be included. (*) is an operation for multiplication.

c) +age

5 / not valid

→ the first character should be a letter or some certain characters.

d) age_

0 / not valid

→ Variable names should be ~~not~~ avoided ending in underscores which may result in conflict with names of variables automatically by commands and procedures.

e) _age

3 / Valid

f) Age

3 / Valid

~~not~~ practical variable in java !!

g) 1age

5 / not valid

→ the first character of a variable must be a letter or one of the characters @, # or \$.

h) Age 1

5 / not valid

→ the variable should have / must be one word, no spacing.