Programming Logic & Design

Chapter 4 – Decision Structures and Boolean Logic

Queen's College

CSD 1133 – CPCM -2023S

Chapter 4 – Decision Structures

Topics:

- Introduction to Decision Structures
- Dual Alternative Decision Structures
- Comparing Strings
- Nested Decision Structures
- The Case Structure
- Logical Operators
- Boolean Variables

Chapter 4 – Decision Structures

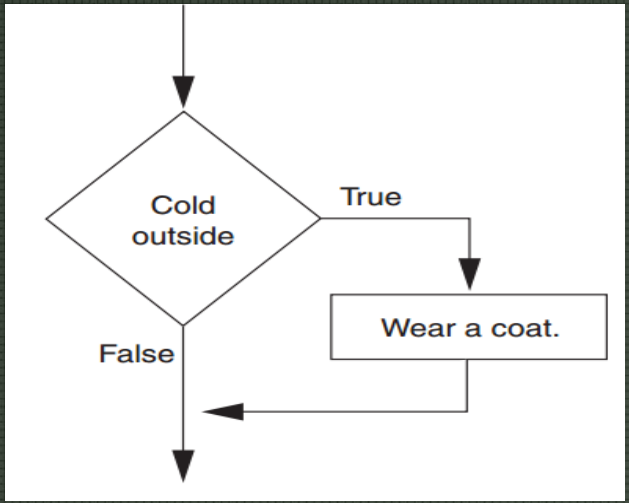
Introduction to Decision Structures

A control structure is a logical design that controls the order in which a set of statements executes. We have used only the simplest type of control structure: the sequence structure. Sequence structure is a set of statements that execute in the order that they appear

Although the sequence structure is heavily used in programming, it cannot handle every type of task. Some problems simply cannot be solved by performing a set of ordered steps, one after the other. For example, consider a pay calculating program that determines whether an employee has worked overtime. If the employee has worked more than 40 hours, he or she gets paid extra for all the hours over 40. Otherwise, the overtime calculation should be skipped. Programs like this require a different type of control structure: one that can execute a set of statements only under certain circumstances. This can be accomplished with a *decision structure*. In a decision structure's simplest form, a specific action is performed only if a certain condition exists. If the condition does not exist, the action is not performed.

Chapter 4 – Decision Structures

Decision Structures

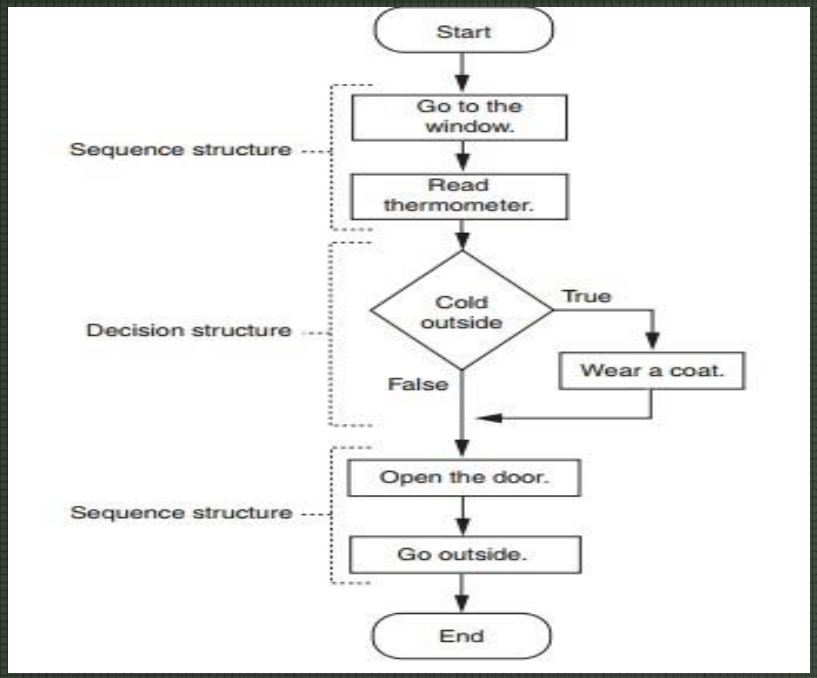


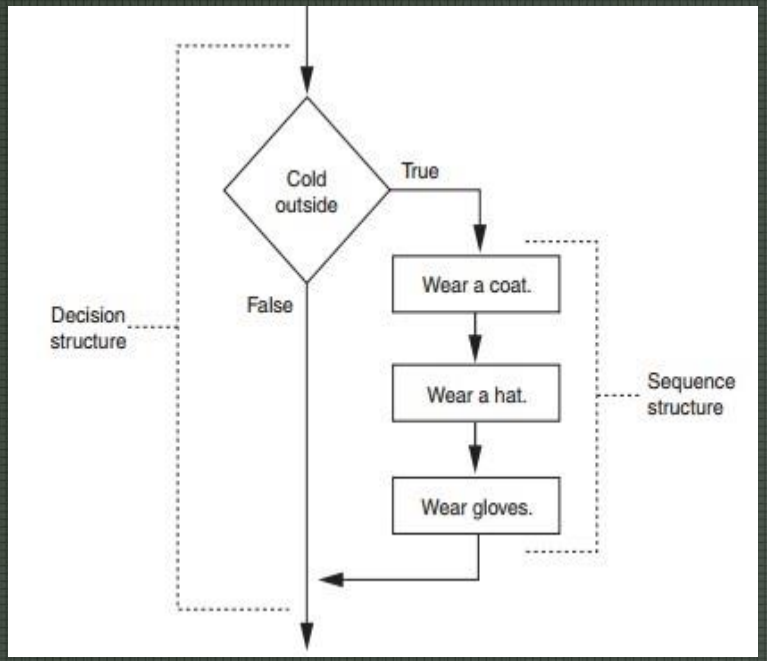
In the above flowchart, the diamond symbol indicates some condition that must be tested. In this case, we are determining whether the condition *Cold outside* is true or false. If this condition is true, the action *Wear a coat* is performed. If the condition is false, the action is skipped. The action is *conditionally executed* because it is performed only when a certain condition is true. We cannot use decision structures alone to create a complete program. We use a decision structure to handle any part of a program that needs to test a condition and conditionally execute an action depending on the outcome of the condition. For other parts of a program you need to use other structures.

Chapter 4 – Decision Structures

Combining Structures

For example, following figure shows a complete flowchart that combines a decision structure with two sequence structures.





Quite often, structures must be nested inside of other structures. For example, look at the partial flowchart in the above figure (right). It shows a decision structure with a sequence structure nested inside it.

Chapter 4 – Decision Structures

Writing a Decision Structure in Pseudocode

In pseudocode we use the If-Then statement to write a single alternative decision structure. Here is the general format of the If-Then statement:

```
If condition Then
    statement
    statement
    etc.
End If
```

These statements are conditionally executed.

Boolean Expressions and Relational Operators

All programming languages allow you to create expressions that can be valuated as either true or false. These are called Boolean expressions, Typically, the Boolean expression that is tested by an If-Then statement is formed with a relational operator. A relational operator determines whether a specific relationship exists between two values. For example, the greater than operator (>) determines whether one value is greater than another.

The following is an example of an expression that uses the greater than (>) operator to compare two variables, length and width:

length > width

Chapter 4 – Decision Structures

Relational Operators

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

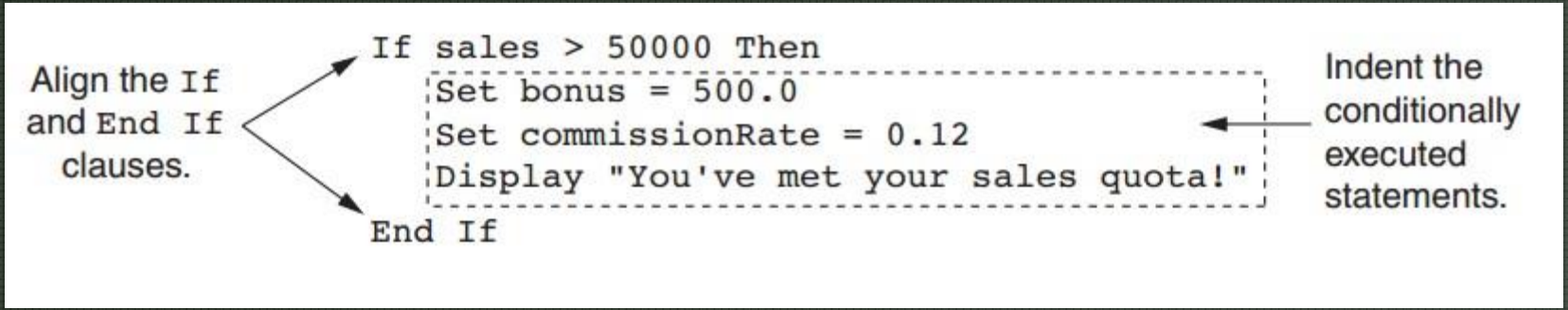
Two of the operators, >= and <=, test for more than one relationship. The >= operator determines whether the operand on its left is greater than or equal to the operand on its right. The == operator determines whether the operand on its left is equal to the operand on its right. If both operands have the same value, the expression is true. We use two = characters as the equal to operator to avoid confusion with the assignment operator, which is one = character. Several programming languages, most notably Java, Python, C, and C++, also follow this practice.

Chapter 4 – Decision Structures

Programming Style and the If-Then Statement

We use the following conventions when we write an If-Then statement:

- Make sure the If clause and the End If clause are aligned.
- Indent the conditionally executed statements that appear between the If clause and the End If clause.



Problem

Kathryn teaches a science class and her students are required to take three tests. She wants to write a program that her students can use to calculate their average test score. She also wants the program to congratulate the student enthusiastically if the average is greater than 95.

Chapter 4 – Decision Structures

Solution :

Here is the algorithm

1.Get the first test score.
2. Get the second test score.
3. Get the third test score.
4. Calculate the average.
5. Display the average.
6. If the average is greater than 95, congratulate the user.

Pseudocode:

```
// Declare variables
Declare Real test1, test2, test3, average
// Get test 1
Display "Enter the score for test #1."
Input test1
// Get test 2
Display "Enter the score for test #2."
Input test2
```

Chapter 4 – Decision Structures

```
// Get test 3
Display "Enter the score for test #3."
Input test3
// Calculate the average score.
Set average = (test1 + test2 + test3) / 3
// Display the average.
Display "The average is ", average
// If the average is greater than 95 congratulate the user.
If average > 95 Then
Display "Congratulations! Great average!"
End If
```

Program Output (with Input Shown in Bold)

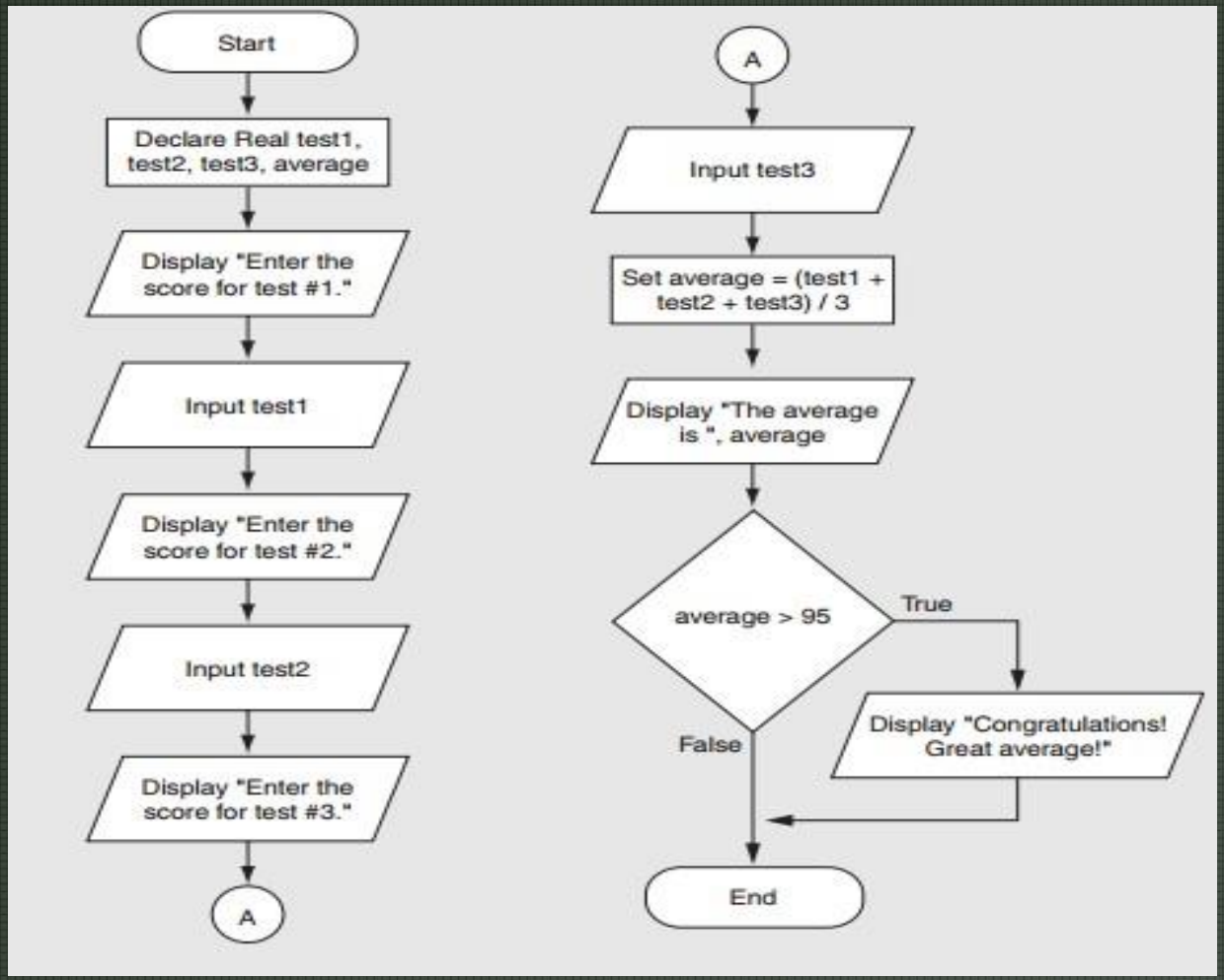
```
Enter the score for test #1.
82 [Enter]
Enter the score for test #2.
76 [Enter]
Enter the score for test #3.
91 [Enter]
The average is 83
```

Program Output (with Input Shown in Bold)

```
Enter the score for test #1.
93 [Enter]
Enter the score for test #2.
99 [Enter]
Enter the score for test #3.
96 [Enter]
The average is 96
Congratulations! Great average!
```

Chapter 4 – Decision Structures

Flowchart for Program



Chapter 4 – Decision Structures

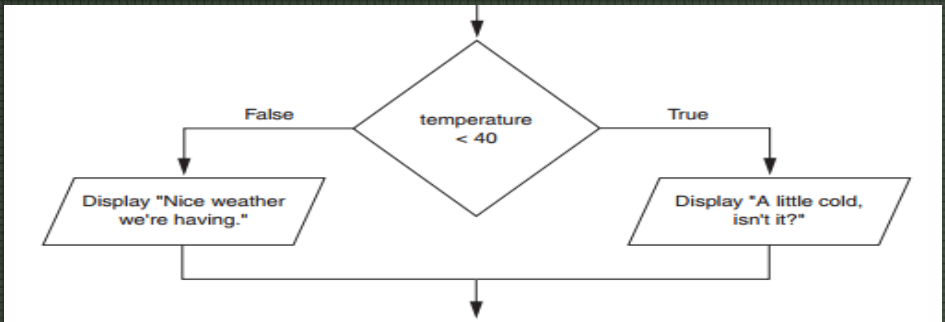
Checkpoint:

1. What is a control structure?
2. What is a decision structure?
3. What is a single alternative decision structure?
4. What is a Boolean expression?
5. What types of relationships between values can you test with relational operators?
6. Write a pseudocode If-Then statement that assigns 0 to x if y is equal to 20.
7. Write a pseudocode If-Then statement that assigns 0.2 to commission if sales is greater than or equal to 10,000.

Chapter 4 – Decision Structures

Dual Alternative Decision Structures

A dual alternative decision structure has two possible paths of execution—one path is taken if a condition is true, and the other path is taken if the condition is false



In pseudocode we write a dual alternative decision structure as an If-Then-Else statement. Here is the general format of the If-Then-Else statement:

```
If condition Then
    statement
    statement
    etc.
Else
    statement
    statement
    etc.
End If
```

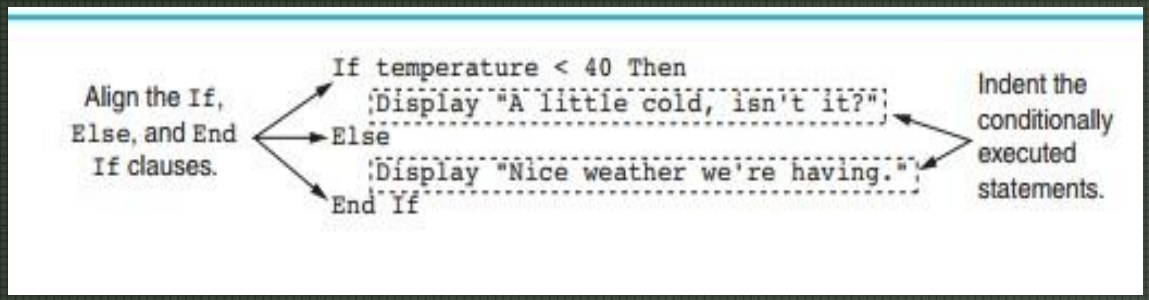
These statements are executed if the condition is true.

These statements are executed if the condition is false.

Chapter 4 – Decision Structures

Dual Alternative Decision Structures

The following pseudocode shows an example of an If-Then-Else statement. This pseudocode matches the flowchart that was in the previous slide.



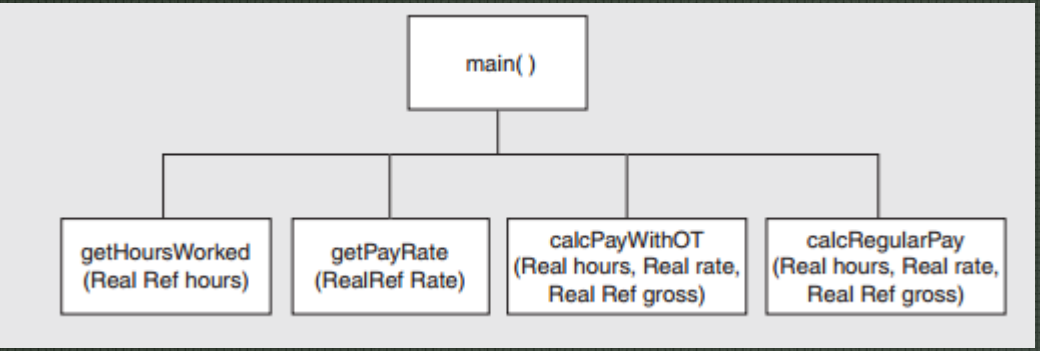
Problem :

Chris owns an auto repair business and has several employees. If an employee works over 40 hours in a week, Chris pays that employee 1.5 times his or her regular hourly pay rate for all hours over 40. Chris has asked you to design a simple payroll program that calculates an employee's gross pay, including any overtime wages.

Chapter 4 – Decision Structures

Solution :

You go through the top-down design process and create the hierarchy chart shown below. As shown in the hierarchy chart, the main module will call four other modules. The following are summaries of those modules:



Pseudocode :

```
1  // Global constants
2  Constant Integer BASE_HOURS = 40
3  Constant Real OT_MULTIPLIER = 1.5
4
5  Module main()
6     // Local variables
7     Declare Real hoursWorked, payRate, grossPay
8
9     // Get the number of hours worked.
10    Call getHoursWorked(hoursWorked)
11
```

Chapter 4 – Decision Structures

```
12     // Get the hourly pay rate.
13     Call getPayRate(payRate)
14
15     // Calculate the gross pay.
16     If hoursWorked > BASE_HOURS Then
17        Call calcPayWithOT(hoursWorked, payRate,
18                           grossPay)
19     Else
20        Call calcRegularPay(hoursWorked, payRate,
21                            grossPay)
22     End If
23
24     // Display the gross pay.
25     Display "The gross pay is $", grossPay
26  End Module
27
28  // The getHoursWorked module gets the number
29  // of hours worked and stores it in the
30  // hours parameter.
31  Module getHoursWorked(Real Ref hours)
32     Display "Enter the number of hours worked."
33     Input hours
34  End Module
35
36  // The getPayRate module gets the hourly
37  // pay rate and stores it in the rate
38  // parameter.
39  Module getPayRate(Real Ref rate)
40     Display "Enter the hourly pay rate."
41     Input rate
42  End Module
43
44  // The calcPayWithOT module calculates pay
45  // with overtime. The gross pay is stored
46  // in the gross parameter.
```

Chapter 4 – Decision Structures

```
47 Module calcPayWithOT(Real hours, Real rate,
48                      Real Ref gross)
49    // Local variables
50    Declare Real overtimeHours, overtimePay
51
52    // Calculate the number of overtime hours.
53    Set overtimeHours = hours - BASE_HOURS
54
55    // Calculate the overtime pay
56    Set overtimePay = overtimeHours * rate *
57                      OT_MULTIPLIER
58
59    // Calculate the gross pay.
60    Set gross = BASE_HOURS * rate + overtimePay
61 End Module
62
63 // The calcRegularPay module calculates
64 // pay with no overtime and stores it in
65 // the gross parameter.
66 Module calcRegularPay(Real hours, Real rate,
67                       Real Ref gross)
68    Set gross = hours * rate
69 End Module
```

Program Output (with Input Shown in Bold)

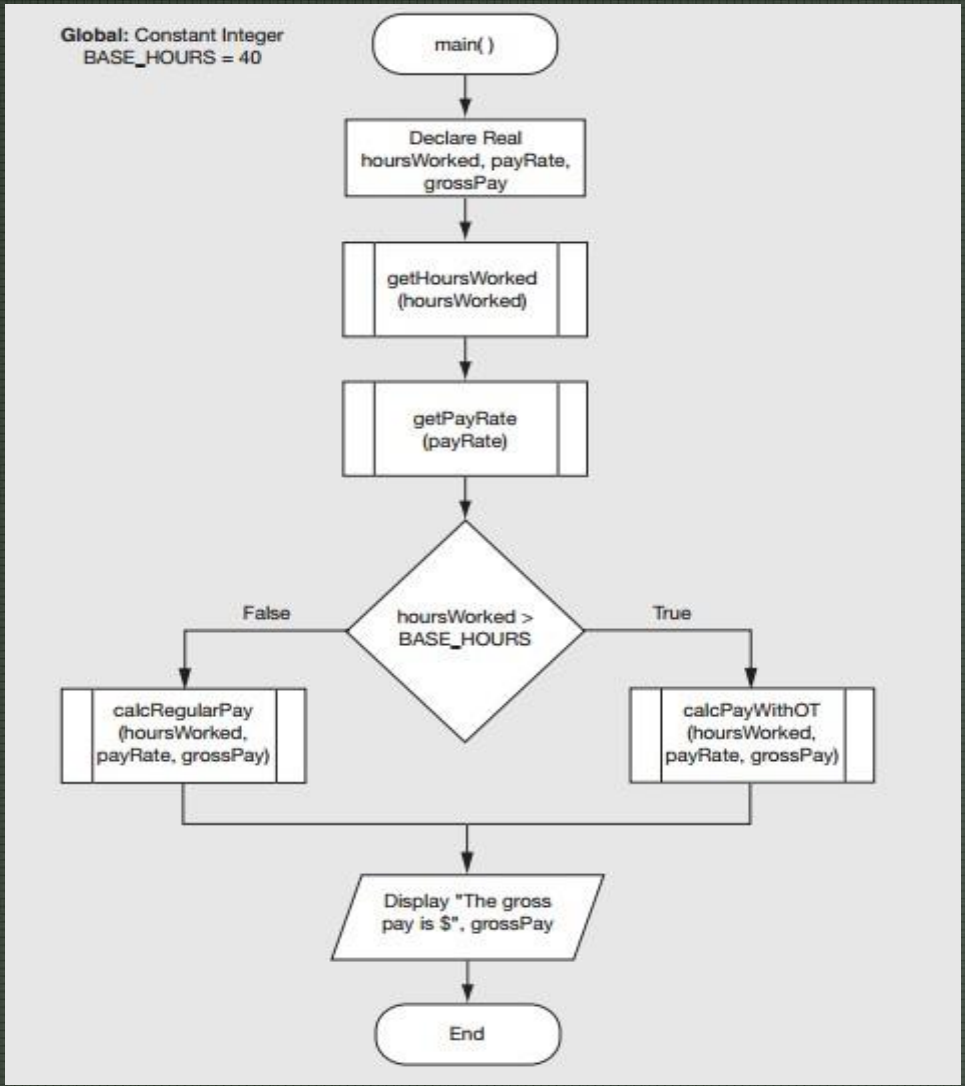
```
Enter the number of hours worked.
40 [Enter]
Enter the hourly pay rate.
20 [Enter]
The gross pay is $800
```

Program Output (with Input Shown in Bold)

```
Enter the number of hours worked.
50 [Enter]
Enter the hourly pay rate.
20 [Enter]
The gross pay is $1100
```

Chapter 4 – Decision Structures

Flow diagram:



Chapter 4 – Decision Structures

Comparing Strings

You saw in the preceding examples how numbers can be compared. Most programming languages also allow you to compare strings. For example, look at the following pseudocode:

```
Declare String name1 = "Mary"
Declare String name2 = "Mark"
If name1 == name2 Then
        Display "The names are the same"
Else
        Display "The names are NOT the same"
End If
```

```
1  // A variable to hold a password.
2  Declare String password
3
4  // Prompt the user to enter the password.
5  Display "Enter the password."
6  Input password
7
8  // Determine whether the correct password
9  // was entered.
10 If password == "prospero" Then
11    Display "Password accepted."
12 Else
13    Display "Sorry, that is not the correct password."
14 End If
```

Program Output (with Input Shown in Bold)

```
Enter the password.
ferdinand [Enter]
Sorry, that is not the correct password.
```

Program Output (with Input Shown in Bold)

```
Enter the password.
prospero [Enter]
Password accepted.
```

Chapter 4 – Decision Structures

Nested Decision Structures:

You can also nest decision structures inside of other decision structures. In fact, this is a common requirement in programs that need to test more than one condition.
For example, consider a program that determines whether a bank customer qualifies for a loan. To qualify, two conditions must exist:

(1) the customer must earn at least $30,000 per year,

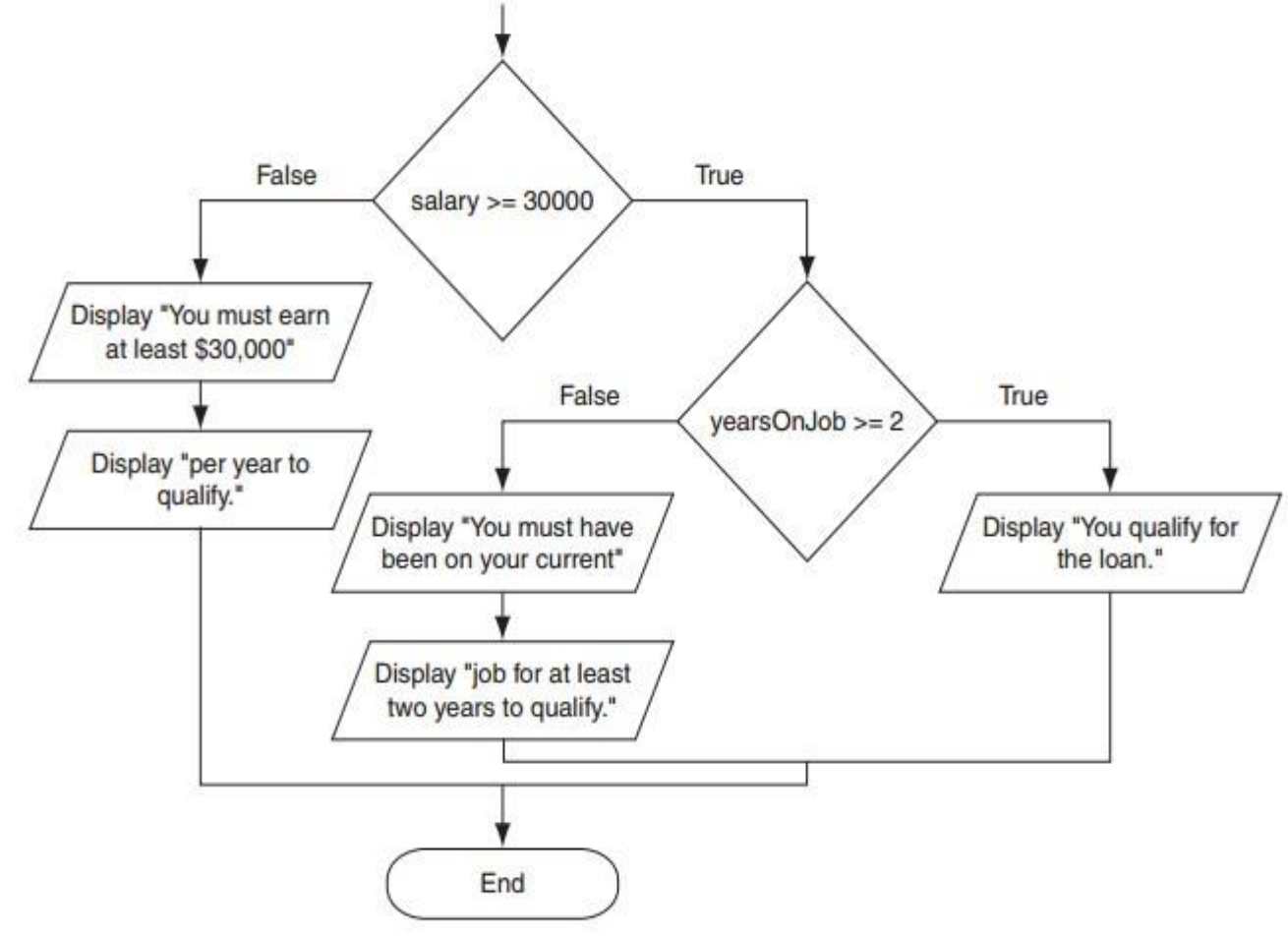
(2) the customer must have been employed at his or her current job for at least two years.

Figure in the next slide shows a flowchart for an algorithm that could be used in such a program. Assume that the salary variable contains the customer's annual salary, and the yearsOnJob variable contains the number of years that the customer has worked on his or her current job.

Chapter 4 – Decision Structures

FlowChart:

:



Chapter 4 – Decision Structures

Pseudocode:

```
1  // Declare variables
2  Declare Real salary, yearsOnJob
3
4  // Get the annual salary.
5  Display "Enter your annual salary."
6  Input salary
7
8  // Get the number of years on the current job.
9  Display "Enter the number of years on your"
10 Display "current job."
11 Input yearsOnJob
12
13 // Determine whether the user qualifies.
14 If salary >= 30000 Then
15    If yearsOnJob >= 2 Then
16       Display "You qualify for the loan."
17    Else
18       Display "You must have been on your current"
19       Display "job for at least two years to qualify."
20    End If
21 Else
22    Display "You must earn at least $30,000"
23    Display "per year to qualify."
24 End If
```

Program Output (with Input Shown in Bold)

```
Enter your annual salary.
35000 [Enter]
Enter the number of years on your
current job.
1 [Enter]
You must have been on your current
job for at least two years to qualify.
```

Program Output (with Input Shown in Bold)

```
Enter your annual salary.
25000 [Enter]
Enter the number of years on your
current job.
5 [Enter]
You must earn at least $30,000
per year to qualify.
```

Program Output (with Input Shown in Bold)

```
Enter your annual salary.
35000 [Enter]
Enter the number of years on your
current job.
5 [Enter]
You qualify for the loan.
```

Chapter 4 – Decision Structures

Programming Task :

Dr. Suarez teaches a literature class and uses the following 10 point grading scale for all of his exams:

Test Score Grade

90 and above A

80–89 B

70–79 C

60–69 D

Below 60 F

He has asked you to write a program that will allow a student to enter a test score and then display the grade for that score.

(Design a pseudocode and flowchart)

* For discussion during the class

Chapter 4 – Decision Structures

The If-Then-Else If Statement

Most languages provide a special version of the decision structure known as the If-Then-Else If statement, which makes this type of logic simpler to write. In pseudocode we will write the If-Then-Else If statement using the following general format:

```
If condition_1 Then
   statement
   statement                  } If condition_1 is true these statements are executed,
   etc.                       } and the rest of the structure is ignored.
Else If condition_2 Then
   statement
   statement                  } If condition_2 is true these statements are executed,
   etc.                       } and the rest of the structure is ignored.
Insert as many Else If clauses as necessary
Else
   statement
   statement                  } These statements are executed if none of the
   etc.                       } conditions above are true.
End If
```

When the statement executes, condition_1 is tested. If condition_1 is true, the statements that immediately follow are executed, up to the Else If clause. The rest of the structure is ignored. If condition_1 is false, however, the program jumps to the very next Else If clause and tests condition_2. If it is true, the statements that immediately follow are executed, up to the next Else If clause. The rest of the structure is then ignored.

Chapter 4 – Decision Structures

Pseudocode:

```
1  // Variable to hold the test score
2  Declare Real score
3
4  // Get the test score.
5  Display "Enter your test score."
6  Input score
7
8  // Determine the grade.
9  If score < 60 Then
10     Display "Your grade is F."
11 Else If score < 70 Then
12     Display "Your grade is D."
13 Else If score < 80 Then
14     Display "Your grade is C."
15 Else If score < 90 Then
16     Display "Your grade is B."
17 Else
18     Display "Your grade is A."
19 End If
```

Program Output (with Input Shown in Bold)

```
Enter your test score.
78 [Enter]
Your grade is C.
```

Program Output (with Input Shown in Bold)

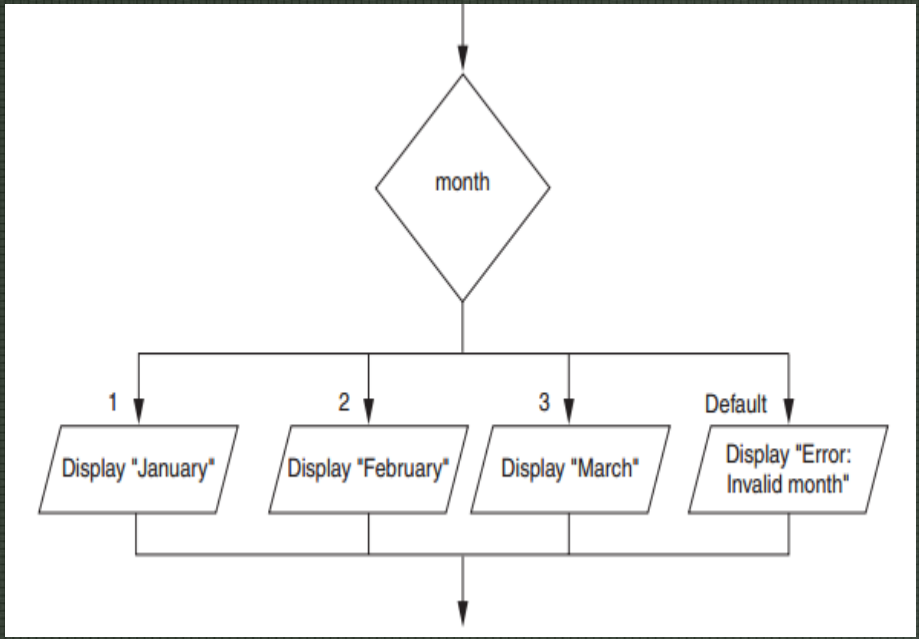
```
Enter your test score.
84 [Enter]
Your grade is B.
```

You never have to use the If-Then-Else If statement because its logic can be coded with nested If-Then-Else statements. However, a long series of nested If-Then-Else statements has some disadvantages when you are debugging code. For example the code can grow complex and become difficult to understand since the indenting is important in nested statements, a long series of nested If Then-Else statements can become too long to be displayed on the computer screen without horizontal scrolling.

Chapter 4 – Decision Structures

The Case Structure

The case structure is a multiple alternative decision structure. It allows you to test the value of a variable or an expression and then use that value to determine which statement or set of statements to execute




```
Select testExpression      ← This is a variable or an expression.
   Case value_1:
      statement
      statement            } These statements are executed if the
      etc.                   testExpression is equal to value_1.
   Case value_2:
      statement
      statement            } These statements are executed if the
      etc.                   testExpression is equal to value_2.
   Insert as many Case sections as necessary
   Case value_N:
      statement
      statement            } These statements are executed if the
      etc.                   testExpression is equal to value_N.
   Default:
      statement
      statement            } These statements are executed if the testExpression
      etc.                   is not equal to any of the values listed after the Case
                             statements.
End Select                 ← This is the end of the structure.
```

In many languages the case structure is called a switch statement.

Chapter 4 – Decision Structures

Problem:

Lenny, who owns Lenny's Stereo and Television, has asked you to write a program that will let a customer pick one of three TV models and then displays the price and size of the selected model. Here is the algorithm:

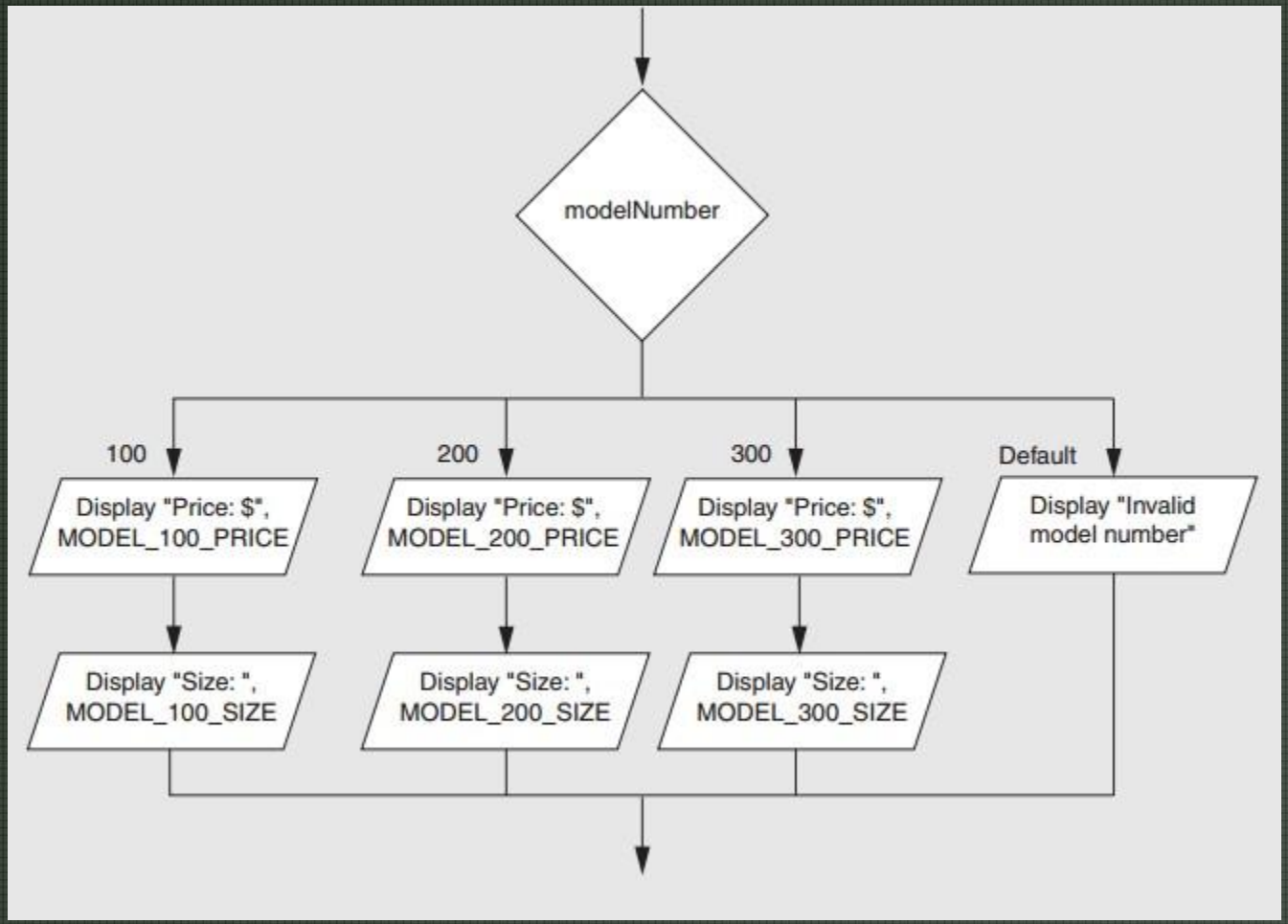
1. Get the TV model number.
2. If the model is 100, then display the information for that model. Otherwise, if the model is 200, then display the information for that model. Otherwise, if the model is 300, then display the information for that model.

Solution:

At first, you consider designing a nested decision structure to determine the model number and display the correct information. But you realize that a case structure will work just as well because a single value, the model number, will be used to determine the action that the program will perform. The model number can be stored in a variable, and that variable can be tested by the case structure.

Chapter 4 – Decision Structures

Flow Chart :



Chapter 4 – Decision Structures

Pseudocode :

```
 1 // Constants for the TV prices
 2 Constant Real MODEL_100_PRICE = 199.99
 3 Constant Real MODEL_200_PRICE = 269.99
 4 Constant Real MODEL_300_PRICE = 349.99
 5
 6 // Constants for the TV sizes
 7 Constant Integer MODEL_100_SIZE = 24
 8 Constant Integer MODEL_200_SIZE = 27
 9 Constant Integer MODEL_300_SIZE = 32
10
11 // Variable for the model number
12 Declare Integer modelNumber
13
14 // Get the model number.
15 Display "Which TV are you interested in?"
16 Display "The 100, 200, or 300?"
17 Input modelNumber
18
19 // Display the price and size.
20 Select modelNumber
21    Case 100:
22       Display "Price: $", MODEL_100_PRICE
23       Display "Size: ", MODEL_100_SIZE
24    Case 200:
25       Display "Price: $", MODEL_200_PRICE
26       Display "Size: ", MODEL_200_SIZE
27    Case 300:
28       Display "Price $", MODEL_300_PRICE
29       Display "Size: ", MODEL_300_SIZE
30    Default:
31       Display "Invalid model number"
32 End Select
```

Program Output (with Input Shown in Bold)

```
Which TV are you interested in?
The 100, 200, or 300?
100 [Enter]
Price: $199.99
Size: 24
```

Program Output (with Input Shown in Bold)

```
Which TV are you interested in?
The 100, 200, or 300?
200 [Enter]
Price: $269.99
Size: 27
```

Chapter 4 – Decision Structures

Logical Operators:

The logical AND operator and the logical OR operator allow you to connect multiple Boolean expressions to create a compound expression

Following Table shows examples of several compound Boolean expressions that use logical operators.

Operator	Meaning
AND	The AND operator connects two Boolean expressions into one compound expression. Both subexpressions must be true for the compound expression to be true.
OR	The OR operator connects two Boolean expressions into one compound expression. One or both subexpressions must be true for the compound expression to be true. It is only necessary for one of the subexpressions to be true, and it does not matter which.
NOT	The NOT operator is a unary operator, meaning it works with only one operand. The operand must be a Boolean expression. The NOT operator reverses the truth of its operand. If it is applied to an expression that is true, the operator returns false. If it is applied to an expression that is false, the operator returns true.

The AND operator takes two Boolean expressions as operands and creates a compound Boolean expression that is true only when both subexpressions are true. The following is an example of an If-Then statement that uses the AND operator:

Chapter 4 – Decision Structures

```
If temperature < 20 AND minutes > 12 Then
        Display "The temperature is in the danger zone."
End If
```

The OR operator takes two Boolean expressions as operands and creates a compound Boolean expression that is true when either of the subexpressions is true. The following is an example of an If-Then statement that uses the OR operator:

```
If temperature < 20 OR temperature > 100 Then
        Display "The temperature is in the danger zone."
End If
```

Sometimes you will need to design an algorithm that determines whether a numeric value is within a specific range of values or outside a specific range of values. When determining whether a number is inside a range, it is best to use the AND operator. For example, the following If-Then statement checks the value in x to determine whether it is in the range of 20 through 40:

Chapter 4 – Decision Structures

```
If x >= 20 AND x <= 40 Then
        Display "The value is in the acceptable range."
End If
```

When determining whether a number is outside a range, it is best to use the OR operator. The following statement determines whether x is outside the range of 20 through 40:

```
If x < 20 OR x > 40 Then
        Display "The value is outside the acceptable range."
End If
```

Find the error in the code below

```
If x < 20 AND x > 40 Then
Display "The value is outside the acceptable range."
End If
```

Chapter 4 – Decision Structures

Checkpoint

1.Assume the variables a = 2, b = 4, and c = 6.Find in each of the following conditions whether its value is true or false

a == 4 OR b > 2

6 <= c AND a > 3

1 != b AND c != 3

a >= -1 OR a <= b

NOT (a > 2)

2.Write an If-Then statement that displays the message “The number is valid” if the variable speed is within the range 0 through 200.

3.Write an If-Then statement that displays the message “The number is not valid” if the variable speed is outside the range 0 through 200

4. What is a multiple alternative decision structure?

5. How do you write a multiple alternative decision structure in pseudocode?

6.When you write an If-Then-Else statement, under what circumstances do the statements that appear between the Else clause and the End If clause execute?

Chapter 4 – Decision Structures

Programming Exercises : (Pseudocode /Flowchart)

1. Areas of Rectangles

The area of a rectangle is the rectangle's length times its width. Design a program that asks for the length and width of two rectangles. The program should tell the user which rectangle has the greater area, or whether the areas are the same.

2. Software Sales

A software company sells a package that retails for $99. Quantity discounts are given according to the following table:

Quantity	Discount
10–19	20%
20–49	30%
50–99	40%
100 or more	50%

Design a program that asks the user to enter the number of packages purchased. The program should then display the amount of the discount (if any) and the total amount of the purchase after the discount

3.Roman Numeral

Design a program that prompts the user to enter a number within the range of 1 through 10. The program should display the Roman numeral version of that number. If the number is outside the range of 1 through 10, the program should display