Chapter 1
General Problem Solving Concepts

Overview

- Problem Solving in Everyday Life
- Types of Problems
- Problem Solving with Computers
- Difficulties with Problem Solving

Learning Objectives

1. Describe difference between heuristic and algorithmic solutions
2. List and describe SIX (6) problem-solving steps for algorithmic solution
3. Use problem-solving steps to solve problem

Problem Solving in Everyday Life

SIX (6) steps in problem-solving process:

1. Identify the problem
2. Understand the problem – require knowledge base (q4 - pp20)
3. Identify alternative ways to solve problem
4. Select best alternative
5. List solution steps for alternative chosen (q2 - pp20, q5 – pp21)
6. Evaluate solution

Problem Solving in Everyday Life

Example 1:

Let's say you are preparing for examination in hostel / apartment, and you are very hungry right now.

Try to solve the problem above by using six steps of problem solving process.

Types of Problems

Problems can be solved with:

- Algorithmic solutions
- Heuristic solutions
- Combination of algorithmic, heuristic solutions

Algorithmic Solutions

- Can be solved with a series of actions (in steps)
- These steps are called ALGORITHM
- Example:
  - Balancing a cheque book
  - Baking a cake
  - Withdrawing money from ATM machine
  - Paying for your parking ticket via auto pay machine
  - What else?

Heuristic Solutions

- Problem which couldn't be solved through a direct set of steps.
- Require knowledge, experience & a process of trial and error (repeating six steps more than once)
- Example:
  - How to buy best stock from market
  - Should the company be expanded
  - Baking a delicious cake
  - Raising up a kid
  - What else?

Combination of Both

- Most problems require a combination of algorithmic and heuristic solutions
- Example:
  - Repairing a car
  - Driving a car
  - To win in a computer game
  - What else??

Problem Solving with Computers

Definitions:

- Solution ⇔ instructions followed to produce best result
- Result ⇔ outcome, computer-assisted answer
- Program ⇔ instructions for solution using computer language

Difficulties with Problem Solving

- Lack of problem solving experience
- Inadequate solution steps
- Incorrect problem definition
- Alternatives chosen incorrectly
- Invalid logic
- Incorrect solution evaluation

Beginning Problem-Solving Concepts for the Computer

Lesson 2

3 Types of Problems

- Computational
  - problems involving some kind of mathematical processing
- Logical
  - Problems involving relational or logical processing
- Repetitive
  - Problems involving repeating a set of mathematical and/or logical instructions.

Fundamental Concepts

- The building blocks of equations and expressions
  - Constants
  - Variables
  - Operators
  - Functions (predefined—more about user-defined functions later)

Constants

- A value
  - a specific alphabetical and/or numeric value
  - Does not change during the processing of all the instructions in a solution
- Can be of any data type
  - Numeric, alphabetical or special symbols
- Examples
  - 3, 5, 10, “Mulder”, “Scully”, “+”, “-”, “/”
  - Note: “” indicates datum rather than variable name

Constants

- In some programming languages, constants can be named
  - Provides protection to the constant value and other areas of the program.
  - Cannot be changed once given initial value
  - Note: no spaces allowed within names
  - Examples
    - SALES_TAX_RATE = 6
    - COMMISSION_RATE = 6

Constant name are usually in
ALL CAPS

Variables

- The name references the memory or storage location where the value is stored
  - May change during processing
  - May be of any data type
  - A name is assigned to each variable used in a solution.
    - Examples
      - Age, LastName, Address, PayRate

Rules for Naming Variables

- the name should be meaningful
  - Use PayRate not x or y or z
- Do not use spaces
- Do not use special symbols, except the underscore
  - Examples
    - PayRate, Pay_Rate, PAYRATE, or PAY_RATE

- Use appropriate naming convention
- Be consistent!
  - Some languages are case sensitive
  - Either combine terms (PayRate) or include an underscore to separate (Pay_Rate)

Data Types

- Most common types
  - Numeric
  - Character
  - Logical (True/False)
- Most languages include other data types
  - Date
  - Currency
  - User-defined
  - Strings

Rules for Data Types

- All data to be used in calculations must be declared as a numeric data type
- Each data type uses a data set or domain
    - Integers - -32768 -> 0 -> 32767 but is programming language dependent
    - Characters – all alphanumeric characters included in the computer's coding scheme
    - Logical – the words "true" and "false"
- Data types cannot be mixed
- Programmer designates the data type

Data Types & Data Sets

Data Type	*Data Set*	*Examples*
Integer	All whole numbers	3456 -43
Real	All real numbers (whole + decimal)	3456.78 -123.45 0.000123
Character (uses quotation marks)	All letters, numbers, and special symbols	“A”, “a”, “1”, “+”, “%”
String (uses quotation marks)	Combinations of more than one character	“Mulder” “123-45-6789”
Logical	True or False	True False

Numeric Types

- Includes all types of numbers
  - Natural Numbers - The set of positive (counting) numbers
    - Ex. {1,2,3,4,5,…}
  - Integers - The set of real numbers consisting of the natural numbers, their additive inverses, and zero
    - Ex. 3, 5, -5, 0, -1 and 32,767
  - Real numbers (floating point numbers)
    - Ex. 3.1459, 1.618039887, -5745 & Scientific Notation

Character Types

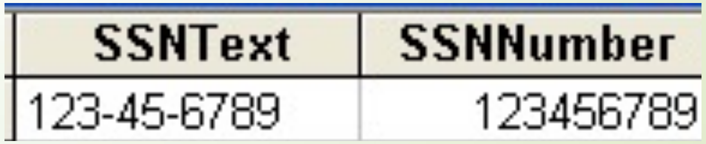
- Alphanumeric data set
    - Consists of
        - all single-digit numbers,
        - letters, and
        - special characters available to the computer
    - contained within quotation marks

String Data Type

- Made up of one or more characters.
- Contained within quotation marks
- Cannot be used within calculations
  - There is some debate about using strings…
    - *The new consensus is that if the value will be used in a mathematical calculation, make it a number, otherwise make it a string.*
    - *The alternative view (older view) is to look at the amount of storage that is needed for the value. A number generally takes up less memory space than a string.*

Strings - Caution

- “123” is not the same thing as 123
  - The first one is a string
  - The second one is a number
- Becomes important when deciding when a piece of data is a string or a number
  - Ex. Social Security Numbers
    - The first one contains 11 characters and is left aligned whereas the second contains 9 numbers and is right aligned



SSNText	SSNNumber
123-45-6789	123456789

Concatenation

- Joins character or string data together
  - Operators: + and &
  - Dependent on programming language
  - & can mix data types
  - + cannot mix data types
  - Examples
    - FName & “ ” & MI & “ ” & LName
    - “James” + “ ” + “T.” + “ ” + “Kirk” = “James T. Kirk”
    - “James” & “ ” & “T.” & “ ” & “Kirk” = “James T. Kirk”
      - Note the space at the end of the T.

Logical (Boolean) Data Type

- Consists just two pieces of data
  - True and False
  - Different programming languages also allow
    - Yes and No
    - 1 (True) and 0 (False)
    - On and Off

Functions

- Small sets of instructions that perform specific tasks and return values.
- Two types:
  - Pre-defined
    - Built into a computer language or application
  - User-defined
    - More about these later

Functions

- Benefits
  - Reduces the amount of code that needs to be written, thus reducing errors of repetition
  - Shortens the development time
  - Improves readability

Function Parameters

- Data usually placed within parentheses that the function uses without altering the data
  - In Sqrt(n), the n represents the parameter, in this case a number

Function Types

- Mathematical (sqrt, abs, round, etc.,)
- String (length, mid, left, etc.,)
- Conversion (string, int)
- Statistical (Average, max, min, sum)
- Utility (Date, Time)
- Specific to each programming language

Operands, Resultants & Operators

- Operands
  - The data that the operator connects and processes
- Resultant
  - The answer that results when the operation is executed

- Operators
  - The data connectors within expressions and equations.
  - Two types of operations
    - Unary
      - Negation
    - Binary
      - Addition, Subtraction, Multiplication, Floating Point Division and Integer Division

Operator Types

- Mathematical
  - +, -, *, /
  - //, Mod {Integer Division & Modulo}
  - ^, {Exponentiation}
- Relational
  - <, >, <=, >=, =, <>
- Logical
  - And, Or, Not

The Boolean Data Type

- Returns either the keywords True or False or
- Returns a non-zero or zero value.
  - -9, -1, 1, 2 are equivalent to True
  - 0 is equivalent to False
- Is stored in two bytes

Relational Operators

- Perform comparisons
  - Ex. If Y > Z then ...

=	equal
<>	not equal
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Logical (Boolean) Operators

- And, Or, Not
- Boolean Expressions are used
  - To create more complicated Boolean Expressions.
  - If a = b AND c < d Then …
- They are defined by using Truth Tables…
  - Know the significance of each!

The AND Operator

- ***Significance: The only time the result is True is if both A and B are True.***

A =	***B = then***	***A AND B***
True	True	True
True	False	False
False	True	False
False	False	False

The OR Operator

- ***Significance: The only time the result is False is if both A and B are False.***

A =	***B = then***	***A OR B***
True	True	True
True	False	True
False	True	True
False	False	False

The NOT Operator

Significance: Whatever the value of A is, NOT A will be the opposite.

A =	*NOT A*
True	False
False	True

Hierarchy of Operations

- Data and Operators are combined to form expressions and equations
- To evaluate these in the proper order, a hierarchy of operations, or Order of Precedence, is used.
  - Note: Different programming languages use different Order of Precedence…
  - Note: Whenever you want to clarify an equation, use the parentheses!

The Order of Precedence

OPERATOR	*TYPE*	*NAME*
( )	Parentheses	Parentheses
^	Arithmetic	Exponent (Powers)
* /	Arithmetic	Multiplcation, floating-point division
\	Arithmetic	Integer division
Mod	Arithmetic	Modulus
+ -	Arithmetic	Addition, subtraction
= <> <= >= > <	Comparison (all have same level of precedence	Equal to, not equal to, less than or equal to, greater than or equal to, greater than, less than
NOT	Logical	Logical Negation
AND	Logical	Logical And
OR	Logical	Logical Or
XOR	Logical	Logical Exclusive Or
EQV	Logical	Logical Equivalent

Expressions & Equations

- Expressions
  - Process the data and the operands, through the use of the operators
  - Does not store the resultant (answer)
  - Examples
    - Price * SalesTaxRate
    - (Hours – 40) * Wage * 1.5
    - Height * Width

Equations

- Same as an expression, except the equation stores the resultant (answer) in a memory location (must be on the left side of Assignment Operator)
  - Examples

```
SalesTax          = Price * SalesTaxRate
OvertimePay       = (Hours – 40) * Wage * 1.5
Area              = Height * Width
```

Planning Your Solution

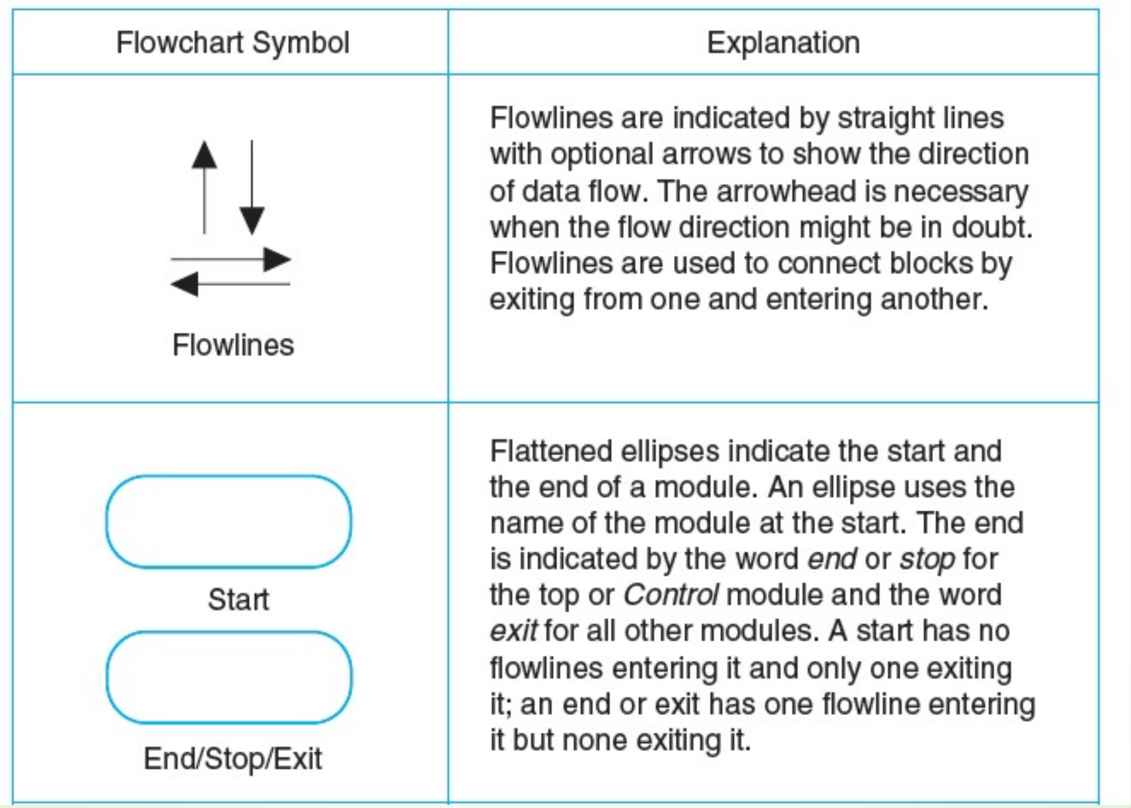
Lesson 3

Analyzing the Problem

Understand requirements:

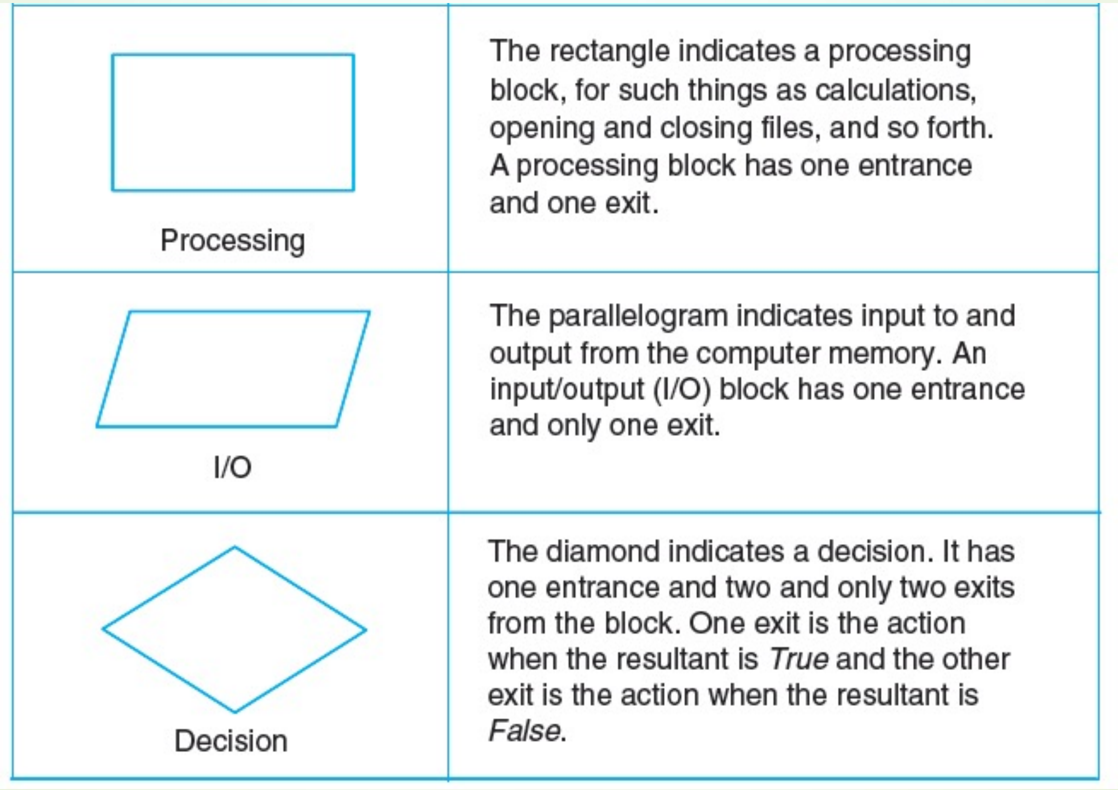
1. The given data
2. The required results
3. The processing that is required in the problem
4. A list of solution alternatives

Flowchart Symbols



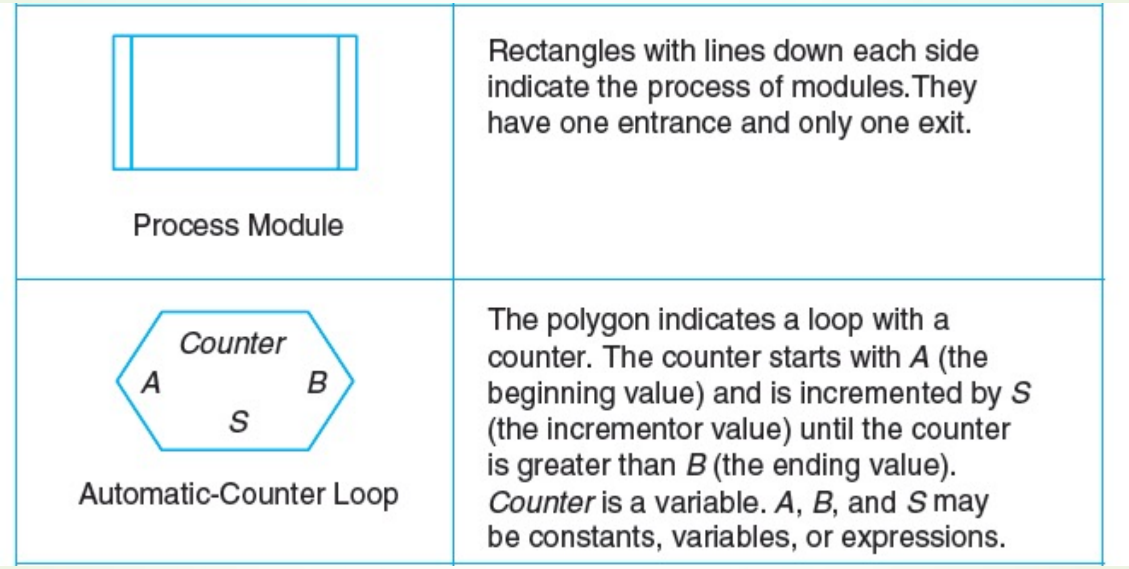
Flowchart Symbol	Explanation
Flowlines	Flowlines are indicated by straight lines with optional arrows to show the direction of data flow. The arrowhead is necessary when the flow direction might be in doubt. Flowlines are used to connect blocks by exiting from one and entering another.
Start End/Stop/Exit	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 *end* or *stop* for the top or *Control* module and the word *exit* 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.

Flowchart Symbols



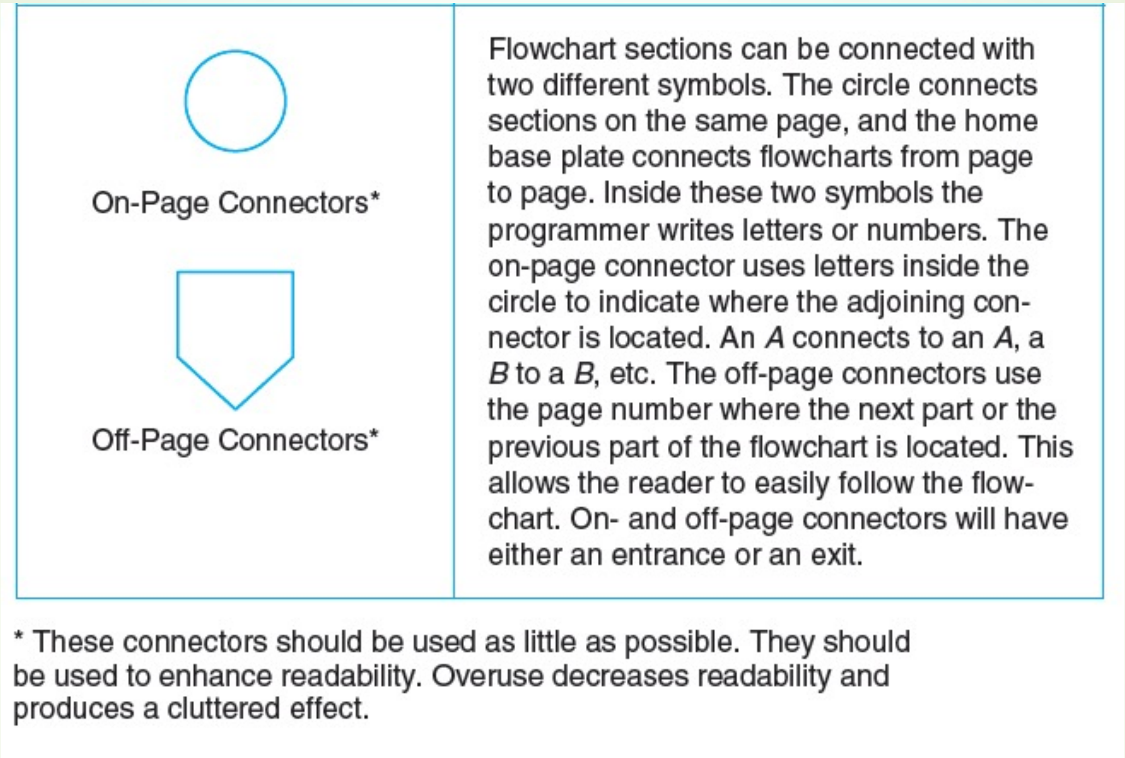
Processing	The rectangle indicates a processing block, for such things as calculations, opening and closing files, and so forth. A processing block has one entrance and one exit.
I/O	The parallelogram indicates input to and output from the computer memory. An input/output (I/O) block has one entrance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

Flowchart Symbols



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

Figure 3.9 Flowchart Symbols



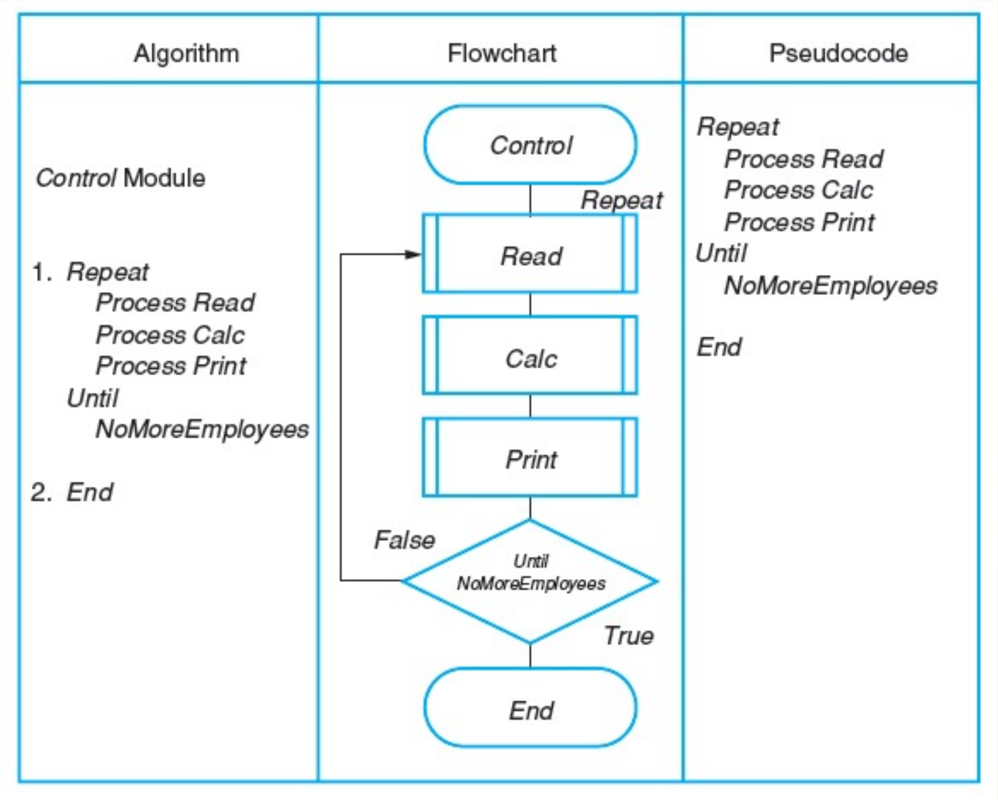
On-Page Connectors*

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.

* These connectors should be used as little as possible. They should be used to enhance readability. Overuse decreases readability and produces a cluttered effect.

The Algorithms and Flowcharts for the Payroll Problem



Algorithm

Control Module

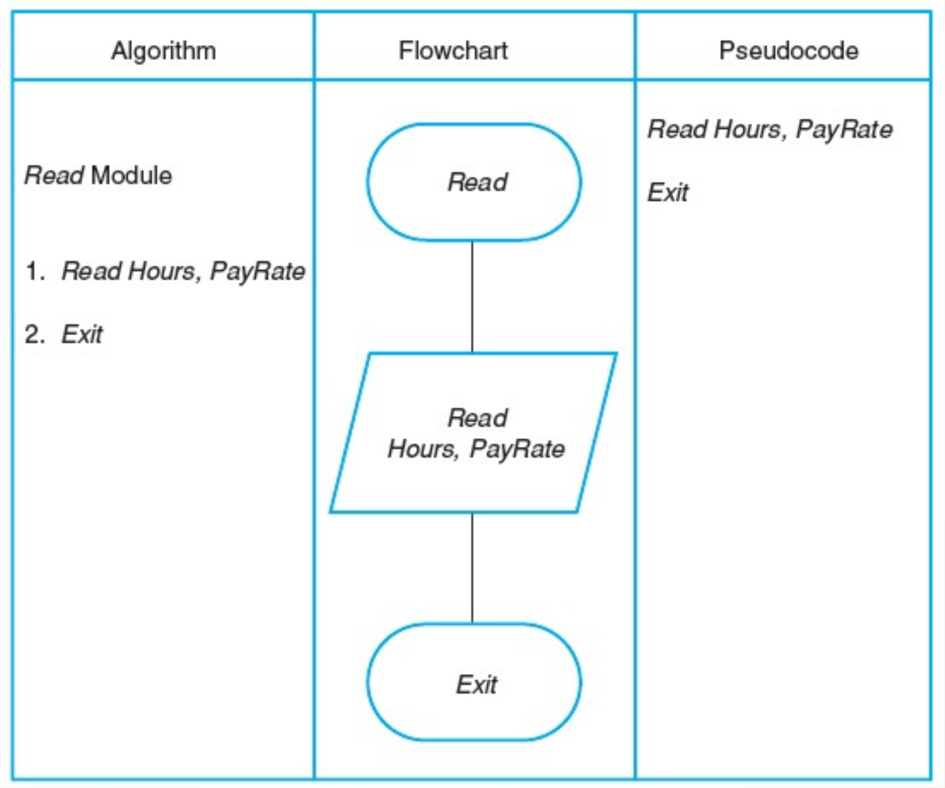
```
1. Repeat
       Process Read
       Process Calc
       Process Print
   Until
       NoMoreEmployees

2. End
```

Flowchart

Pseudocode

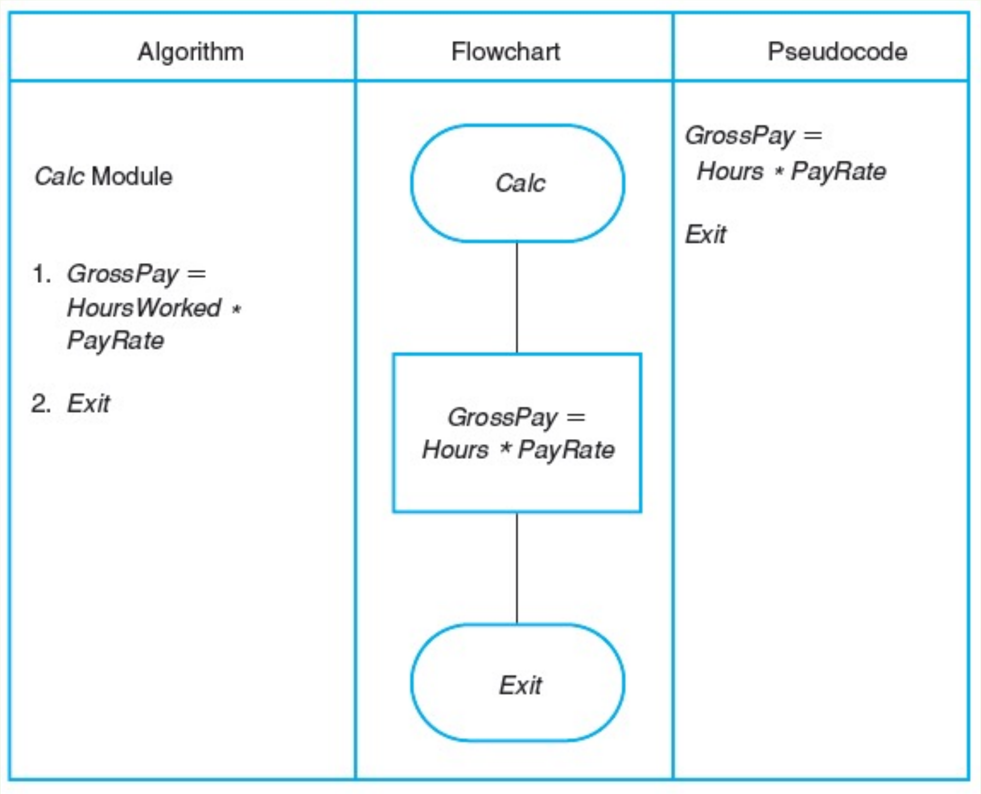
```
Repeat
    Process Read
    Process Calc
    Process Print
Until
    NoMoreEmployees

End
```

The Algorithms and Flowcharts for the Payroll Problem



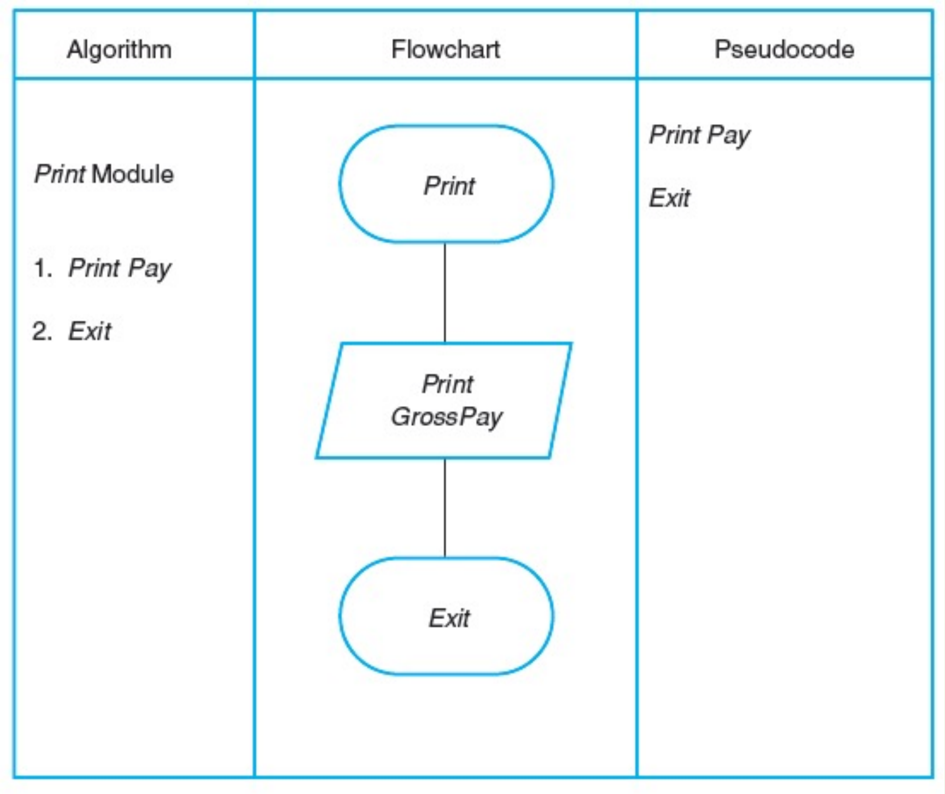
Algorithm	Flowchart	Pseudocode
Read Module 1. *Read Hours, PayRate* 2. *Exit*	*Read* *Read Hours, PayRate* *Exit*	*Read Hours, PayRate* *Exit*

The Algorithms and Flowcharts for the Payroll Problem



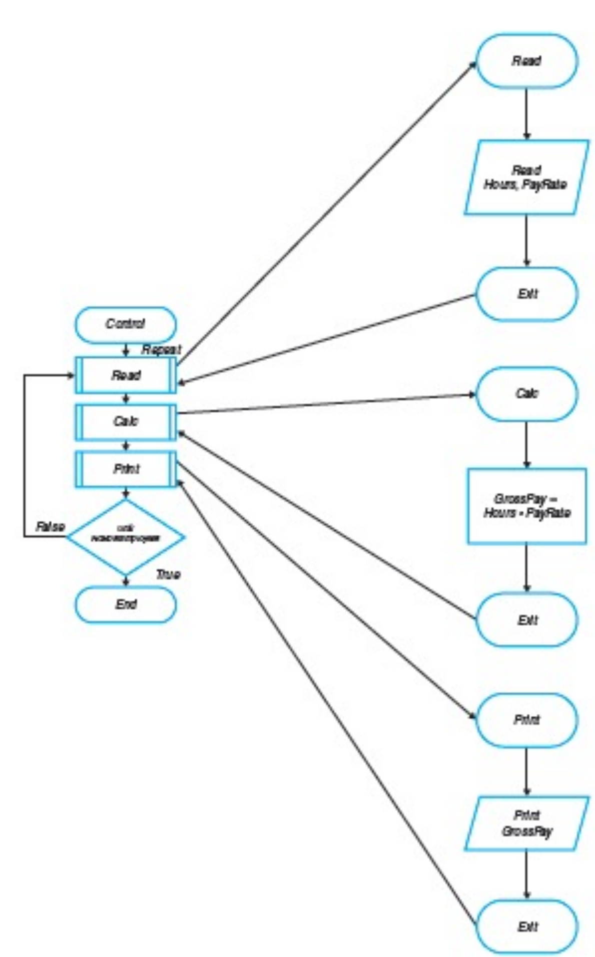
Algorithm	Flowchart	Pseudocode
Calc Module 1. *GrossPay* = *HoursWorked* * *PayRate* 2. *Exit*	*Calc* *GrossPay* = *Hours* * *PayRate* *Exit*	*GrossPay* = *Hours* * *PayRate* *Exit*

The Algorithms and Flowcharts for the Payroll Problem

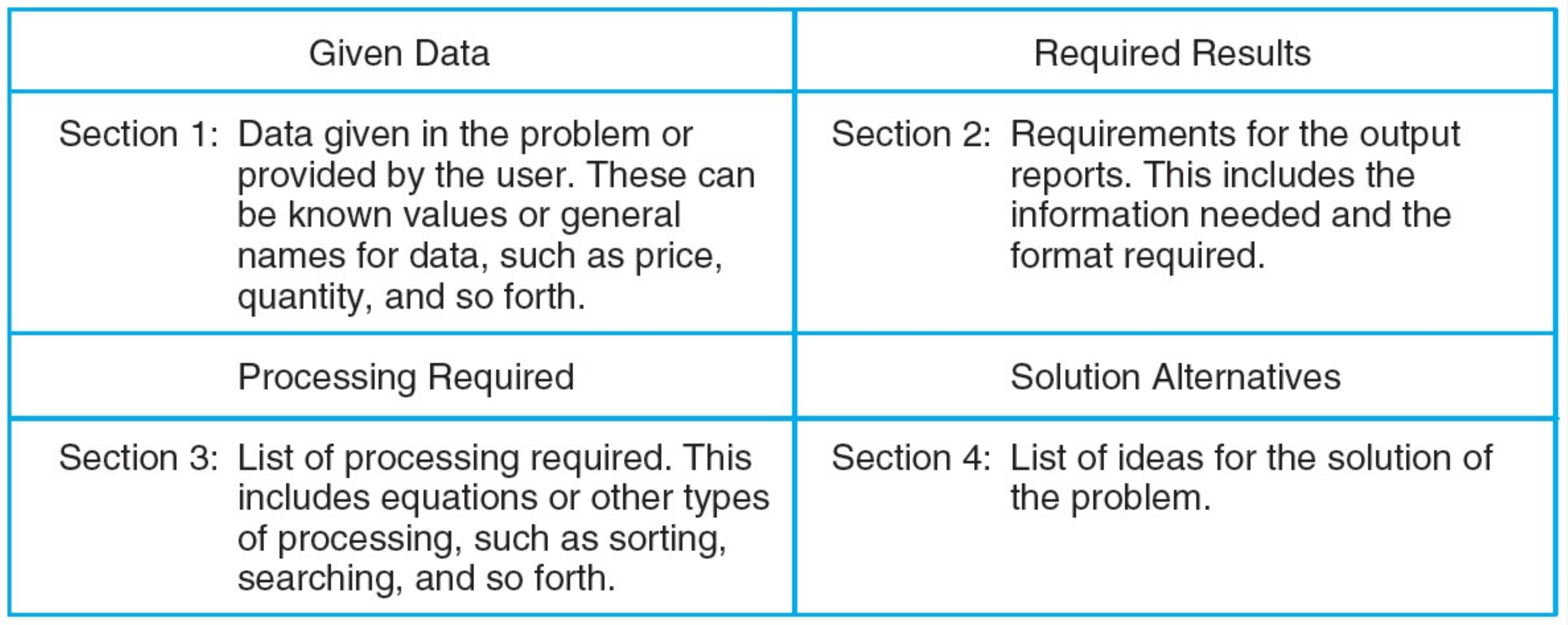


Algorithm	Flowchart	Pseudocode
Print Module 1. *Print Pay* 2. *Exit*		*Print Pay* *Exit*

Order of Execution of Instructions

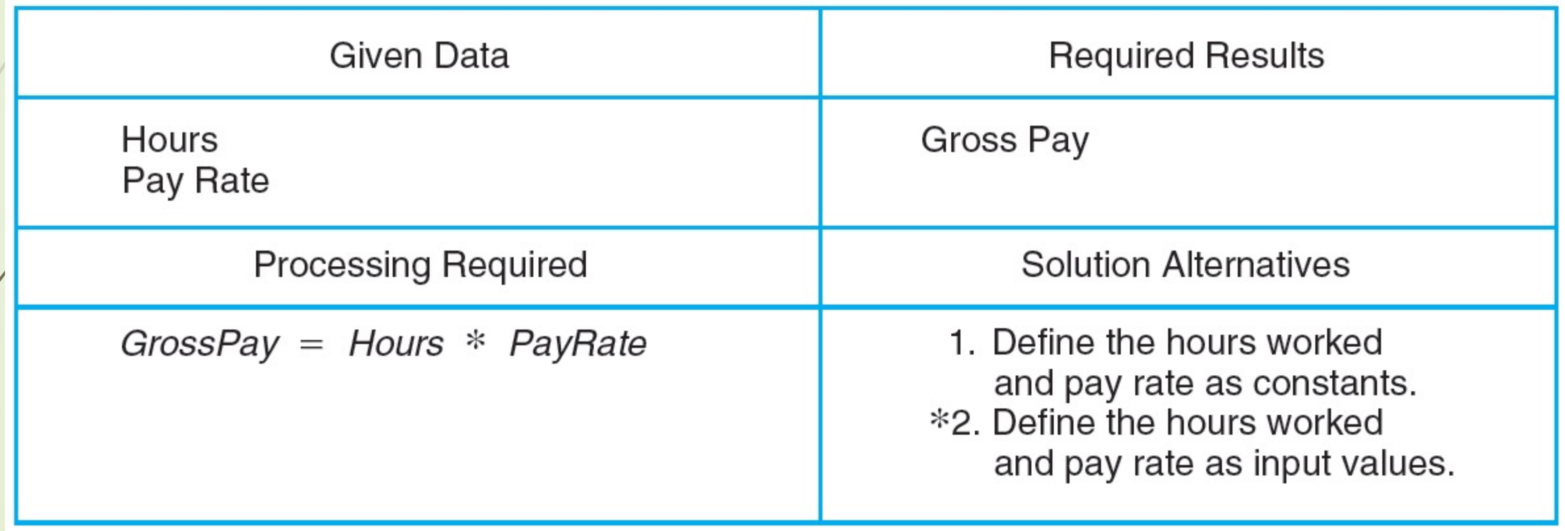


Problem Analysis Chart



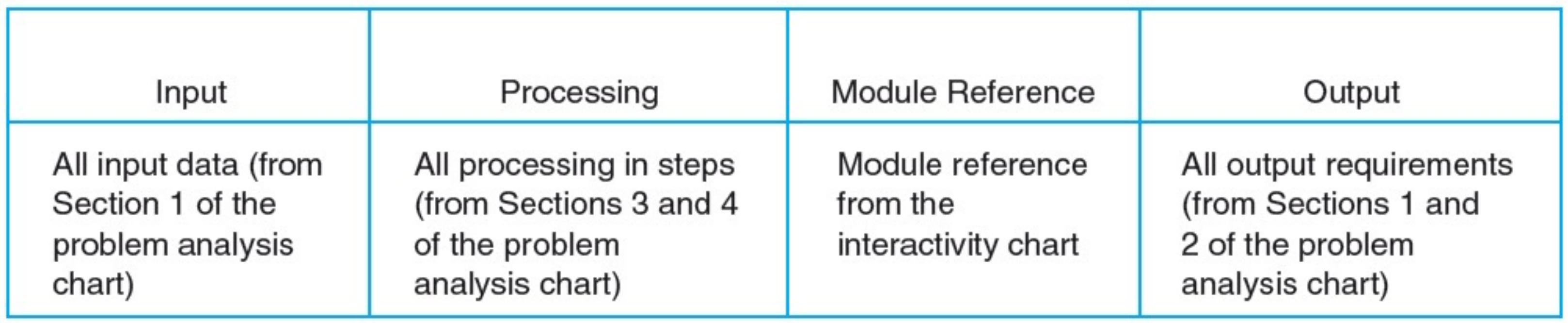
Given Data	Required Results
Section 1: Data given in the problem or provided by the user. These can be known values or general names for data, such as price, quantity, and so forth.	Section 2: Requirements for the output reports. This includes the information needed and the format required.
Processing 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.

Problem Analysis Chart for the Payroll Problem



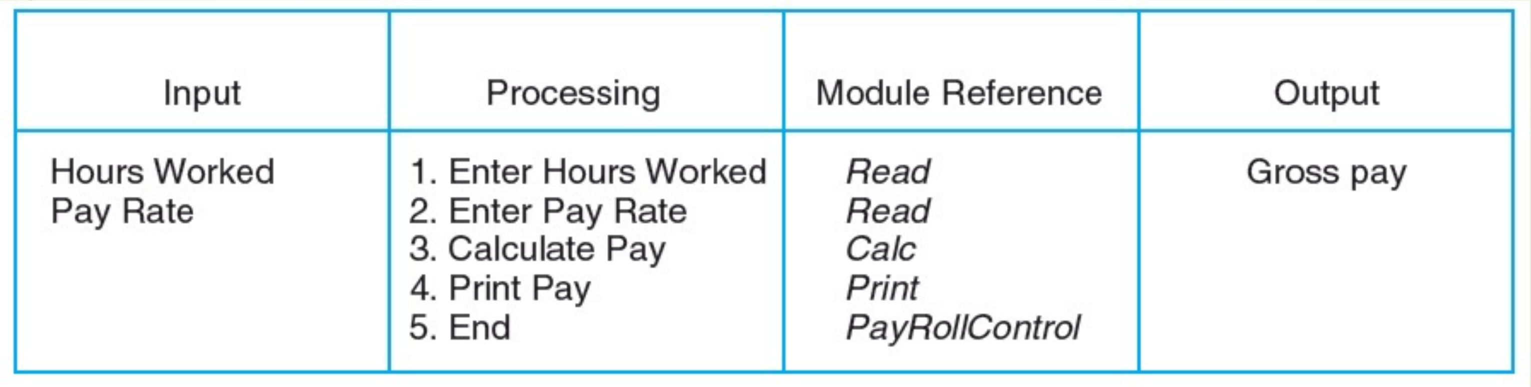
Given Data	Required Results
Hours Pay Rate	Gross Pay
Processing Required	Solution Alternatives
$GrossPay = Hours * PayRate$	1. Define the hours worked and pay rate as constants. *2. Define the hours worked and pay rate as input values.

The IPO(input-processing-output) Chart



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



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