Program Logic Formulation
Computer Software

• Refers to all computer programs or routines which direct and control the computer hardware in the performance of its data processing functions.
Major Categories of Software

- Application software
- System software
Application Software

• A software that perform a specific task for computer users (e.g., Word processing).
System Software

• A software that was designed to **help** programmers **or** to **control** the computer system (e.g. OS; programming tools: translators, debuggers, editors).
Systems Analysis

• The computer age equivalent of the scientific method.

• In systems analysis, the problem is defined and analyzed and solutions are recommended to management.

• This process has been formalized and called the Systems Analysis and Design Method.
System Definition

• A collection of people, machines, & methods organized to accomplish a set of specific functions.

• An integrated whole that is composed of diverse, interacting, specialized structures and subfunctions.

• A group of subsystems united by some interaction or interdependence, performing many duties but functioning as a single unit.
Computer-based IS

- A computer-based information system fits all three definitions. It involves people, machines, and methods (procedures & data); it behaves as an integrated whole; and most information systems are composed of a group of subsystems.
General Systems Theory

• Provides another way of defining a system – as a collection of inputs, outputs, and processor activities with feedback, a boundary, and an environment.
Systems Analyst

- A person trained in the analysis of complex business system, which typically involve a computer-based information system.
Systems Development Life Cycle

• SDLC is used to describe the steps or phases a system goes through from the time it is conceived until it is phased out of existence.

• The names and number of phases in a system life cycle vary with different industries, organizations within industries, & systems analysts.
Systems Development Life Cycle

1. Problem identification, opportunities, objectives
2. Determining information requirements
3. Analyzing system needs
4. Designing the recommended system
5. Developing and documenting the software
6. Testing and maintaining the system
7. Implementing & evaluating the system
Notes on SDLC

• Identifying these phases provides a framework for controlling, estimating, and observing the evolution of a system.

• Each phase produces documentation, w/c is written or pictorial information that describes the system.

• This documentation in turn forms the basis of a management review that determines whether the project should proceed to the next phase.
Software Development Process

• In this process user needs are translated into software requirements; software requirements are transformed into design; the design is implemented in code; and the code is tested, documented, and certified for use.
Notes on Software Development

• **Top-down Software Design**
  - The process of designing a program by first identifying its major components, which are called modules; then decomposing them into lower level components.

• **Structured Program Walkthrough**
  - A method of program inspection wherein a segment of design or code is presented for questioning; comments about technique, style, possible errors, deviation from standards, & other problems.
Notes on Software Development

• Alpha Test
  – A program state during the structured code walkthrough.

• Beta Test
  – The software is not certified or approved for routine use until it has survived this type of test.
  – Beta testing is done by users who accept the pioneer status associated with being first-time users.
Documentation Categories

1. Design documentation
   • Contains data flow diagrams; functional, performance, & data storage requirements; & management reports.

2. Programming documentation
   • Used by technical people to define the architecture of an information system.

3. User documentation
   • Consists of training, operations, & reference manuals that describe how to use the system.
Installation Phase

• A phase in which a system is integrated into its operational environment and is tested to ensure that it performs as required.

• Putting the system into use by the people who must run and maintain it.

• People must be trained, new procedures instituted, & the system thoroughly “shaken down” to eliminate any error or deficiencies.
Methods of Conversion

- Direct
- Parallel
- Pilot
- Phased
Programming

• Program
  – List of computer instructions required to arrive at the desired results.

• Algorithm
  – A precisely expressed procedure for obtaining the problem solution.
Programming

• Programming Language
  – A formalize notation that allows algorithms to be presented in a rigorous and precise way.
  – Types: machine language, assembly language, high-level language, very high-level language (or 4GL)
Programming

• Algorithm may be expressed through:
  – Flowchart
    • A system of symbols for expressing algorithms; indicates flow of control/sequence of operations.
  – Pseudocode
    • A textual representation of an algorithm; close to natural language; becomes part of the program documentation.
Programming

- Program Flowchart Symbols
- Commonly-used Pseudolanguage Instructions (Operations)
Pseudolanguage Instructions

Format:

\[
\text{OPERATION} \ <\text{operand/s}> \\
\]

where:

\[
\begin{align*}
\text{OPERATION} & \quad \text{pseudocode operation or instruction} \\
\text{operand/s} & \quad \text{object/s of a pseudocode operation or instruction}
\end{align*}
\]
Pseudolanguage Instructions

• **OPEN**
  – Allows the program that contains the statement to have access to the records of the designated file.

• **READ**
  – Used to copy the contents of a record from an external magnetic medium into an input memory area.
Pseudolanguage Instructions

• **WRITE**
  – Used to copy the contents of an output memory area to a file stored on an external magnetic medium or a file produced on a printer.

• **CLOSE**
  – Terminates a program’s access to the records of a file previously opened.
Pseudolanguage Instructions

- **IF**
  - Used to compare the contents of a memory area with those of another memory area or a constant in order to determine whether the condition is true or false.

- **PERFORM**
  - Instructs the computer to perform a specified procedure.
Pseudolanguage Instructions

• MOVE
  – Used to tell the computer that the data contained in one area in memory is to be copied to another area in memory.

• COMPUTE
  – Used to indicate that a mathematical computation is to take place.
Pseudolanguage Instructions

• DISPLAY
  – Used to display a literal or the contents of an identifier on the screen.

• ACCEPT
  – Allows data entered through the keyboard (or any input device) to be stored in a memory area.
Pseudolanguage Instructions

• DECLARE
  – Used to define the various identifiers to be used in a program.

• ENTRY
  – Marks the beginning of a procedure.

• EXIT
  – Marks the end of a procedure.
Pseudolanguage Instructions

- **START**
  - Marks the beginning of a program’s control procedure.

- **STOP**
  - Causes the program execution to terminate.
Flowchart Example

MAIN-ROUTINE

OPEN INPUT ADDRESS-FILE
OUTPUT MAIL-LABELS-FILE

READ ADDRESS-FILE

EOF

PRINT-ADDRESS

CLOSE ADDRESS-FILE
MAIL-LABELS-FILE

STOP
BEGIN MAIN-ROUTINE
  OPEN INPUT ADDRESS-FILE
  OUTPUT MAIL-LABELS-FILE
  READ ADDRESS-FILE
  PERFORM UNTIL EOF="T"
    PERFORM
      PRINT-ADDRESS
      END PERFORM
  READ ADDRESS-FILE
  END PERFORM
  CLOSE ADDRESS-FILE
  OUTPUT MAIL-LABELS-FILE
END MAIN-ROUTINE
Problem Solving & Solution Design Concepts

• A knowledge of problem solving & programming concepts is a necessity for those who work w/ computers: either as programmers or as users.

• Problem solving steps (specifically algorithm definition) is similar in all programming languages.
Six (6) Steps in Problem Solving:

1. Identify the problem
2. Understand the problem
3. Identify alternative ways to solve the problem
4. Select the best way to solve the problem from the alternatives
5. List instructions that enable you to solve the problem by the selected method
6. Evaluate the solution
Seven (7) Basic Elements of Programming

1. Data: constants, variables
2. Input: reading of values from input devices (keyboard, I/O disk drives)
3. Output: writing of information to any output device (screen, disk drive, printer)
4. Operations: comparing values, assigning values, combining values
Seven (7) Basic Elements of Programming

5. Conditions / Selections: If-Then-Else, Case, Switches
6. Loops / Iterations: While-Do, Repeat-Until, For-Do
7. Subroutines / Modules: functions, procedures
Problems that can be solved on computers:

1. Computational
   - Problems involving some kind of mathematical processing

2. Logical
   - Involve relational or logical processing

3. Repetitive
   - Involve repeating a set of mathematical and/or logical instructions
Attributes of a Well-designed Program

- Program logic was mapped out using a planning tool
- Top-down approach
- Modular
- Structured
Logical Control Structures

- Elementary building blocks of structured programs
- Statements that control the order in which other program statements are executed
- Refer to the different ways in which program instructions may be executed
The Four (4) Logical Control Structures

1. Sequence
2. Selection / Decision
3. Iteration / Loop
4. Case
SEQUENCE

- Instructions are executed in the order in which they appear
- “Step-by-step” execution of instructions

*General Representation:*
SELECTION / DECISION

• A logical control structure that execute instructions depending on the existence of a condition

• Sometimes called an “If-Then-Else” logical control structure

*General Representation*:

```
  Condition met?
  F    T
    Instruction 1    Instruction 2
```

Flowchart diagram:

- Diamond shape indicating a decision point: 
  - Label: Condition met?
  - Outcomes:
    - F (False) leads to Instruction 1
    - T (True) leads to Instruction 2

- Arrows connecting the decision point to the instructions:
  - F arrow points to Instruction 1
  - T arrow points to Instruction 2
ITERATION / LOOP

• A logical control structure indicating the repeated execution of a series of steps (or instructions).

*General Representation:*
CASE

- A logical control structure that is used when there are numerous paths to be followed depending on the content of a given variable.

**General Representation:**

```
  Variable
    ↓
   /   \
  Case 1 Case 2 Case 3 Case n
    |     |     |     |
   Action 1 Action 2 Action 3 Action n
```

- Variable
  - Case 1
    - Action 1
  - Case 2
    - Action 2
  - Case 3
    - Action 3
  - Case n
    - Action n
Example: Program Development Flow

START

PROGRAM SPECIFICATIONS

ALGORITHM

CODING

TRANSLATION

SYNTAX ERROR?

DEBUGGING

RUN PROGRAM

LOGIC ERROR?

STOP

A

B

Y

N

A

B
DATA

• Data are *unorganized facts*.
• They go into a computer as input and are processed by the program.
• What is returned to the user is output, or *information*. 
DATA

• Constant
  – A value that never changes during the processing of all the instructions in a solution.

• Variable
  – The value of a variable does change during processing.
  – Also called as “identifier”
CONSTANT

• Can be any type of data: numerical, alphanumerical (or character), or special symbol

• Two (2) kinds of constants
  – **Literal**: refers to the actual value itself (e.g. 3.1416, “pshs”)
  – **Named**: uses a name or alias to represent an actual or literal value (e.g. PI, school_name)
VARIABLE

- Can be categorized by the kind of data it can hold.
- They must hold data that are of the same type, otherwise a mismatch error will occur.
- Can be any type of data: numerical, alphanumerical (or character), logical, or special symbol.
Rules for naming identifiers:

- Can be any string composed of letters, digits, and underscore ( _ ). No special characters.
- Should begin with either a letter or an underscore
- No blank spaces in variable names
- Should not be more than 128 characters
- Must be descriptive
Example: Constants & variables on the computer

- Constants
  - 8935084, -1.5, 3.1416, “pshs”, “*”

- Variables
DATA TYPES

• Numeric
• Character
• Logical
• Date / Time
Numerical Data

• Include all types of numbers (i.e., integers, non-integers)
• The only data type that can be used in calculations
• Subtypes:
  – **Integer**: negative numbers & whole numbers
  – **Real**: decimal numbers
  – **Float**: numbers in exponential / scientific form
Numerical Data Data Set

**Data Set**

- The set of symbols necessary to specify a datum as a particular data type.

- Data set for the numerical data type
  - All base 10 numbers
  - Positive (+) and the negative (-) sign
Character Data

• Consists of all numbers, letters, and special characters available to the computer (#, &, *, +, -, 0-9, A-Z, a-z) and placed within quotation marks.

• Cannot be used for calculations even if they consist of only numbers.

• **String**: means a string of characters

• **Concatenation**: means joining of two or more pieces of character or string data
Logical Data

• Consist of two pieces of data in the data set – the words TRUE and FALSE.
• Logical data are used in making a yes or no decision.
OPERATORS

• Are the data connectors within expressions and equations.
• They tell the computer how to process the data.
• They also tell the computer what type of processing needs to be done (i.e., mathematical, relational, or logical).
Types of operators used in calculations & problem solving:

1. Mathematical
2. Relational
3. Logical
Operands

• Are the data an operator connects and processes.

Resultant

• The answer that results when the operation is completed.
Mathematical Operators

• Include the following:
  – Addition +
  – Subtraction -
  – Multiplication *
  – Division /
  – Integer Division \ 
  – Modulo Division MOD
  – Powers ^ or **
  – Functions FunctionName (parameters)
Relational Operators

• Include the following:
  – Equal to =
  – Less than <
  – Greater than >
  – Less than or equal to <=
  – Greater than or equal to >=
  – Not equal to < > or !=
Relational Operators

• A programmer uses relational operators to program decisions.
• The resultant of a relational operator is the logical data type **TRUE** or **FALSE**.
• Are also used to *control* repetitive instructions called loops.
Logical Operators

• Are used to connect relational expressions (decision-making expressions) & to perform operations on logical data.

• Logical operators include the following:
  – Not       NOT
  – And       AND
  – Or        OR
Hierarchy of Operations

• An order in which operations (mathematical, relational, & logical) take place.

• Parentheses >> Functions >> Mathematical Operators >> Relational Operators >> Logical Operators
Hierarchy of Operations

- To reorder the normal processing sequence the programmer uses parentheses.
- The processing of the operands (as directed by the operators) always starts with the innermost parentheses and works outward, & processes from left to right.
# Hierarchy of Operations

<table>
<thead>
<tr>
<th>Order of Operations</th>
<th>Operand Data Type</th>
<th>Resultant Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses ( )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Functions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Mathematical Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Operator</th>
<th>Data Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.</td>
<td>Power</td>
<td>Numerical</td>
<td>Numerical</td>
</tr>
<tr>
<td>3.</td>
<td>, MOD</td>
<td>Numerical</td>
<td>Numerical</td>
</tr>
<tr>
<td>4.</td>
<td>*, /</td>
<td>Numerical</td>
<td>Numerical</td>
</tr>
<tr>
<td>5.</td>
<td>+, -</td>
<td>Numerical</td>
<td>Numerical</td>
</tr>
</tbody>
</table>

## Relational Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Operator</th>
<th>Data Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>=, &lt;, &gt;, &lt;=, &gt;=, &lt;&gt;</td>
<td>Numerical, Character</td>
<td>Logical</td>
</tr>
</tbody>
</table>

## Logical Operators

<table>
<thead>
<tr>
<th>Order</th>
<th>Operator</th>
<th>Data Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</td>
<td>NOT</td>
<td>Logical</td>
<td>Logical</td>
</tr>
<tr>
<td>8.</td>
<td>AND</td>
<td>Logical</td>
<td>Logical</td>
</tr>
<tr>
<td>9.</td>
<td>OR</td>
<td>Logical</td>
<td>Logical</td>
</tr>
</tbody>
</table>
Expressions & Equations

• An Expression processes data (the operands) through the use of operators.
• An equation stores the resultant of an expression in a memory location in the computer through the equal sign ( = ).
• Equations are often called “assignment statements.”
• The equal sign does not mean equality, but means “replaced by” or “is assigned the value of.”
• The right-hand side of the equation is processed before the assignment is made.
### Expressions & Equations

<table>
<thead>
<tr>
<th>Expressions</th>
<th>Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A + B</strong></td>
<td><strong>C = A + B</strong></td>
</tr>
<tr>
<td>A and B are numerical; the resultant is numerical and not stored</td>
<td>C, A, and B are numerical; the resultant is stored in C</td>
</tr>
<tr>
<td><strong>A &lt; B</strong></td>
<td><strong>C = A &lt; B</strong></td>
</tr>
<tr>
<td>A and B are numerical or character; the resultant is logical and is not stored</td>
<td>A and B are numerical or character; the resultant is stored in C; C is logical</td>
</tr>
<tr>
<td><strong>A OR B</strong></td>
<td><strong>C = A OR B</strong></td>
</tr>
<tr>
<td>A and B are logical; the resultant is logical and is not stored</td>
<td>C, A, and B are logical; the resultant is stored in C</td>
</tr>
</tbody>
</table>