UNCLASSIFIED//FOUO

Pseudocode, Logic and Design

1

Learning Objectives

3.1 Introduction
3.2 Defining and Calling a Module
3.3 Local Variables
3.4 Passing Arguments to Modules
3.5 Global Variables and Global Constants
3.6 Focus on Languages: Java, Python, and C++

2

2

UNCLASSIFIED

3.1 Introduction (1 of 2)

- A module is a group of statements that exists for the purpose of performing a specific task within a program.
- Most programs are large enough to be broken down into several subtasks.
- Divide and conquer: It's easier to tackle smaller tasks individually.

UNCLASSIFIED 3

3

UNCLASSIFIED

3.1 Introduction (2 of 2)

Benefits of using modules

- Simpler code
  - Small modules easier to read than one large one
- Code reuse
  - Can call modules many times
- Better testing
  - Test separate and isolate then fix errors
- Faster development
  - Reuse common tasks
- Easier facilitation of teamwork
  - Share the workload
- Easier Maintenance
  - Smaller, simpler code is easier to maintain

UNCLASSIFIED 4

4

3.2 Defining and Calling a Module (1 of 7)

- The code for a module is known as a module definition.

```
Module showMessage()
   Display "Hello world."
End Module
```

- To execute the module, you write a statement that calls it.

```
Call showMessage()
```

5

5

UNCLASSIFIED

3.2 Defining and Calling a Module (2 of 7)

- **A module's name should be descriptive enough so that anyone reading the code can guess what the module does.**
- **No spaces in a module name.**
- **No punctuation.**
- **Cannot begin with a number.**

UNCLASSIFIED 6

6

UNCLASSIFIED

3.2 Defining and Calling a Module (3 of 7)

- **Definition contains two parts**
  - **A header**
    - **The starting point of the module**
  - **A body**
    - **The statements within a module**

```
Module name( )
        Statement
        Statement
        Etc.
End Module
```

UNCLASSIFIED 7

7

3.2 Defining and Calling a Module (4 of 7)

- **A call must be made to the module in order for the statements in the body to execute.**

Figure 3-2 The `main` module

The program begins executing at the `main` module.

```
Module main()
   Display "I have a message for you."
   Call showMessage()
   Display "That's all, folks!"
End Module

Module showMessage()
   Display "Hello world"
End Module
```

When the end of the `main` module is reached, the program stops executing.

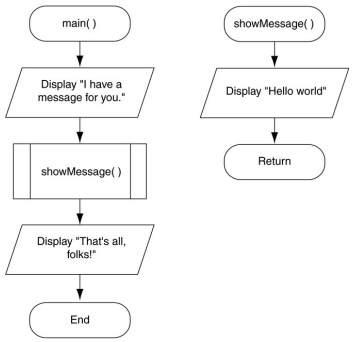
8

8

3.2 Defining and Calling a Module (3 of 7)

- When flowcharting a program with modules, each module is drawn separately.

Figure 3-6 Flowchart for Program 3-1



9

9

UNCLASSIFIED

3.2 Defining and Calling a Module (6 of 7)

- **A top-down design is used to break down an algorithm into modules by the following steps:**
  - **The overall task is broken down into a series of subtasks.**
  - **Each of the subtasks is repeatedly examined to determine if it can be further broken down.**
  - **Each subtask is coded.**

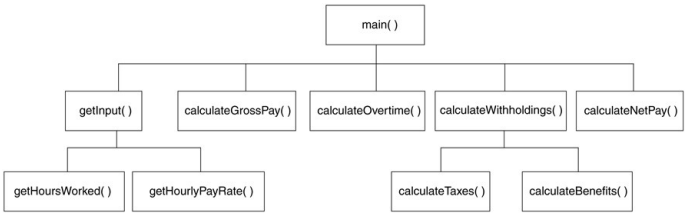
UNCLASSIFIED 10

10

3.2 Defining and Calling a Module (7 of 7)

- A hierarchy chart gives a visual representation of the relationship between modules.
- The details of the program are excluded.

Figure 3-7 A hierarchy chart



11

11

UNCLASSIFIED

3.3 Local Variables (1 of 1)

- A local variable is declared inside a module and cannot be accessed by statements that are outside the module.
- A variable's scope is the part of the program in which the variable can be accessed.

UNCLASSIFIED 12

12

3.3 Local Variables (2 of 3)

- **Duplicate Variable Names:** Variables within the same scope must have different names.

```
Module getTwoAges()
   Declare Integer age
   Display "Enter your age."
   Input age

   Declare Integer age
   Display "Enter your pet's age."
   Input age
End Module
```

This will cause an error! A variable named `age` has already been declared.

13

13

UNCLASSIFIED

3.3 Local Variables (3 of 3)

- **Duplicate Variable Names: Variables in different scopes can have the same name.**

```
Module main()
   Call showSquare()
   Call showHalf()
End Module

Module showSquare()
   Declare Real number
   Declare Real square

   Display "Enter a number."
   Input number
   Set square = number^2
   Display "The square of that number is ", square
End Module

Module showHalf()
   Declare Real number
   Declare Real half

   Display "Enter a number."
   Input number
   Set half = number / 2
   Display "Half of that number is ", half
End Module
```

Scope of the `number` variable in the `showSquare` module.

Scope of the `number` variable in the `showHalf` module.

UNCLASSIFIED 14

14

UNCLASSIFIED

3.4 Passing Arguments to Modules (1 of 3)

- Sometimes, one or more pieces of data need to be sent to a module.
- An argument is any piece of data that is passed into a module when the module is called.
- A parameter is a variable that receives an argument that is passed into a module.
- The argument and the receiving parameter variable must be of the same data type.
- Multiple arguments can be passed sequentially into a parameter list.

UNCLASSIFIED 15

15

3.4 Passing Arguments to Modules (2 of 3)

Figure 3-15 Two arguments passed into two parameters

```
Module main()
   Display "The sum of 12 and 45 is"
   Call showSum(12, 45)
End Module


Module showSum(Integer num1, Integer num2)
   Declare Integer result
   Set result = num1 + num2
   Display result
End Module
```

16

16

UNCLASSIFIED

3.4 Passing Arguments to Modules (3 of 3)

Pass by Value vs. Pass by Reference

- **Pass by Value means that only a copy of the argument's value is passed into the module.**
  - **One-directional communication: Calling module can only communicate with the called module.**
- **Pass by Reference means that the argument is passed into a reference variable.**
  - **Two-way communication: Calling module can communicate with called module; and called module can modify the value of the argument.**

UNCLASSIFIED 17

17

UNCLASSIFIED

3.5 Global Variables & Global Constants (1 of 2)

- **A global variable is accessible to all modules.**
- **Should be avoided because:**
  - **They make debugging difficult**
  - **Making the module dependent on global variables makes it hard to reuse module in other programs**
  - **They make a program hard to understand**

UNCLASSIFIED 18

18

UNCLASSIFIED

3.5 Global Variables & Global Constants (2 of 2)

- **A global constant is a named constant that is available to every module in the program.**
- **Since a program cannot modify the value of a constant, these are safer than global variables.**

UNCLASSIFIED 19

19