

# PROGRAMMING CONCEPTS

Chapter 3

# Communicating with the computer

۲

- ❑ you have to learn its system of communication or language .
- ❑ The meaning of an instruction is essentially the same in many computer language or application.
- ❑ **Syntax** refers to the rules governing the computer operating system , the application, and the languages.
- ❑ An errors is called a **bug** .
- ❑ A process called **debugging** .

# Communicating with the computer

۳

- many bugs are result of syntax errors but some are logic errors .
- All syntax errors must be corrected before you execute and test your program .

# Organizing the Problem

4

- Certain organizational tools will help you learn to solve problems on the computer .
  
- The tools include the:
  - ▣ Problem Analysis Chart (PAC)
  - ▣ Structure/Interactivity Charts
  - ▣ Input Processing Output (IPO) Chart
  - ▣ Algorithms.
  - ▣ Flowcharts

# Analyzing the problem

- Understand the Problem
- Analyze the Requirements of the Problem
- A good way to analyze a problem is to separate it into four parts, problem analysis chart( PAC) :
  1. *The given data.(constant and variables)*
  2. *The required results.(the out put)*
  3. *The processing that is required in the problem  
.(equations and expressions)*
  4. *A list of solution alternatives .*

# Problem Analysis Chart (PAC)

6

Given Data	Required Results
Section 1: Data given in the problem or provided by the user. These can be known values or general names for data, such as price, quantity, and so forth.	Section 2: Requirements for the output reports. This includes the information needed and the format required.
Processing Required	Solution Alternatives
Section 3: List of processing required. This includes equations or other types of processing, such as sorting, searching, and so forth.	Section 4: List of ideas for the solution of the problem.

# PAC Payroll Example

v

Calculate the gross pay of an employee. The formula to be used is

$$\text{GrossPay} = \text{Hours} * \text{PayRate}$$

Develop PAC for a solution to this problem?

# PAC Payroll Example

^

Given Data	Required Results
Hours Pay Rate	Gross Pay
Processing Required	Solution Alternatives
$GrossPay = Hours * PayRate$	<ol style="list-style-type: none"><li>1. Define the hours worked and pay rate as constants.</li><li>*2. Define the hours worked and pay rate as input values.</li></ol>

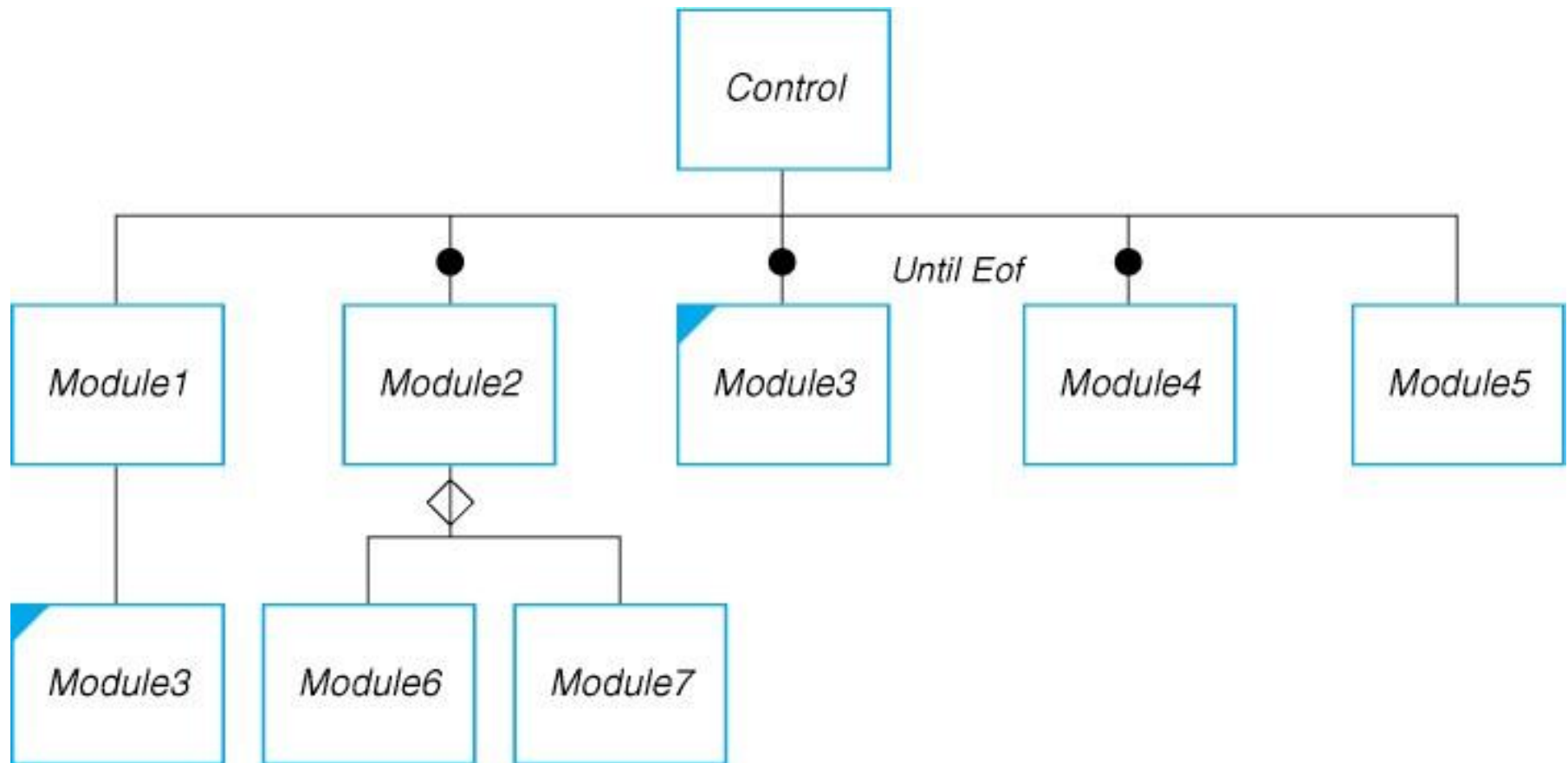


# Interactivity (Structure) Chart

9

- ❑ Divide processing into subtasks called **modules**
- ❑ Then connect these **modules** together to show the interaction of processing between modules.
- ❑ Each modules should contain the tasks to accomplish one function.
- ❑ There will be one module that controls the flow to most of other modules called the ***Control or main module*** .
- ❑ The subtasks of this module are then located below it in the structure chart .

# The Interactivity Chart

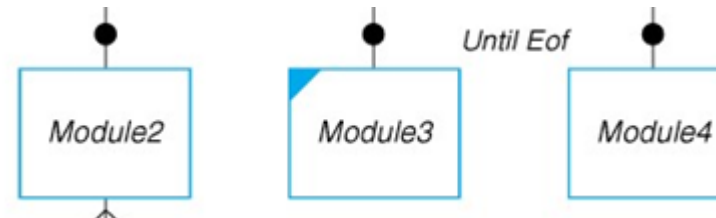


# The Interactivity Chart

- Indicates duplicate modules by darkening the upper left-hand corner of each module.



- The darkened circles indicate that the module is part of a set of modules that are processed many time – those in loop .



# The Interactivity Chart

۱۲

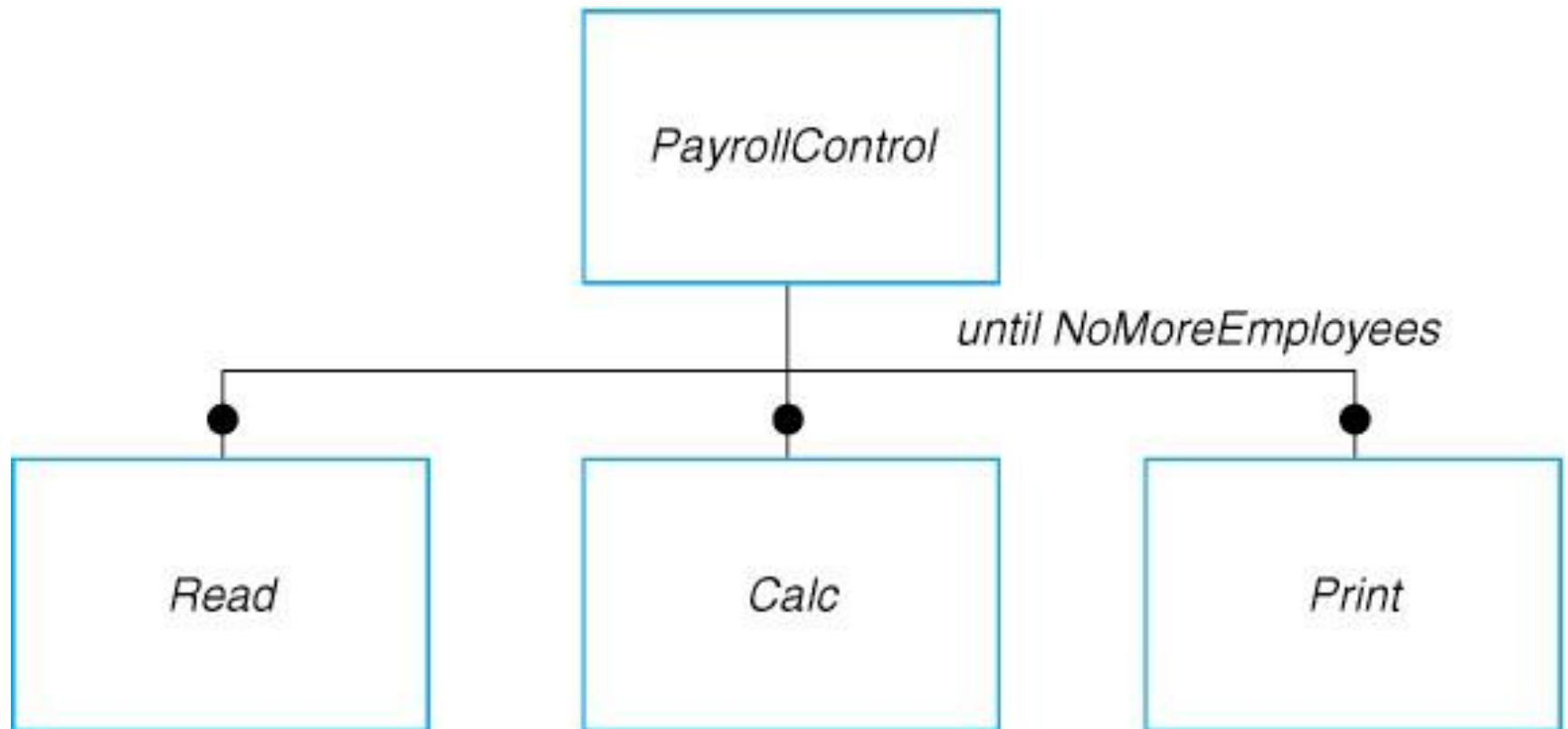
- The diamond on the vertical line above the box indicates which modules are involved in a decision .



- Annotation may be placed beside the circle or diamond to indicate the condition

# The Interactivity Chart for the Payroll Problem

۱۳



# Developing Input Processing Output (IPO) Chart

١٤

- IPO chart shows:
  - What data item are input
  - What processing takes place on that data
  - What information will be the end result, the output
  - Where in the solution the processing takes place.

Input	Processing	Module Reference	Output
All input data (from Section 1 of the problem analysis chart)	All processing in steps (from Sections 3 and 4 of the problem analysis chart)	Module reference from the interactivity chart	All output requirements (from Sections 1 and 2 of the problem analysis chart)

# The IPO Chart for the Payroll Problem

۱۵

Input	Processing	Module Reference	Output
Hours Worked Pay Rate	1. Enter Hours Worked 2. Enter Pay Rate 3. Calculate Pay 4. Print Pay 5. End	<i>Read</i> <i>Read</i> <i>Calc</i> <i>Print</i> <i>PayRollControl</i>	Gross pay

- The module references are show which will perform each step in the processing

# Writing the Algorithm

۱۶

- The next step of organizing a solution is to develop sets of instructions for the computer, called algorithms.
- The programmer writes a separate set of instructions for each module in the structure chart .
- The number of instruction is determined by the way the programmer chooses to solve the problem .



## The Form of an Algorithm

<i>Control Module</i>	<i>Name of Module (list of parameters)</i>
1. <i>Instruction</i>	1. <i>Instruction</i>
2. <i>Instruction</i>	2. <i>Instruction</i>
3. ..	3. ..
4. ..	4. ..
..	..
— . end	— , exit

- The **Control** module uses an **End** since this is the end of the processing
- The other modules use **Exit** because the processing continues .

# Drawing the Flowchart

۱۸

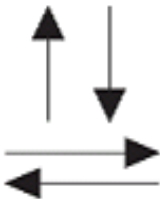


- Graphic representations of the algorithms.
- The algorithms and flowcharts are the final steps in organizing a solution.
- A flowcharts shows the flow of the processing from the beginning to the end of a solution.
- Each block in a flowchart represents one instruction from an algorithm.



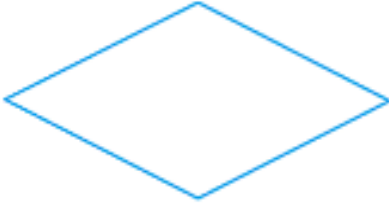
# Drawing the Flowchart

۱۹


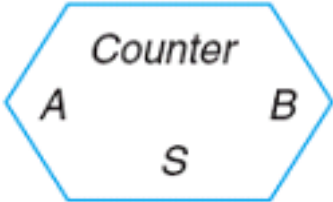
- **Flowlines** indicate the direction of the data flow.
- Most block have one or more *entrances*
- Most block have only one exit .
- Since ,in most cases , data can flow to only one other block .
- Exception a block representing a decision instruction
- A loop enables the computer to perform a task repeatedly during the processing of solution .

# Flowchart Symbols

Flowchart Symbol	Explanation
 Flowlines	Flowlines are indicated by straight lines with optional arrows to show the direction of data flow. The arrowhead is necessary when the flow direction might be in doubt. Flowlines are used to connect blocks by exiting from one and entering another.
 Start  End/Stop/Exit	Flattened ellipses indicate the start and the end of a module. An ellipse uses the name of the module at the start. The end is indicated by the word <i>end</i> or <i>stop</i> for the top or <i>Control</i> module and the word <i>exit</i> for all other modules. A start has no flowlines entering it and only one exiting it; an end or exit has one flowline entering it but none exiting it.

 <p>Processing</p>	<p>The rectangle indicates a processing block, for such things as calculations, opening and closing files, and so forth. A processing block has one entrance and one exit.</p>
 <p>I/O</p>	<p>The parallelogram indicates input to and output from the computer memory. An input/output (I/O) block has one entrance and only one exit.</p>
 <p>Decision</p>	<p>The diamond indicates a decision. It has one entrance and two and only two exits from the block. One exit is the action when the resultant is <i>True</i> and the other exit is the action when the resultant is <i>False</i>.</p>

# Flowchart Symbols

Flowchart Symbol	Explanation
 Process Module	Rectangles with lines down each side indicate the process of modules. They have one entrance and only one exit.
 Automatic-Counter Loop	The polygon indicates a loop with a counter. The counter starts with <i>A</i> (the beginning value) and is incremented by <i>S</i> (the incrementor value) until the counter is greater than <i>B</i> (the ending value). <i>Counter</i> is a variable. <i>A</i> , <i>B</i> , and <i>S</i> may be constants, variables, or expressions.

# Flowchart Symbols



On-Page Connectors\*

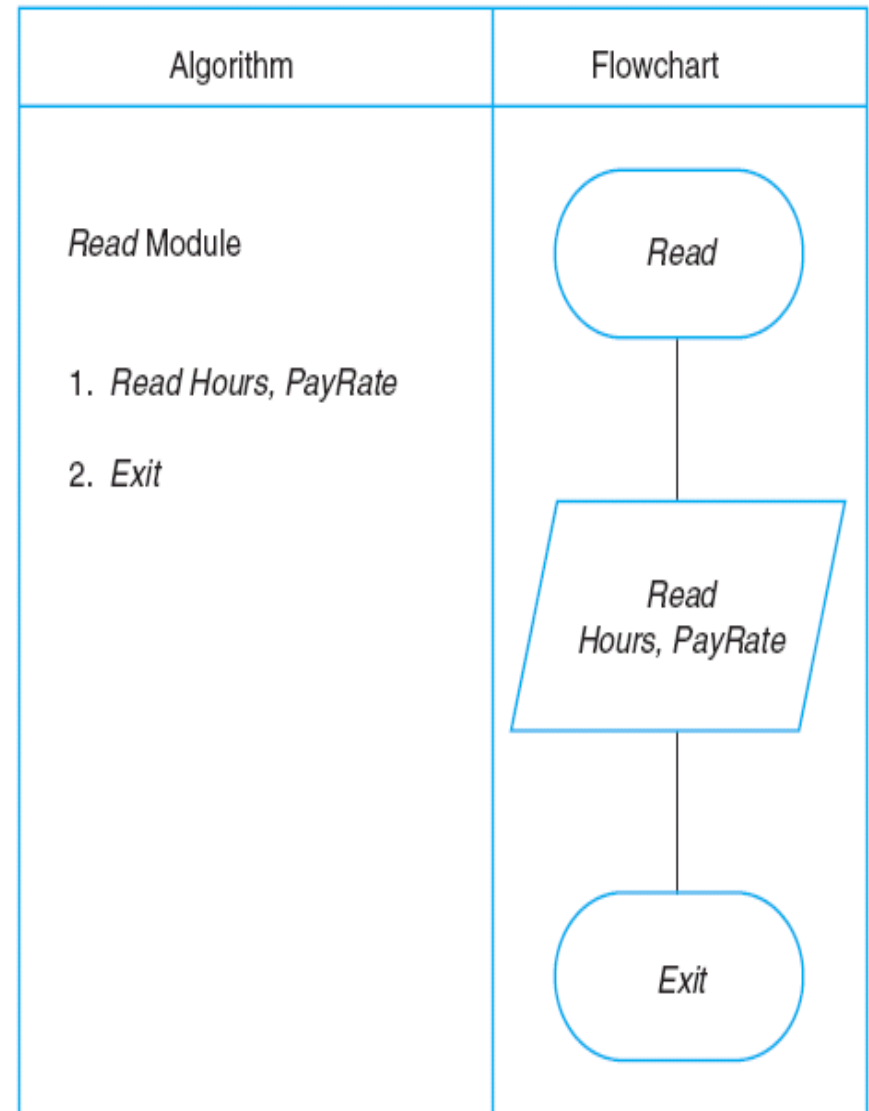
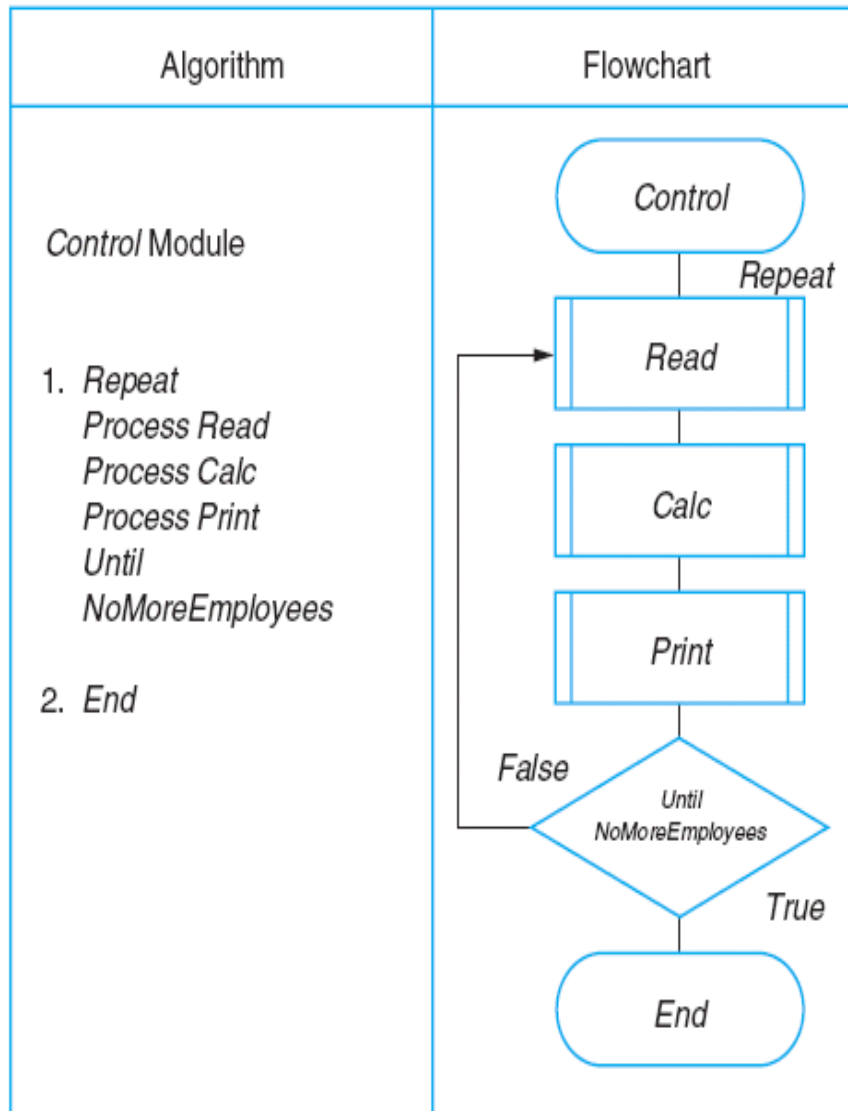


Off-Page Connectors\*

Flowchart sections can be connected with two different symbols. The circle connects sections on the same page, and the home base plate connects flowcharts from page to page. Inside these two symbols the programmer writes letters or numbers. The on-page connector uses letters inside the circle to indicate where the adjoining connector is located. An *A* connects to an *A*, a *B* to a *B*, etc. The off-page connectors use the page number where the next part or the previous part of the flowchart is located. This allows the reader to easily follow the flowchart. On- and off-page connectors will have either an entrance or an exit.

# The algorithms and Flowcharts for the Payroll Problem

٢٤



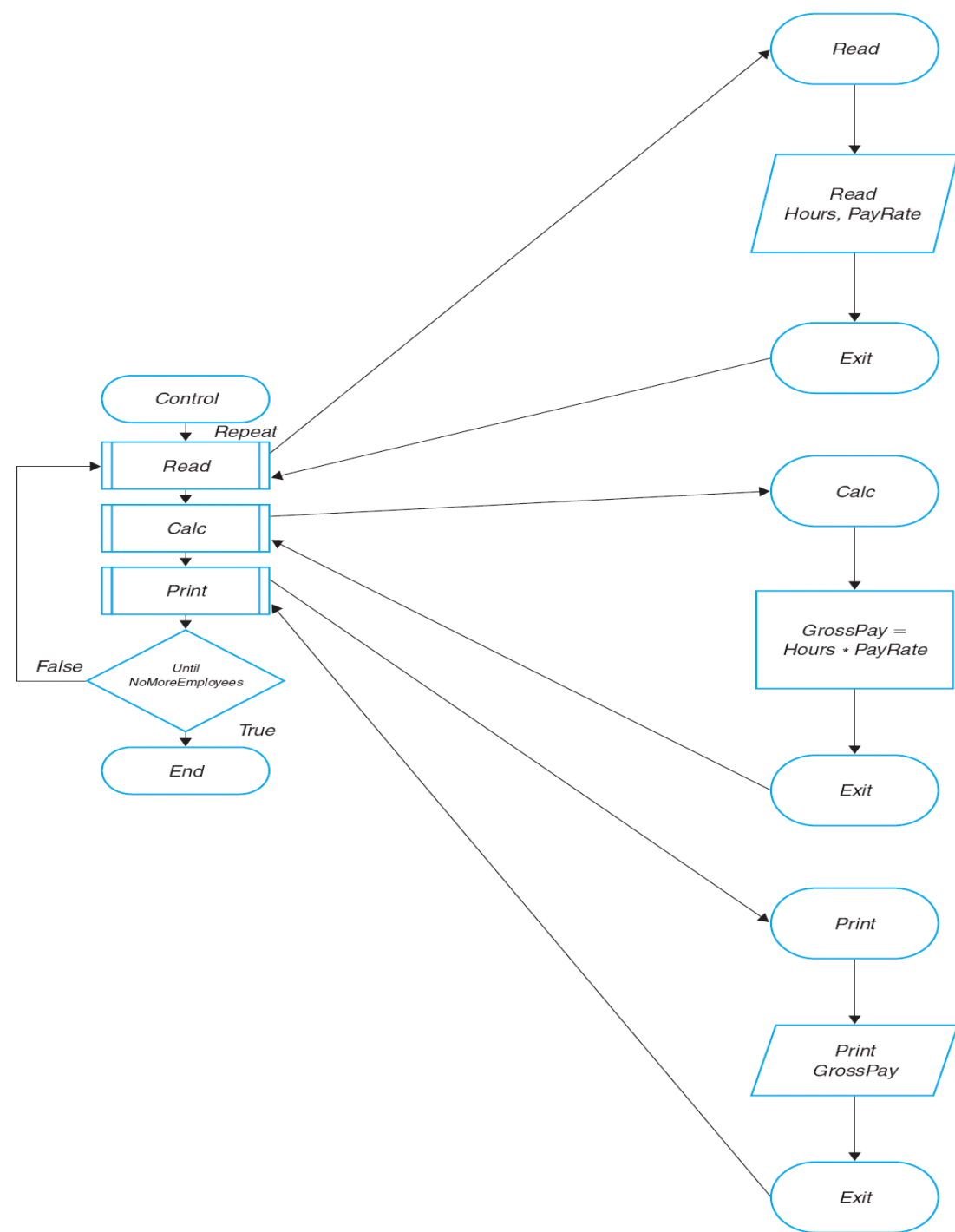


# The algorithms and Flowcharts for the Payroll Problem

٢٥

Algorithm	Flowchart	Algorithm	Flowchart
<p><i>Calc Module</i></p> <ol style="list-style-type: none"><li>1. <math>GrossPay = HoursWorked * PayRate</math></li><li>2. <i>Exit</i></li></ol>	<pre>graph TD; A([Calc]) --&gt; B[GrossPay = Hours * PayRate]; B --&gt; C([Exit])</pre>	<p><i>Print Module</i></p> <ol style="list-style-type: none"><li>1. <i>Print Pay</i></li><li>2. <i>Exit</i></li></ol>	<pre>graph TD; A([Print]) --&gt; B[/Print GrossPay/]; B --&gt; C([Exit])</pre>

# Order of Execution of Instructions



# Algorithm and Flowchart Form

۲۷

Algorithm	Flowchart	Annotation	Test	Internal Documentation	External Documentation

# Pseudocode

۲۸

Pseudocode is similar to the algorithm without the numbers and details.

It closely follows the algorithm, but is characteristically closer to what you would write in a computer language.

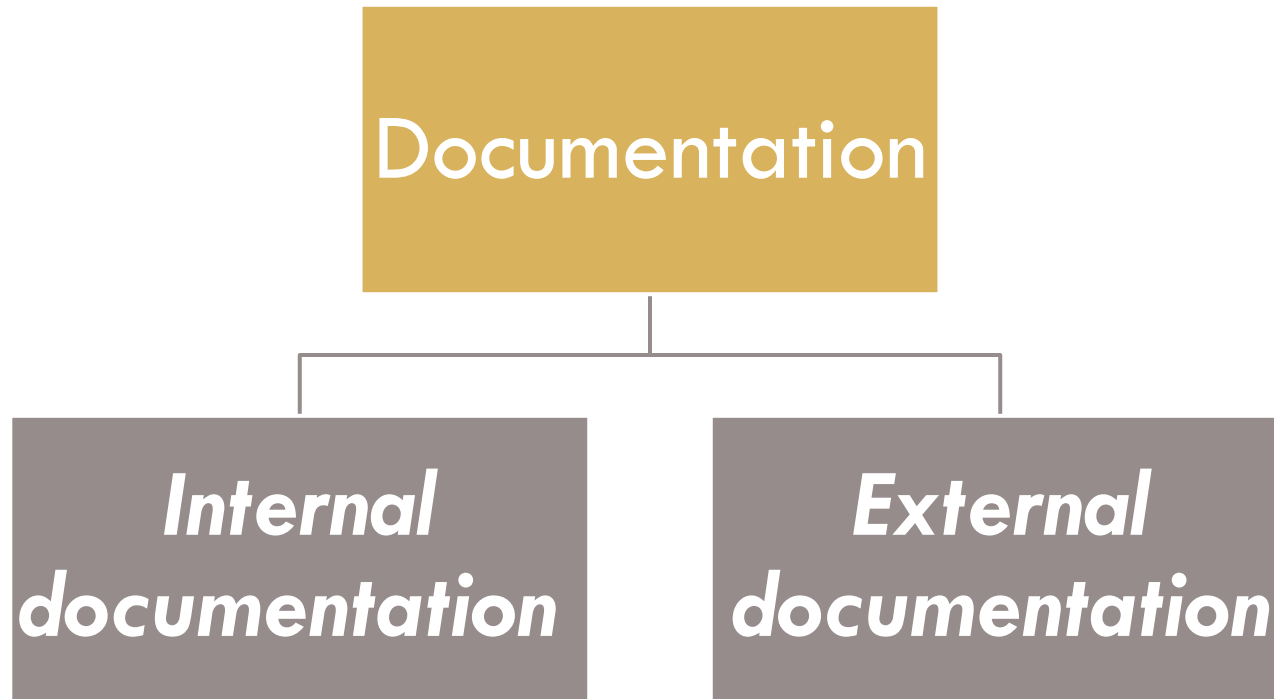
# Pseudocode

٢٩

Algorithm	Flowchart	Pseudocode
<p><i>Control Module</i></p> <ol style="list-style-type: none"> <li>Repeat             <ul style="list-style-type: none"> <li>Process Read</li> <li>Process Calc</li> <li>Process Print</li> </ul>             Until             <ul style="list-style-type: none"> <li>NoMoreEmployees</li> </ul> </li> <li>End</li> </ol>	<pre> graph TD     Control([Control]) -- Repeat --&gt; Read[Read]     Read --&gt; Calc[Calc]     Calc --&gt; Print[Print]     Print --&gt; Decision{Until&lt;br/&gt;NoMoreEmployees}     Decision -- False --&gt; Read     Decision -- True --&gt; End([End])             </pre>	<p>Repeat Process Read Process Calc Process Print Until NoMoreEmployees End</p>

Algorithm	Flowchart	Pseudocode
<p><i>Read Module</i></p> <ol style="list-style-type: none"> <li>Read Hours, PayRate</li> <li>Exit</li> </ol>	<pre> graph TD     Read([Read]) --&gt; ReadIO[/Read&lt;br/&gt;Hours, PayRate/]     ReadIO --&gt; Exit([Exit])             </pre>	<p>Read Hours, PayRate Exit</p>

# Documentation



# Internal Documentation

۳۱

## □ Is a file contains:

- People who worked on the program
- List of all variable name (details).
- Note about the development of the program

# External Documentation

۳۲

- For the user of the program.
- Include any thing that allow the user to learn to use the program in the least amount of time (what should to do and what should not do).



# Coding the solution

۳۳

- ❑ Select the appropriate language
- ❑ Code the program

# Testing the solution

٣٤

- Test the a solution to make sure it meet the requirements of the user ,the all instruction on program is correct without any error if a bug is detected the solution has to be modified to correct it.

# Algorithm Instructions

- For example, the square root function is used in an assignment statement as follows:

$$A = \text{Sqrt}(X) + 7$$

Therefore, the result of the square root of X would be returned in the name of the function. This value is added to 7 and then stored in A.

The *Write* instruction outputs values to the *printer*. The list of information to be printed follows the Write, as in

*Write* Name, Age

- The *Print* instruction is similar to the *Write* instruction expect that the output is directed to the screen rather than the printer.

# Algorithm Instructions (continues)

۳۶

- The **End, Exit, or Return** instruction specifies the completion of a module.
- **End** is used to end the Control module and indicates that the processing of the solution is complete.
- **Exit** is used to end a subordinate module if there is no return value, and indicates that the processing will continue in another module, the module where the Process instruction originated
- The Return (variable) is used to place a value in the name of the module. The Return is used when the module is to be processed within an expression.

# An Example

۳۷

- 1- Enter a name and an age into the computer.
- 2- Print a name and an age on the screen.

The algorithm and flowchart to enter a name and age into computer and print it on the screen.

Notice that the algorithm instructions are numbered starting with the first instruction after the name of module

