



School of Computing and Information Technologies

PROGCON - CHAPTER 2

SECTION: AC 192

DATE: Nov 8

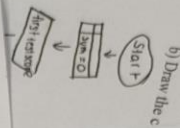
CLASS NUMBER: #18

NAME: Reyes, Niel Angelo

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 module 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 notations 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.

spreadsheets, payroll and inventory, even games
Application Software
Windows, Linux, or UNIX System software
Memory using hardware devices such as
RAM in flowcharts, Input Symbol
Output Symbol
Calculations with them, Processing Symbol
Processing Symbol
Data, such as
Symbol



- Right associativity
Left to right associativity
Numeric
Left to right associativity
Overhead
Order of operations
In scope
Garbage
Local
Global
Rules of precedence
External documentation
Internal documentation
Real numbers
End of job tasks
Housekeeping tasks
Detail loop tasks
Module header
Lower camel
Pascal casing
Kebab case
Module's return statement
Numeric variable
Main Program
Named constant
Modules
Initializing the variable
Encapsulation
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. Numeric
 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.

5. Get the first test score to sum
6. Add the second test score
7. Get the third
8. Add to
9.

- 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 comment 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 ✓ | 29. Internal documentation ✓ | 50. Pascal casing ✓ |
| 9. Data dictionary ✓ | 30. Kebab 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 ✓ | |



School of Computing and Information Technologies

PROGCON - CHAPTER 2

32 //

CLASS NUMBER: #18

SECTION: AC 192

NAME: Reyes, Niel Angelo

DATE: Nov 8

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

- a) Age
is valid
- b) age_*
is invalid - no other special characters allowed other than under score (_) so in this case the * is the special character that are prohibited in variable names
- c) +age
is invalid - no other special characters allowed other than under score (_) so in this case the plus sign (+) is the special character that are prohibited in variable names and that make the variable invalid
- d) age_
is valid
- e) _age
is valid
- f) Age
is valid
- g) 1age
is invalid - here the number 1 is first which is wrong because Variable name should start with letter (a-z or A-Z) or under score (_) and to correct move number 1 right making it Age1
- h) Age 1
is invalid - No spaces are allowed and because there is a space between Age and 1 that it making it Age_1 or make it Age1

corrected by: sophia.