Programming Logic & Design

Chapter 6 – Repetition Structures (Part 2)

Queen's College CSD

1133 – CPCM -2023S

Chapter 6 – Functions

Topics:

- Introduction to Functions: Generating Random Numbers
- Writing Your Own Functions.
- More Library Functions

Chapter 6 – Functions

A function is a special type of module. It is like a regular module in the following ways:

A function is a group of statements that perform a specific task.

When you want to execute a function, you call it.

When a function finishes, however, it returns a value back to the part of the program that called it. The value that is returned from a function can be used like any other value: it can be assigned to a variable, displayed on the screen, used in a mathematical expression (if it is a number), and so on.

Most programming languages come with a library of functions that have already been written. These functions, known as library functions, are built into the programming language, and you can call them any time you need them. Library functions make a programmer's job easier because they perform many of the tasks that programmers commonly need to perform

Random Numbers :

Most programming languages provide a library function that generates random numbers. This chapter uses the random function for this purpose in the pseudocode. Random numbers are useful for lots of different programming tasks. The following are just a few examples:

Chapter 6 – Functions

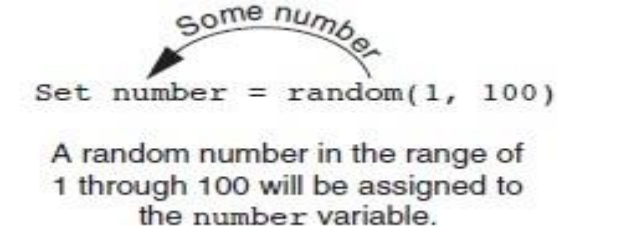
Most programming languages provide a library function that generates random numbers. This chapter uses the random function for this purpose in the pseudocode. Random numbers are useful for lots of different programming tasks. The following are just a few examples:

Random numbers are commonly used in games

Random numbers are useful in simulation programs

Random numbers are useful in statistical programs that must randomly select data for analysis

Random numbers are commonly used in computer security to encrypt sensitive data.



Chapter 6 – Functions

Problem:

Dr. Kimura teaches an introductory statistics class, and has asked you to write a program that he can use in class to simulate the rolling of dice. The program should randomly generate two numbers in the range of 1 through 6 and display them. In your interview with Dr. Kimura, you learn that he would like to use the program to simulate several rolls of the dice, one after the other. You decide to write a loop that simulates one roll of the dice, and then asks the user whether another roll should be performed. As long as the user answers “y” for yes, the loop will repeat.

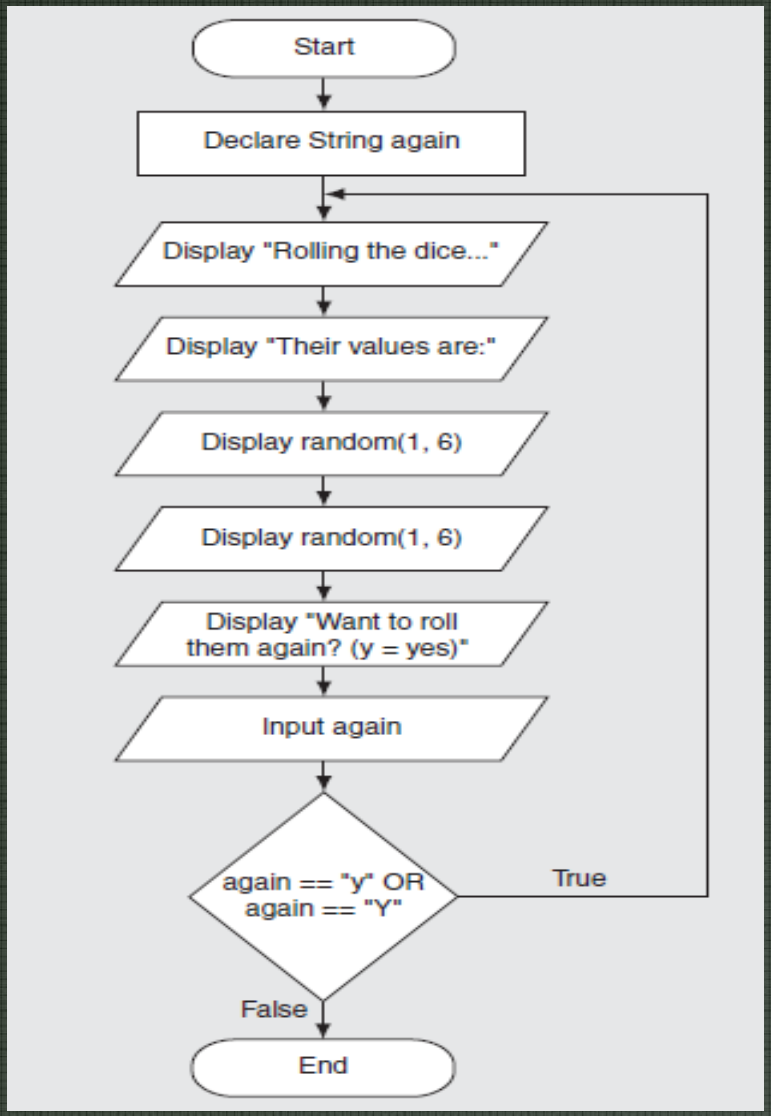
Design the pseudocode and flowchart

```
// Declare a variable to control the
// loop iterations.
Declare String again

Do
   // Roll the dice.
   Display "Rolling the dice..."
   Display "Their values are:"
   Display random(1, 6)
   Display random(1, 6)

   // Do this again?
   Display "Want to roll them again? (y = yes)"
   Input again
While again == "y" OR again == "Y"
```

Chapter 6 – Functions



Chapter 6 – Functions

Problem:

Kimura was so happy with the dice rolling simulator that you wrote for him, he has asked you to write one more program. He would like a program that he can use to simulate ten coin flips, one after the other. Each time the program simulates a coin flip, it should randomly display either “Heads” or “Tails.” You decide that you can simulate the flipping of a coin by randomly generating a number in the range of 1 through 2. You will design a decision structure that displays “Heads” if the random number is 1, or “Tails” otherwise.

Design pseudocode and flowchart.

Explain chapter 6 class examples

Chapter 6 – Functions

Writing Your Own Functions:

Functions are defined in a manner similar to modules. The following are the important characteristics of a function definition.

- The first line of a function definition, the function header, specifies the data type of the value that is returned from the function, the name of the function, and any parameter variables used by the function to accept arguments.
- Following the function header is the function body, which is comprised of one or more statements that are executed when the function is called.
- One of the statements in the function body must be a Return statement. A Return statement specifies the value that is returned from the function when the function ends.

Chapter 6 – Functions

Most programming languages allow you to create both modules and functions. Functions provide many of the same benefits as modules: they simplify code, reduce duplication, enhance your ability to test code, increase the speed of development, and ease the facilitation of teamwork. Because functions return a value, they can be useful in specific situations. For example, you can use a function to prompt the user for input, and then it can return the value entered by the user. Suppose you've been asked to design a program that calculates the sale price of an item in a retail business. To do that, the program would need to get the item's regular price from the user. Here is a function you could define for that purpose:

```
Function Real getRegularPrice()
   // Local variable to hold the price.
   Declare Real price

   // Get the regular price.
   Display "Enter the item's regular price."
   Input price

   // Return the regular price.
   Return price
End Function
```

Then, elsewhere in the program, you could call that function, as shown here:

```
   // Get the item's regular price.
   Set regularPrice = getRegularPrice()
```

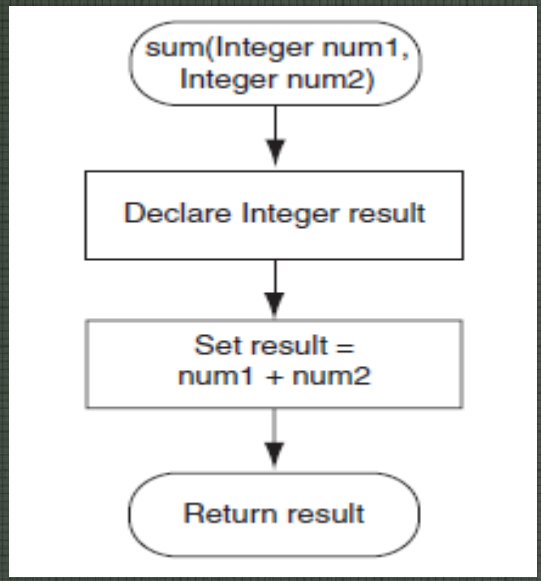
Chapter 6 – Functions

Following program shows a complete pseudocode program that uses the function..

```
1  Module main()
2     // Local variables
3     Declare Integer firstAge, secondAge, total
4
5     // Get the user's age and the user's
6     // best friend's age.
7     Display "Enter your age."
8     Input firstAge
9     Display "Enter your best friend's age."
10    Input secondAge
11
12    // Get the sum of both ages.
13    Set total = sum(firstAge, secondAge)
14
15    // Display the sum.
16    Display "Together you are ", total, " years old."
17 End Module
18
19 // The sum function accepts two Integer arguments and
20 // returns the sum of those arguments as an Integer.
21 Function Integer sum(Integer num1, Integer num2)
22    Declare Integer result
23    Set result = num1 + num2
24    Return result
25 End Function
```

Program Output (with Input Shown in Bold)

```
Enter your age.
22 [Enter]
Enter your best friend's age.
24 [Enter]
Together you are 46 years old.
```



Chapter 6 – Functions

Problem : (with the solution)

Hal owns a business named Make Your Own Music, which sells guitars, drums, banjos, synthesizers, and many other musical instruments. Hal's sales staff works strictly on commission. At the end of the month, each salesperson's commission is calculated according to Table 6-1.

Table 6-1 Sales Commission Rates

Sales This Month	Commission Rate
Less than $10,000.00	10%
$10,000.00–14,999.99	12%
$15,000.00–17,999.99	14%
$18,000.00–21,999.99	16%
$22,000 or more	18%

For example, a salesperson with $16,000 in monthly sales will earn a 14 percent commission ($2,240). Another salesperson with $20,000 in monthly sales will earn a 16 percent commission ($3,200). A person with $30,000 in sales will earn an 18 percent commission ($5,400).

Because the staff gets paid once per month, Hal allows each employee to take up to $2,000 per month in advance. When sales commissions are calculated, the amount of each employee's advanced pay is subtracted from the commission. If any salesperson's commission is less than the amount of his or her advance, the salesperson must reimburse Hal for the difference. To calculate a salesperson's monthly pay, Hal uses the following formula:

$$Pay = Sales \times Commission\ rate - Advanced\ pay$$

Chapter 6 – Functions

Program 6-8 **Commission rate program: `main` module**

```
 1 Module main()
 2    // Local variables
 3    Declare Real sales, commissionRate, advancedPay
 4
 5    // Get the amount of sales.
 6    Set sales = getSales()
 7
 8    // Get the amount of advanced pay.
 9    Set advancedPay = getAdvancedPay()
10
11    // Determine the commission rate.
12    Set commissionRate = determineCommissionRate(sales)
13
14    // Calculate the pay.
15    Set pay = sales * commissionRate - advancedPay
16
17    // Display the amount of pay.
18    Display "The pay is $", pay
19
20    // Determine whether the pay is negative.
21    If pay < 0 Then
22       Display "The salesperson must reimburse"
23       Display "the company."
24    End If
25 End Module
26
```

Chapter 6 – Functions

```
27 // The getSales function gets a salesperson's
28 // monthly sales from the user and returns
29 // that value as a Real.
30 Function Real getSales()
31    // Local variable to hold the monthly sales.
32    Declare Real monthlySales
33
34    // Get the amount of monthly sales.
35    Display "Enter the salesperson's monthly sales."
36    Input monthlySales
37
38    // Return the amount of monthly sales.
39    Return monthlySales
40  End Function
41
```

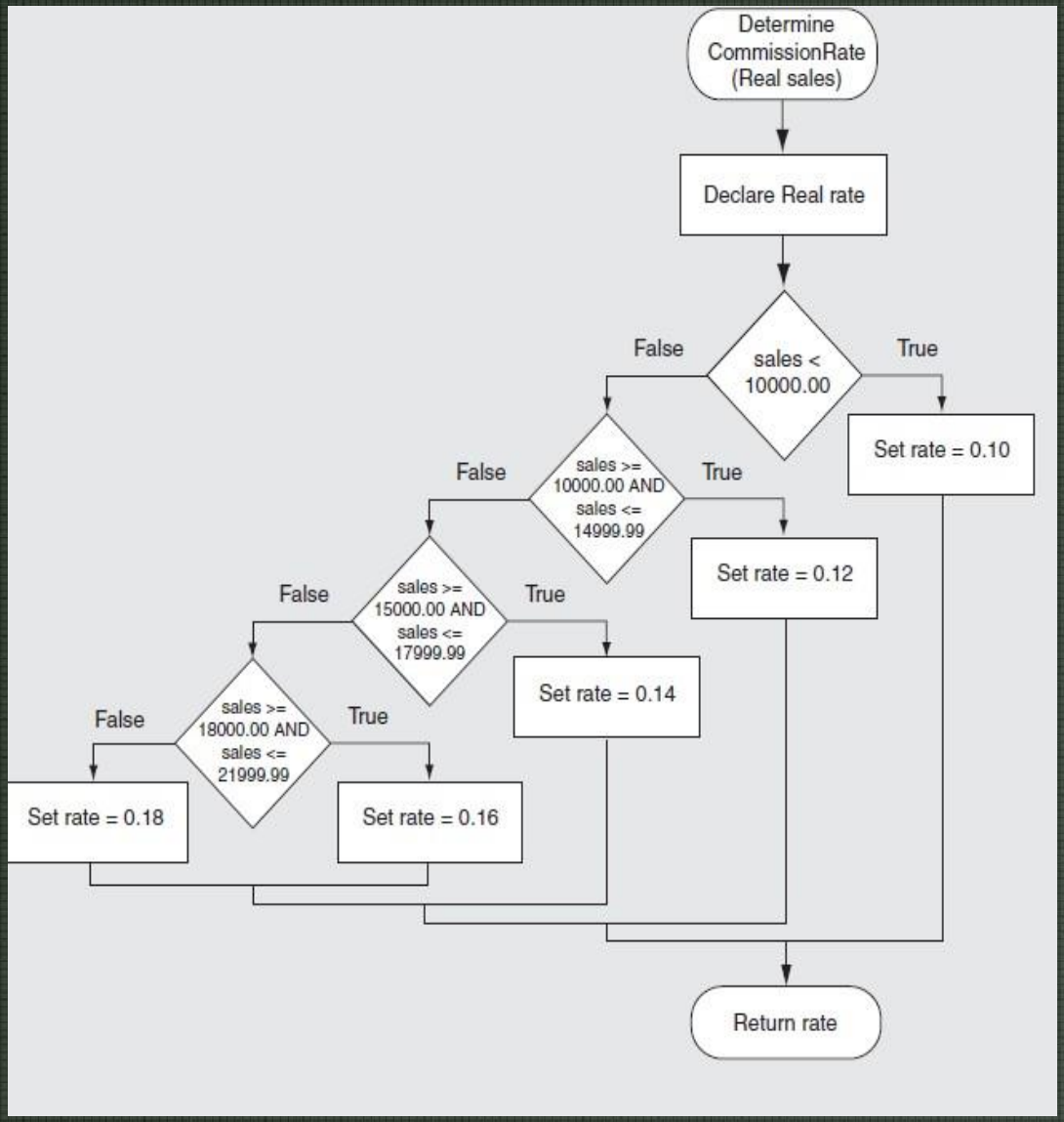
```
42 // The getAdvancedPay function gets the amount of
43 // advanced pay given to the salesperson and
44 // returns that amount as a Real.
45 Function Real getAdvancedPay()
46    // Local variable to hold the advanced pay.
47    Declare Real advanced
48
49    // Get the amount of advanced pay.
50    Display "Enter the amount of advanced pay, or"
51    Display "0 if no advanced pay was given."
52    Input advanced
53
54    // Return the advanced pay.
55    Return advanced
56 End Function
57
```

Chapter 6 – Functions

```
58 // The determineCommissionRate function accepts the
59 // amount of sales as an argument and returns the
60 // commission rate as a Real.
61 Function Real determineCommissionRate(Real sales)
62    // Local variable to hold commission rate.
63    Declare Real rate
64
65    // Determine the commission rate.
66    If sales < 10000.00 Then
67       Set rate = 0.10
68    Else If sales >= 10000.00 AND sales <= 14999.99 Then
69       Set rate = 0.12
70    Else If sales >= 15000.00 AND sales <= 17999.99 Then
71       Set rate = 0.14
72    Else If sales >= 18000.00 AND sales <= 21999.99 Then
73       Set rate = 0.16


74       Else
75          Set rate = 0.18
76       End If
77
78       // Return the commission rate.
79       Return rate
80 End Function
```

Chapter 6 – Functions



Program Output (with Input Shown in Bold)

```
Enter the salesperson's monthly sales.
14650.00 [Enter]
Enter the amount of advanced pay, or
0 if no advanced pay was given.
1000.00 [Enter]
The pay is $758.00
```

Program Output (with Input Shown in Bold)

```
Enter the salesperson's monthly sales.
9000.00 [Enter]
Enter the amount of advanced pay, or
0 if no advanced pay was given.
0 [Enter]
The pay is $900.00
```

Program Output (with Input Shown in Bold)

```
Enter the salesperson's monthly sales.
12000.00 [Enter]
Enter the amount of advanced pay, or
0 if no advanced pay was given.
2000.00 [Enter]
The pay is $-560
The salesperson must reimburse
the company.
```

Chapter 6 – Functions

More Library Functions:

Most programming languages provide several mathematical library functions. These functions typically accept one or more values as arguments, perform a mathematical operation using the arguments, and return the result.

The sqrt Function

The sqrt function accepts an argument and returns the square root of the argument.

Here is an example of how it is used:

Set result = sqrt(16)

The pow Function

Another common mathematical function is the pow function. The purpose of the pow function is to raise a number to a power

Set area = pow(4, 2)

Other Common mathematical functions :

Function Name	Description and Example Usage
abs	Returns the absolute value of the argument. *Example*: After the following statement executes, the variable y will contain the absolute value of the value in x. The variable x will remain unchanged. `y = abs(x)`

Chapter 6 – Functions

`cos`	Returns the cosine of the argument. The argument should be an angle expressed in radians. *Example*: After the following statement executes, the variable `y` will contain the cosine of the angle stored in the variable `x`. The variable `x` will remain unchanged. `y = cos(x)`
`round`	Accepts a real number as an argument and returns the value of the argument rounded to the nearest integer. For example, `round(3.5)` will return 4, and `round(3.2)` will return 3. *Example*: After the following statement executes, the variable `y` will contain the value of the variable `x` rounded to the nearest integer. The variable `x` will remain unchanged. `y = round(x)`

`sin`	Returns the sine of the argument. The argument should be an angle expressed in radians. *Example*: After the following statement executes, the variable `y` will contain the sine of the angle stored in the variable `x`. The variable `x` will remain unchanged. `y = sin(x)`
`tan`	Returns the tangent of the argument. The argument should be an angle expressed in radians. *Example*: After the following statement executes, the variable `y` will contain the tangent of the angle stored in the variable `x`. The variable `x` will remain unchanged. `y = tan(x)`

Chapter 6 – Functions

Checkpoint Questions;

How does a function differ from a module?

What is a library function?

Why are library functions like “black boxes”?

In pseudocode, what does the following statement do?

Set x = random(1, 100)

What is the purpose of the Return statement in a function?

Look at the following pseudocode function definition:

```
Function Integer doSomething(Integer number)
Return number * 2
End Function
```

a. What is the name of the function?

b. What type of data does the function return?

c.Given the function definition, what will the following statement display?

Display doSomething (10)

Chapter 6 – Functions

Programming Exercises:

1. Math Quiz

Design a program that gives simple math quizzes. The program should display two random numbers that are to be added, such as:

247

129

The program should allow the student to enter the answer. If the answer is correct, a message of congratulations should be displayed. If the answer is incorrect, a message showing the correct answer should be displayed.

2. Maximum of Two Values

Design a function named max that accepts two integer values as arguments and returns the value that is the greater of the two. For example, if 7 and 12 are passed as arguments to the function, the function should return 12. Use the function in a program that prompts the user to enter two integer values. The program should display the value that is the greater of the two.