IT 0104-PROGRAM DESIGN AND DEVELOPMENT

PURPOSE

To provide basic knowledge for solving problems using computers and to impart the necessary skills for the development of applications.

INSTRUCTIONAL OBJECTIVES

• To understand the basic concepts of problem solving using computers and to learn different problem solving strategies
• To discuss the importance of algorithms in the problem solving process
• To identify the necessary properties of good algorithms
• To use pseudo-code to implement, test, and debug algorithms for solving simple problems

R.LAKSHMI & R.SARUMATHI
Unit I-Problem Solving Concepts

POINTS TO BE DISCUSSED:

• What is a problem?-Types of problems
• Problem Solving in everyday life.
• Six steps for general problem solving
• Problem solving concepts for computers- Constants, Variables, Operators, Hierarchy of operations, Data types, Equations, Functions, Expressions.
• Organising Problems- Problem Analysis Charts, Structure/Interactivity Charts, IPO Chart, Algorithm, Flowcharts, Internal and External documentation
What is a PROBLEM

- A state of difficulty that needs to be resolved
- PROBLEMS EXIST WHERE GOALS NEED TO BE ATTAINED AND THERE IS UNCERTAINTY ABOUT SOLUTION
Problem Faced in Everyday in Life

• People make decisions everyday

• **Examples:**
  - Should I wear casual or formal today?
  - Should I watch TV or go out to cinema?
  - What career?
  - What course?
  - What shoes?
  - Everything needs a **DECISION AS A SOLUTION TO THE PROBLEM**
What happens when bad decisions are made?

- WASTAGE OF TIME AND RESOURCES

R.LAKSHMI & R.SARUMATHI
Six steps to ensure a Best decision in PROBLEM SOLVING

• Identify the problem
• Understand the problem
• Identify alternative ways to solve the problem
• List instructions that enable you to solve the problem using selected solution
• Select the best way to solve the problem from the list of alternative solutions
• Evaluate the solution
What makes a good decision?

- Well identified problem
- All alternatives considered
- Information overloaded – appropriate alternatives
- Can the person carry out steps/instructions
Important definitions

Solutions that can be solved with a series of known actions are called

*Algorithmic Solutions*

Employping a self-learning approach to the solution of a problem is known as

*Heuristic Solutions*

R.LAKSHMI & R.SARUMATHI
Examples

Algorithmic solution:

- To make a cup of coffee
- To find largest of three numbers

Heuristic solutions:

- how to buy the best stock?
- How to play chess?
Problem solving with computers

Computers use algorithmic solutions

- **Program** – set of instructions that make up solution to a problem
- **Results** – outcome of running the program
- **Testing** – Are the outcomes what you expected and correct
- **Documentation** – two types
  - manual documentation – instructions telling users how to use the program
Problem solving with computers involves several steps

Clearly define the problem.

• Analyse the problem and formulate a method to solve it (see also validation.).
• Describe the solution in the form of an algorithm.
• Draw a flowchart of the algorithm.
• Write the computer program.
• Compile and run the program (debugging).
• Test the program (debugging) (see also verification.).
• Interpretation of results.
Problem solving concepts for computers

- Constants
- Variables
- Operators
- Hierarchy of operations
- Data types
- Equations
- Functions
- Expressions

R.LAKSHMI & R.SARUMATHI
**Constant:**

- A variable whose value is determined when a program description is written and doesn’t change from that value during program execution.
- A constant is a value in a program, that stays the same throughout the program's execution-numeric, alphabetical, special symbols. Ex: \( \pi = 3.142857 \)

**Rules:**

Constant cannot be changed after it is initially given a value
**Variable**: a named location in memory for storing data.

- Variables are values that can change as much as needed during the execution of a program. Ex: `city=“chennai”`
### Rules:

<table>
<thead>
<tr>
<th>Names...</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANNOT start with a number</td>
<td>2i</td>
</tr>
<tr>
<td>CAN contain a number elsewhere</td>
<td>h2o</td>
</tr>
<tr>
<td>CANNOT contain any arithmetic operators...</td>
<td>r*s+t</td>
</tr>
<tr>
<td>CANNOT contain any other punctuation marks...</td>
<td>#@x%£!!a</td>
</tr>
<tr>
<td>CAN contain or begin with an underscore</td>
<td><em>height</em></td>
</tr>
<tr>
<td>CANNOT be a C keyword</td>
<td>struct</td>
</tr>
<tr>
<td>CANNOT contain a space</td>
<td>im stupid</td>
</tr>
<tr>
<td>CAN be of mixed cases</td>
<td>XSquared</td>
</tr>
</tbody>
</table>

---

*R.Lakshmi & R.Sarumathi*
Data vs. Information

Data: Unorganized facts

Information: Processed meaningful report
DATATYPE

- A type is the “kind” of data that variable is allowed to hold.
## Data types and their data sets

<table>
<thead>
<tr>
<th>Data type</th>
<th>Dataset</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric:integer</td>
<td>All whole numbers</td>
<td>8765,-98</td>
</tr>
<tr>
<td>Numeric:Real</td>
<td>All real numbers (whole + decimal)</td>
<td>3786.98,0.0087</td>
</tr>
<tr>
<td>Character(surrounded by quotation marks)</td>
<td>All letters, numerals and special symbols</td>
<td>“a”, ”A”, “=”,”5”, “$”</td>
</tr>
<tr>
<td>String(Surr. By quote marks)</td>
<td>Combinations of more than one character</td>
<td>“art”,”5678”,”01-345-456778”</td>
</tr>
<tr>
<td>Logical</td>
<td>True/false</td>
<td>True, False</td>
</tr>
</tbody>
</table>
Functions

• Small sets of instructions that perform specific tasks and return values
• Specify the basic tasks that are used repeatedly in the problem-solving process thus reducing time and improve the readability of the solution.
• Function name(data)
• Data used by the function are called parameters
• Ex: sqrt(n), max(n1,n2,n3)
• Divided into classes:
TYPES OF FUNCTIONS

- Mathematical function: sqrt(n), abs(n), sign(n)
- String function: Left(S,n), right(S,n), Length(S)
- Conversion function: String(N), Value(S)
- Statistical function: Average(list), Max(list), Sum(list)
- Utility function: Date, Time, Error
Operators

- Operators are symbols that indicate some kind of action is to be performed.
- Data connectors within expressions and equations
- Tell the computer how to process the data
- What type of processing needs to be done
- OPERANDS- data that the operator connects and processes
- RESULTANT-answer of the operation.
<table>
<thead>
<tr>
<th>Operation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subtraction</td>
<td>3 – 2 = 1</td>
</tr>
<tr>
<td>Addition</td>
<td>3 + 2 = 5</td>
</tr>
<tr>
<td>Multiplication</td>
<td>3 * 2 = 6</td>
</tr>
<tr>
<td>Division</td>
<td>6 / 3 = 2</td>
</tr>
<tr>
<td>Modulus</td>
<td>3 % 2 = 1</td>
</tr>
<tr>
<td>Decrement</td>
<td>-- 2 = 1</td>
</tr>
<tr>
<td>Increment</td>
<td>++ 2 = 3</td>
</tr>
<tr>
<td>Power ^</td>
<td>2 ^ 2 = 4</td>
</tr>
</tbody>
</table>
Relational Operators

- > Greater than
- < Less than
- >= Greater than or equal
- <= Less than or equal
- == Equal
- != Not Equal(<>)

R.LAKSHMI & R.SARUMATHI
Logical Operators

• **&& AND**

A && B: result is true iff A and B are true, false otherwise

• **|| OR**

A || B: result is false iff A and B are False, true otherwise

• **! NOT**

NOT A: True if A is False
Hierarchy of operations

1. Functions
2. Power
3. Mod
4. *, /
5. +, -
6. =, <, >, <=, >=, <>
7. NOT
8. AND
9. OR

R.LAKSHMI & R.SARUMATHI
**EXPRESSION:**

- An expression is a code segment that can be evaluated to produce a single value.
- Processes the data, the operands through the use of operators.
  
  \[ A + B \]

**EQUATIONS:**

- stores the result of an expression in a memory location in the computer through the = sign.
  
  \[ C = A + B \]
Evaluating Expressions

Calculate the area of the following figure

\[ \text{Area} = \frac{x}{2} \times b + x \times x \]
Evaluating Expressions

Area = x^2 + x/2 * b

1: 4^2 = 16
2: 4/2 = 2
3: 2*3 = 6
4: 16 + 6 = 22

x = 4 and b = 3

R.LAKSHMI & R.SARUMATHI
How computer stores data

• Each variable name is given a memory location which can hold one and only one value at a time.

• Internal memory is volatile

• The data, information and programs are stored in external medium as **FILES** – program files and data files.

• **BUFFER** - Temporary memory while transferring files from external to internal storage.
Communicating with computer

What is a program?

• A set of step-by-step instructions that directs the computer to perform tasks and produce results.

What is a Programming Language?

• A programming language is a set of rules that instructs a computer what operations to perform.

R.LAKSHMI & R.SARUMATHI
• **Syntax**: Rules governing the computer operating system, the language and the application

• **BUG**: It is an error

• **Debugging**: The process of locating and correcting an error.
Organising Problems

Certain organisational tools can help us to solve problems :-

- Problem Analysis Charts – a beginning analysis of the problem
- Structure/Interactivity Charts – shows the overall layout or structure of the solution
- IPO Chart – shows the input, the processing and the output
- Algorithm – show the sequence of instructions comprising the solution
- Flowcharts – graphic representations of the algorithms

R.LAKSHMI & R.SARUMATHI
**Problem Analysis Chart (PAC)**
separates the problem in 4 parts

<table>
<thead>
<tr>
<th>Given Data</th>
<th>Required Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 1: Data given in the problem or provided by user - Data, constants, variables</td>
<td>Section 2: Requirements to produce the output - information and format required</td>
</tr>
<tr>
<td>Processing required</td>
<td>Solution alternatives</td>
</tr>
<tr>
<td>Section 3: List of processing required – equations, or searching or sorting techniques</td>
<td>Section 4: List of ideas for the solution.</td>
</tr>
</tbody>
</table>

*R.LAKSHMI & R.SARUMATHI*
TASK :- 1) In Pairs create a Problem analysis chart for the average problem
2) Individually – create a Problem Analysis chart for calculating the Gross pay, given the formula GrossPay = Hours * PayRate

<table>
<thead>
<tr>
<th>Given Data</th>
<th>Required Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours</td>
<td>Gross Pay</td>
</tr>
<tr>
<td>Pay Rate</td>
<td></td>
</tr>
<tr>
<td>Processing required</td>
<td>Solution alternatives</td>
</tr>
<tr>
<td>GrossPay = Hours * PayRate</td>
<td>1. Define the hours worked And pay rate as constants 2. Define the hours worked and pay rate as input values</td>
</tr>
</tbody>
</table>
Interactivity (Structure) Chart

- This breakdowns a complex problem into simpler tasks.
- This divides your solution in *modules* – subtasks. Structure charts connect modules together to show the interaction of processing between the modules.
- Each module should contain the tasks to finish/accomplish one function - e.g calculating results, reading input.
- The control module controls the flow to other modules.

*R. Lakshmi & R. Sarumathi*
Interactivity (Structure) Chart

**KEY**

- Circle – iteration
- Diagonal line – same module used twice
- Diamond - selection

**TASK**

1) in pairs draw a structure chart for the Average problem
2) draw a structure chart for the Gross pay problem

*R.LAKSHMI & R.SARUMATHI*
Top-down method, here the program was executed form beginning to end. The user did not control the order of execution of the modules. The program was in control.
Interactivity chart for payroll problem (top-down method)
object-oriented programming – which is event driven – the user is in control. Modules are still used, within each module the execution is procedural.
Interactivity chart for payroll problem (Object-oriented solution)
The IPO Chart

- This extends and organises the information in the problem analysis chart.
- It shows in more detail what data items are input, what processing takes place on the data and what information will be the end result, the output.
- In the IPO chart the output is the first to be completed and then the input and then the processing.
### IPO chart

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Module Reference</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>All input data (from section 1 of the Problem Analysis chart)</td>
<td>All processing in steps (from Sections 3 and 4 of the problem analysis chart)</td>
<td>Module reference from the structure chart</td>
<td>All output requirements (from sections 1 and 2 of the problem analysis chart)</td>
</tr>
</tbody>
</table>
## IPO Chart

<table>
<thead>
<tr>
<th>Input</th>
<th>Processing</th>
<th>Module Reference</th>
<th>Output</th>
</tr>
</thead>
</table>
| **TASK :- 1)** Construct an IPO Chart for the Average Problem Enter 3 numbers | 1. Enter three numbers  
2. calculate the average of the three numbers  
3. print the average  
4. end | Read first number  
Read second number  
Read third number  
Calculate the average of the three numbers  
Print the average | Average |
| **TASK :- 2)** For the Gross Pay problem  
Hours worked  
Pay rate | Enter Hours worked  
1. Enter Pay rate  
2. Calculate Pay  
3. Print pay  
4. End | Read  
Read  
Calc  
Print  
PayrollControl | Grosspay |

*R.LAKSHMI & R.SARUMATHI*
Writing Algorithms

- After using the structure chart and the IPO chart the next step in organising the solution is to for the programmer to develop a set of instructions for the computer – called algorithms or Psuedocode.

  - **Algorithm:**
  - is a systematic procedure that produces - in a finite number of steps - the answer to a question or the solution of a problem.

  - is a sequence of instructions which can be used to solve a given problem

R.LAKSHMI & R.SARUMATHI
Computer operations in Pseudocode

1. Input Data

- Read
  - Read student name

- Get
  - Get student name
Computer operations in Pseudocode

2. Output Information

- Print
  - Print ‘Program Complete’
- Write
  - Write record to file
- Output
  - Output totalAmount
- Display
  - Display ‘Program Complete’
3. Perform Arithmetic

- Add
  - Add num1 to num2
- Subtract
  - Subtract num1 from num2
- Multiply
  - Multiply num1 by num2
- Divide
  - Divide num1 by num2
Computer operations in Pseudocode

4. Assign Values

- Initialise
  - Initialise totalPrice to zero
- Set
  - Set totalPrice to zero
- Store
  - Store zero in totalPrice
Computer operations in Pseudocode

5. Compare Values

- IF...THEN...ELSE
  - IF num1 > num2 THEN
    ADD num1 to total
  ELSE
    ADD num2 to total
  ENDIF
6. Repeat Actions

- **DOWHILE**
  
  - DOWHILE Num1 > Num2
    
    ADD num1 to total
    Multiply total by 3
    Subtract Num2 by 3
  
  ENDDO
sequence
Form of an Algorithm

Control Module
1. Instruction
2. Instruction
3. ...
4. ..
..---end

Name of Module (list of Parameters)
1. Instruction
2. Instruction
3. ..
4. ..
..------exit

Note: Uses End indicating end of processing

Note: Uses Exit bcos processing continues

R.LAKSHMI & R.SARUMATHI
Examples

• Addition
• Conversion from decimal to binary
• The process of boiling an egg
• The process of mailing a letter
• Sorting
• Searching

Let us write down the algorithm for a problem that is familiar to us.

Converting a decimal number into binary
Convert 75 to Binary

<table>
<thead>
<tr>
<th>2</th>
<th>75</th>
<th>remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>37</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1001011
Algorithm for Decimal-to-Binary Conversion

1. Write the decimal number

2. Divide by 2; write quotient and remainder

3. Repeat step 2 on the quotient; keep on repeating until the quotient becomes zero

4. Write all remainder digits in the reverse order (last remainder first) to form the final result
Points to Note

1. The process consists of repeated application of simple steps
2. All steps are unambiguous (clearly defined)
3. We are capable of doing all those steps
4. Only a limited no. of steps needs to be taken
5. Once all those steps are taken according to the prescribed sequence, the required result will be found
6. Moreover, the process will stop at that point
Algorithm (Better Definition)

1st Definition:

*Sequence of steps* that can be taken to solve a problem

Better Definition:

*A precise sequence of a limited number of unambiguous, executable steps* that *terminates* in the form of a *solution*

*R.LAKSHMI & R.SARUMATHI*
Three Requirements

1. **Sequence** is:
   a. Precise
   b. Consists of a limited number of steps

2. Each **step** is:
   a. Unambiguous
   b. Executable

3. The sequence of steps terminates in the form of a **solution**

*R.Lakshmi & R.Sarumathi*
Origin of the Term “Algorithm”

• The name derives from the title of a Latin book: Algoritmi de numero Indorum

• That book was a translation of an Arabic book: Al-Khwarizmi Concerning the Hindu Art of Reckoning

• That book was written by the famous 9-th century Muslim mathematician, Muhammad ibn Musa al-Khwarizmi
Pseudo Code

• Language that is typically used for writing algorithms

• Similar to a programming language, but not as rigid

• The method of expression most suitable for a given situation is used:
  
  – At times, plain English

  – At others, a programming language like syntax
Flowchart

- A graphical representation of a process (e.g. an algorithm), in which graphic objects are used to indicate the steps & decisions that are taken as the process moves along from start to finish.
- Individual steps are represented by boxes and other shapes on the flowchart, with arrows between those shapes indicating the order in which the steps are taken.
Flowchart Symbols

Start or stop
Process
Input or output
Decision
Flow line
Connector
Off-page connector
Process Module

Automatic-counter loop

Flowchart Symbols

R. LAKSHMI & R. SARUMATHI
A program that is easy to read & understand, and therefore, easy to maintain & enhance
correct program?

A program with correct syntax & semantics

R.LAKSHMI & R.SARUMATHI
Developing Short Programs

1. Read, understand the problem
2. Do you have all the required data?
   No: Get it
   Else assume it. State it explicitly
3. Do the design
4. Write test cases
5. Write the code on a piece of paper
6. Hand-check it
7. Type it in
8. Run & check it on test cases
9. Errors? fix & redo 9
   Done!

R.LAKSHMI & R.SARUMATHI
DOCUMENTATION

Internal Documentation:

- Remarks written with the instructions to explain what is being done in the program

External Documentation:

- Manuals/Help Menus written about the solution.