CIS1400 – Programming Logic and Technique

Topic 4 → Selection Control Structures

Chapter Topics

4.1 Introduction to Decision Structures

4.2 Dual Alternative Decision Structures

4.3 Comparing Strings

4.4 Nested Decision Structures

4.5 The Case Structure

4.6 Logical Operators

4.7 Boolean Variables

Sample Program: Overtime PayCalculator

4.1 Introduction to Decision Structures

A decision structure allows a program to perform actions only under certain conditions

Different types of decisions include

- If, also called single alternative
- If then else, also called dual alternative
- Case structure for multiple alternative decisions

Often referred to as Selection Control Structures

Can be combined with Sequential Control Structures

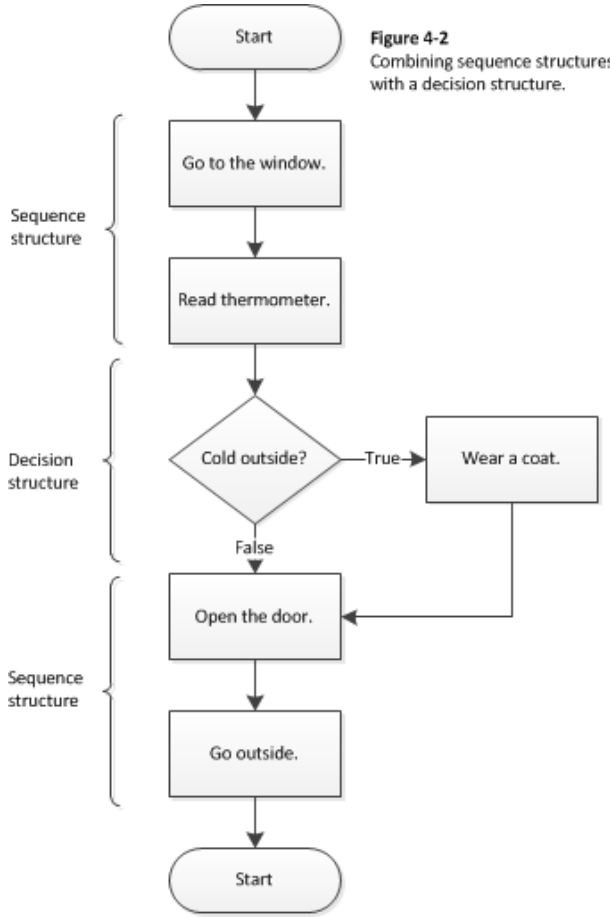


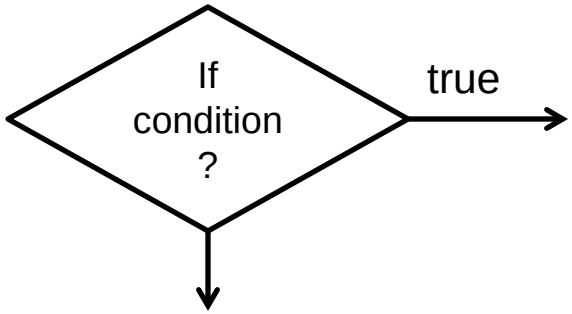
Figure 4-2
Combining sequence structures with a decision structure.

4.1 Introduction to Decision Structures

The if statement

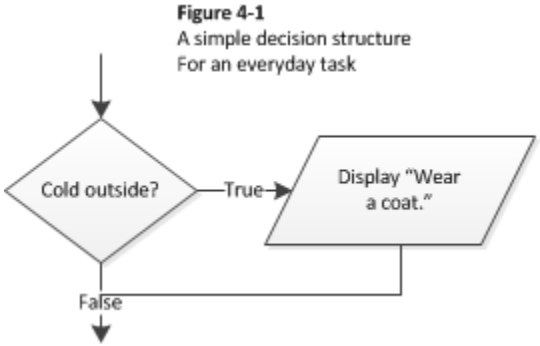
- Boolean expression represents condition
  - Boolean expression often involves logical (as opposed to mathematical) operators.
- An action only occurs if the condition is true

If condition Then
Statement
Statement
End If



- A diamond symbol used in flowcharts to represent decision

4.1 Introduction to Decision Structures




```
If coldOutside Then
  Display "Wear a coat."
End If
```

Note alignment on pseudocode and symbols on flowchart. If clause and End If clause surround statements to be executed when condition evaluates to Boolean 'true'.

4.1 Introduction to Decision Structures

Relational Operators

- Determines whether a specific relationship exists between two values
- Used within the condition, a Boolean expression

x > y x<y x >=y x <=y x==y x!=y

Table 4-1 Relational operators

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

Be careful of == versus = operator. Some languages use a single = to represent assignment as well as equality. Some languages may use < > in place of !=.

4.2 Dual Alternative Decision Structures

If then else statement

- Executes one group of statements if it's Boolean expression is true, or another group if its Boolean expression is false

If condition Then
    Statement
    Statement
Else
    Statement
    Statement
End If

The Dual Alternative Selection structure has two possible paths of execution: one if the condition is true, and another if the condition is false.

4.2 Dual Alternative Decision Structures

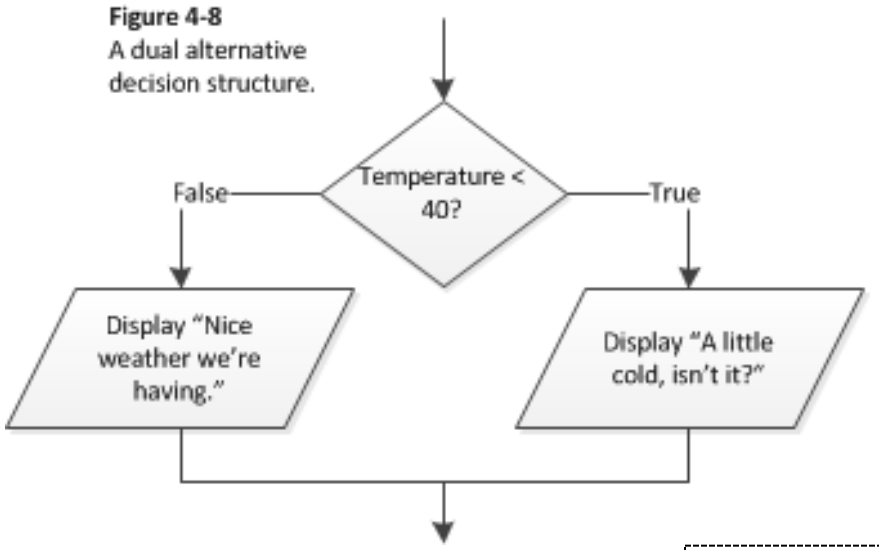


Figure 4-8
A dual alternative decision structure.

Note alignment on pseudocode and symbols on flowchart

```
If temperature < 40 Then
  Display "A little cold, isn't it?"
Else
  Display "Nice weather we're having."
End If
```

4.3 Comparing Strings

Most languages allow you to compare strings

Program 4-3

```
 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
```

Remember the ASCII representation of character from a previous week. Some languages do not allow relational operators—functions are used to perform the comparisons between strings. It is really the ASCII representation that is used when using relational operators with characters.

Program Output (with Input Shown in Bold)

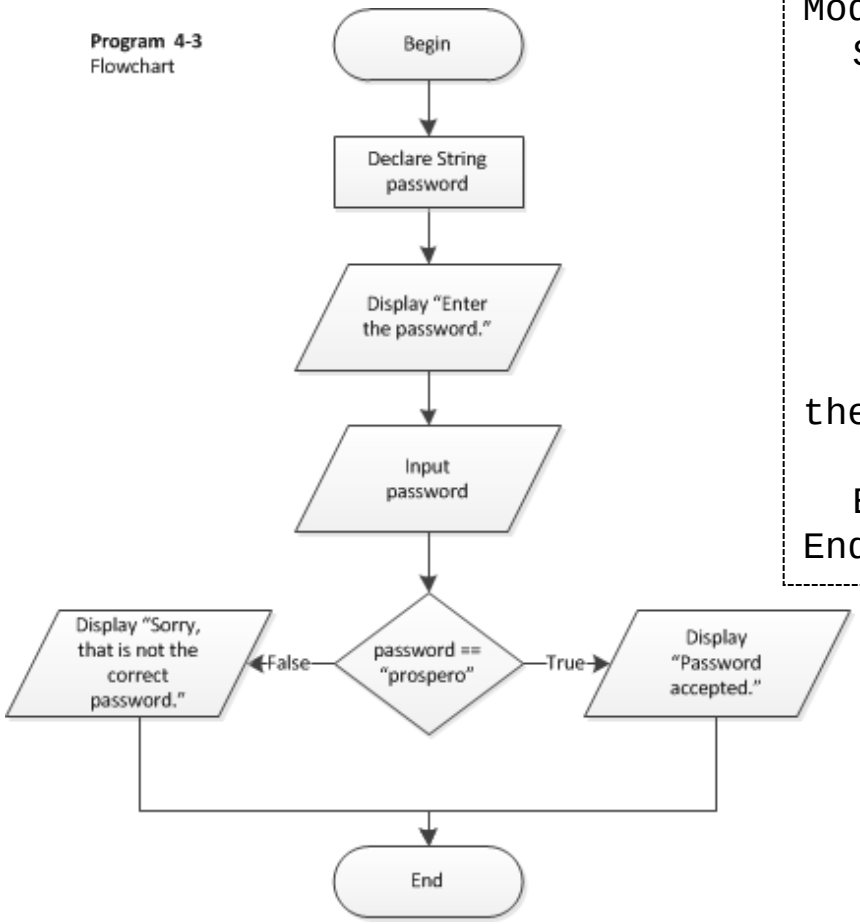
```
Enter the password.
ferdinand [Enter]
Sorry, that is not the correct password.
```

Program Output (with Input Shown in Bold)

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

4.3 Comparing Strings

Visual Basic Syntax (p39 VBLC)




```
Module Module1
  Sub Main()
    Dim password as String
    Console.Write("Enter the password: ")
    password = Console.ReadLine()
    If password = "prospero" Then
      Console.WriteLine("Password accepted.")
    Else
      Console.Writeline("Sorry, that is not _
the correct password.")
    End If
  End Sub
End Module
```

4.3 Comparing Strings

Other String Concerns

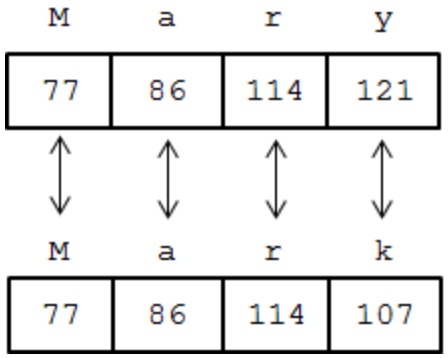
- String and strings can be compared

  name1 == name 2

- String and string literals can be compared

  Month != "October"

- String comparisons are generally case sensitive
- You can also determine whether one string is greater than or less than another string (allows for sorting strings)



"Mark" < "Mary"

ASCII representation used for comparison (see **Other String Comparisons** in textbook)

4.4 Nested Decision Structures

Decisions are nested in order to test more than one condition

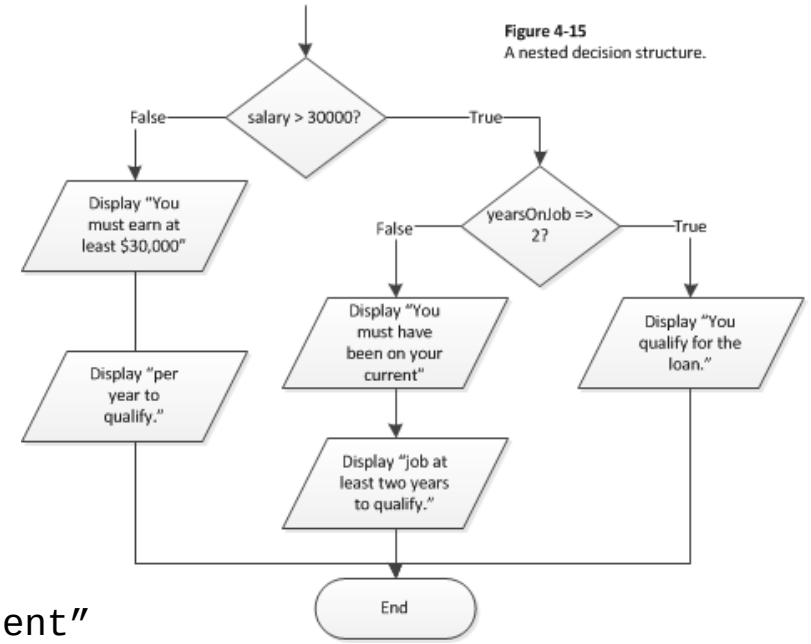


Figure 4-15
A nested decision structure.

Program 4-5

```
If salary >= 30000 Then
   If yearsOnJob >= 2 Then
      Display "You qualify for the loan."
   Else
      Display "You must have been on your current"
      Display "job for at least two years to qualify."
   End If
Else
   Display "You must earn at least $30,000"
   Display "per year to qualify."
End If
```

4.4 Nested Decision Structures

```
If score < 60 Then
  Display “Grade is F.”
Else
  If score < 70 Then
    Display “Grade is D.”
  Else
    If score < 80 Then
      Display “Grade is C.”
    Else
      If score < 90 Then
        Display “Grade is B.”
      Else
        Display “Grade is A.”
      End If
    End If
  End If
End If
```

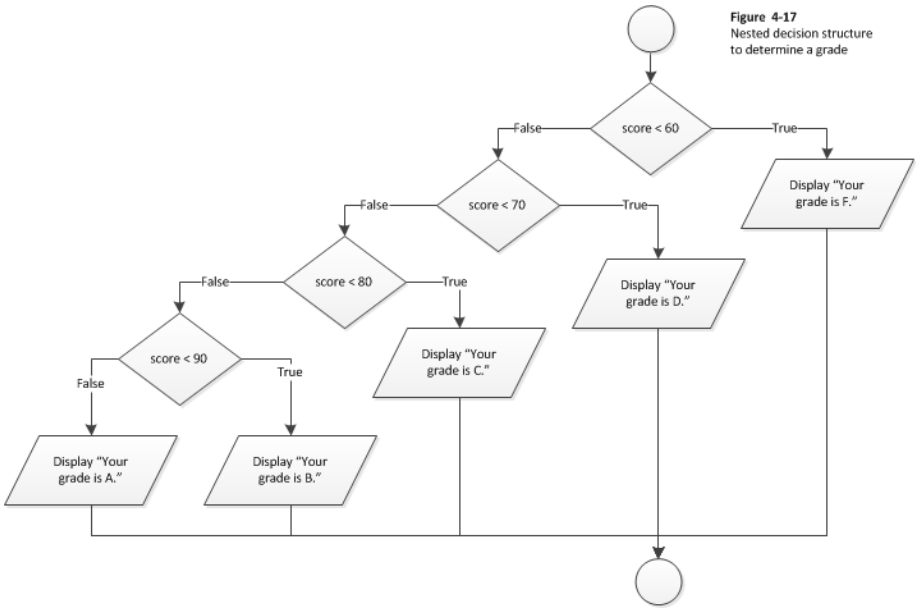


Figure 4-17
Nested decision structure to determine a grade

4.4 Nested Decision Structures

The if then else if statement can make nested logic simpler to write

Program 4-6

```
If score < 60 Then
  Display “Grade is F.”
Else If score < 70 Then
  Display “Grade is D.”
Else If score < 80 Then
  Display “Grade is C.”
Else If score < 90 Then
  Display “Grade is B.”
Else
  Display “Grade is A.”
End If
```

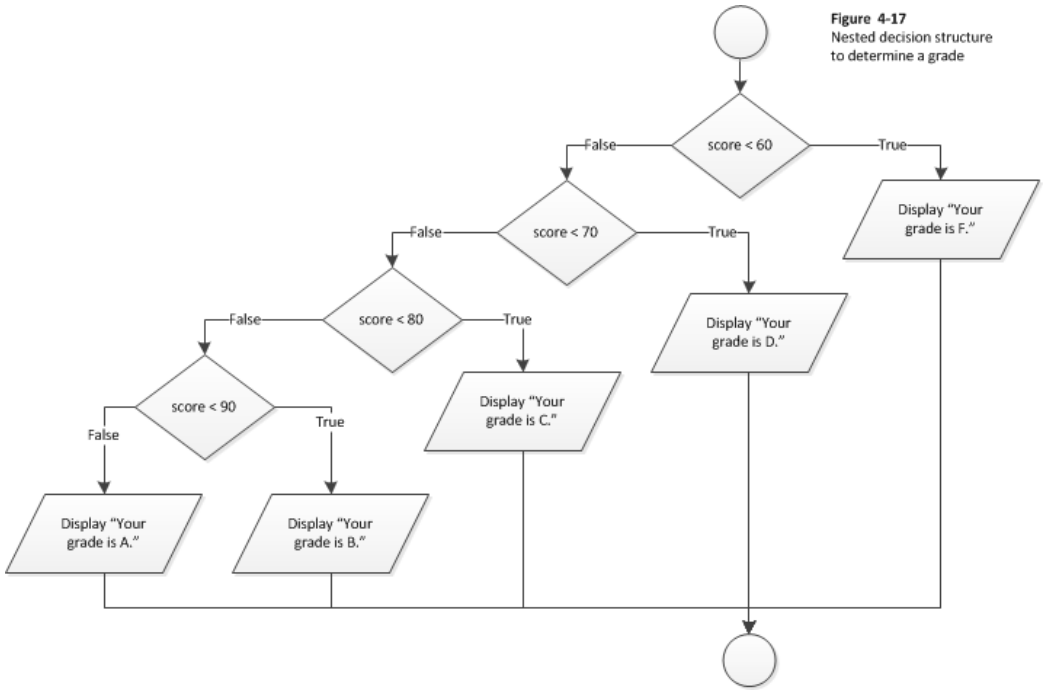


Figure 4-17
Nested decision structure to determine a grade

4.5 The Case Structure

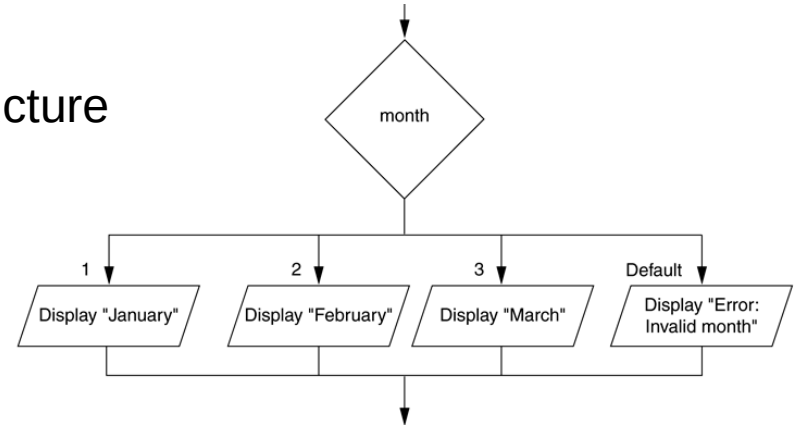
The case structure lets variable value or an expression determine path of program execution

- Can be used as an alternative to nested decisions
- Also referred to as Multiple Alternative Selection Control Structures or Switch Statement

Case statements are not required

- Same logic can be accomplished with nested decision structures.

Figure 4-18 A case structure



4.5 The Case Structure

testExpression is usually a variable, but in many languages it can also be anything that gives a value.

- If the testExpression matches one of the Case values,
  - Set of statements following that Case are executed.
- If testExpression does not match any Case values,
  - Program branches to Default statement and execute statements following the Default statement.

```
Select testExpression
    Case value_1:
        statement
        statement
    Case value_N:
        statement
        statement
    Default:
        statement
        statement
End Select
```

4.5 The Case Structure

```
Select month
  Case 1:
    Display “January”
  Case 2:
    Display “February”
  Case 3:
    Display “March”
  Default:
    Display “Error: Invalid month”
End Select
```

Visual Basic Syntax (p44 VBLC)

```
Select Case month
  Case 1
    Console.WriteLine(“January”)
  Case 2
    Console.WriteLine(“February”)
  Case 3
    Console.WriteLine(“March”)
  Case Else
    Console.WriteLine(Error: Invalid month”)
End Select
```

4.6 Logical Operators

Logical Operators are used between complete conditions to create complex Boolean expressions

- AND – Both conditions must be true
- OR – Either condition must be true
- NOT – Reverses the truth of an expression

AND

	True	False
True	True	False
False	False	False

OR

	True	False
True	True	True
False	True	False

NOT

True	False
False	True

4.6 Logical Operators

Note complete conditions on either side of AND and OR logical operators.

AND example

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

OR example

temperature < 20 OR > 100

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

NOT example

```
If NOT (temperature > 100) Then
  Display "This is below the maximum temperature."
End If
```

4.6 Logical Operators

Range Checking

- Often used for range checking
  - When checking for a number **inside** a range, use AND

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

  - When checking for a number **outside** a range, use OR

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

Only 1 sub-expression can feasibly be true!

4.7 Boolean Variables

A variable of the Boolean data type can hold one or two values: true or false

Often used as flags to represent the existence of a condition

```
Declare Real average
Declare Boolean highScore = False
If average > 95 Then
  Set highScore = True
End If
. . .
If highScore Then
  Display “That’s a high score!”
End If
```

Visual Basic Syntax (p48 VBLC)

```
Dim highScore As Boolean = False
If average > 95 Then
  highScore = true
End If
. . .
If highScore Then
  Console.WriteLine(“That’s a high _
score!”)
End If
```

Sample Program: Overtime PayCalculator

Modification "*In The Spotlight*" page 132

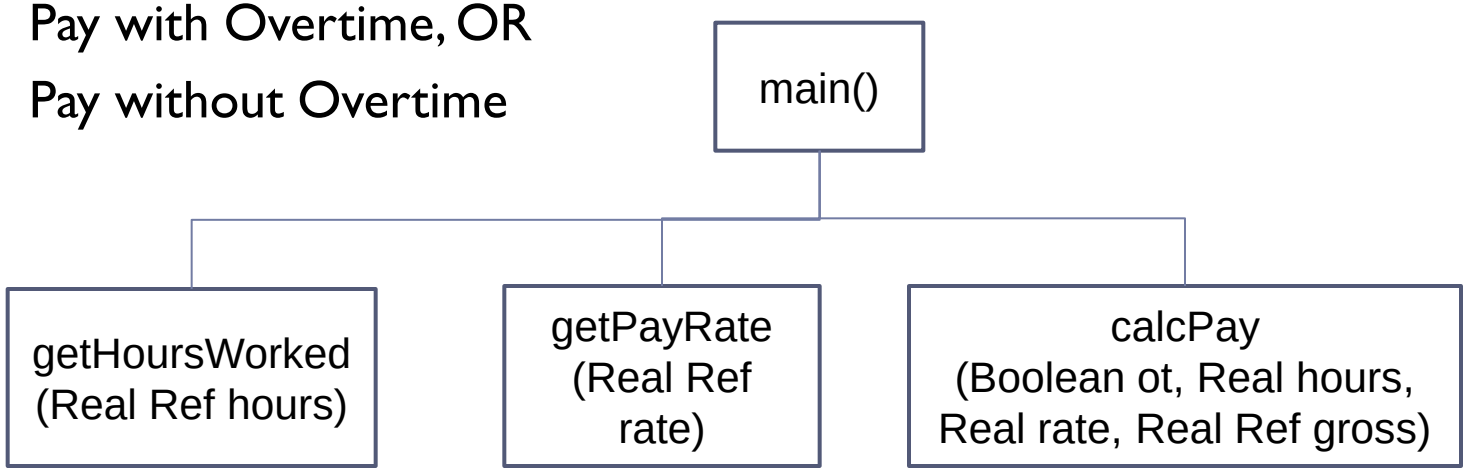
- Overtime Pay
  - Hours > 40
  - 1.5 times regular hourly rate
- ***What is required for each phase of the program?***

1. What must be read as input?
   - Get hours worked
   - Get pay rate
2. How will the input be processed?
   - Regular pay* = hours * pay rate
   - Overtime pay = (hours – 40) * pay rate * 1.5
   - Gross pay = regular pay + overtime pay
3. What will be done with the output?
   - Display gross pay

**hours = 40 if overtime*

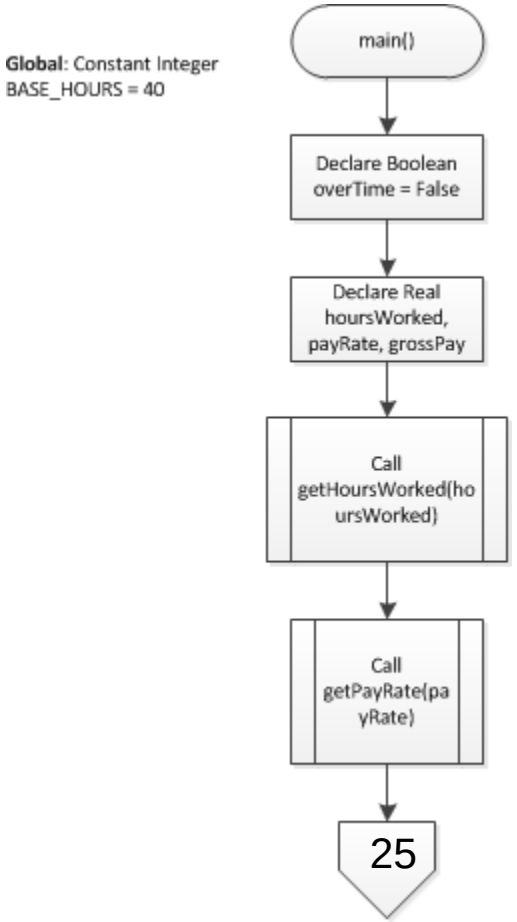
Sample Program: Overtime PayCalculator

- "*When using modules in a program, you generally isolate each task within the program in its own module.*" (p80)
  - Get input
    - Hours worked
    - Pay rate
  - Process input
    - Pay with Overtime, OR
    - Pay without Overtime



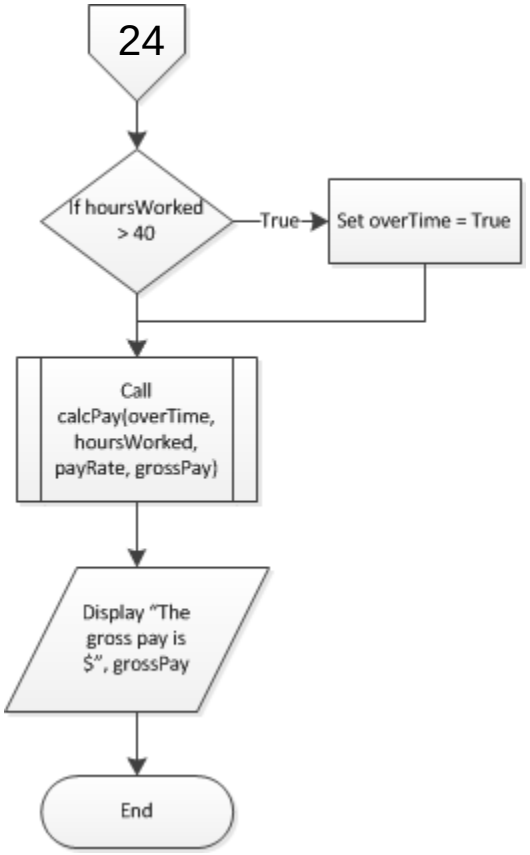
Sample Program: Overtime PayCalculator

```
// Global named constant
Constant Integer BASE_HOURS = 40

Module main()
  // Local variables
  Declare Boolean overTime = False
  Declare Real hoursWorked, payRate, grossPay

  // Get the number of hours worked.
  Call getHoursWorked(hoursWorked)

  // Get the hourly pay rate
  Call getPayRate(payRate)
```



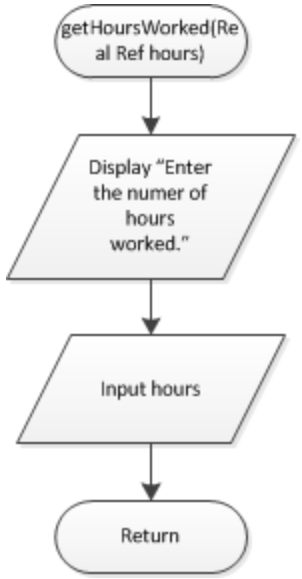
Sample Program: Overtime PayCalculator

```
  // Determine if overtime
  If hoursWorked > BASE_HOURS Then
    Set overTime = True
  End If

  // Calculate the gross pay.
  Call calcPay(overTime, hoursWorked,
    payRate, grossPay)

  // Display gross pay.
  Display "The gross pay is $", grossPay
End Module
```



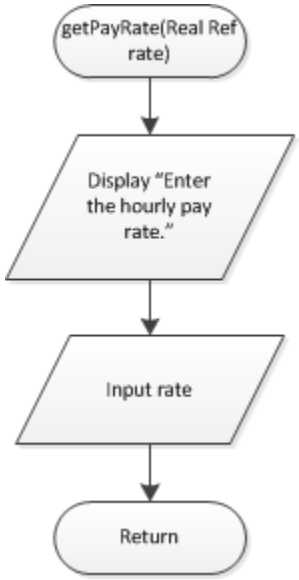
Sample Program: Overtime PayCalculator

```
// The getHoursWorked module gets the number
// hours worked and stores it in the
// reference variable hours.
Module getHoursWorked(Real Ref hours)
  Display “Enter the number of hours worked.”
  Input hours
End Module
```



Sample Program: Overtime PayCalculator

```
// The getPayRate module gets the hourly
// pay rate and stores it in the
// reference variable rate.
Module getPayRate(Real Ref rate)
  Display “Enter the hourly pay rate.”
  Input rate
End Module
```



Sample Program: Overtime PayCalculator

```
// The calcPay module calculates gross pay
// with or without overtime.
Module calcPay(Boolean ot, Real hours,
      Real rate, Real Ref gross)
  // Local named constant
  Constant Real OT_MULTIPLIER = 1.5

  // Local variables
  Declare Real otHours, otPay

  // Calculate gross pay.
  If ot Then
    Set otHours = hours – BASE_HOURS
    Set otPay = otHours * rate * OT_MULTIPLIER
    Set gross = BASE_HOURS * rate + otPay
  Else
    Set gross = hours * rate
  End If
End Module
```