THOMSON

SIMPLE PROGRAM DESIGN

A STEP-BY-STEP APPROACH

LESLEY ANNE ROBERTSON

THOMSON

SIMPLE PROGRAM DESIGN

A STEP-BY-STEP APPROACH

LESLEY ANNE ROBERTSON

Chapter 1

Program design

Objectives

- To describe the steps in the program development process
- To introduce the current program design methodology
- To introduce procedural and object-oriented programming
- To introduce algorithms and pseudocode
- To describe program data

1.1

Steps in program development

Steps in program development

1. Define the problem into three separate components:
   - inputs
   - outputs
   - processing steps to produce required outputs.

Steps in program development

2. Outline the solution.
   - Decompose the problem to smaller steps.
   - Establish a solution outline.
   - Initial outline may include:
     - major processing steps involved
     - major subtasks
     - user interface
     - major control structures
     - major variable and record structures
     - mainline logic

Steps in program development

3. Develop the outline into an algorithm.
   - The solution outline is now expanded into an algorithm.
     - What is an algorithm? – a set of precise steps that describe exactly the tasks to be performed and the order in which they are to be carried out.
     - Pseudocode will be used to represent the solution algorithm

Steps in program development

4. Test the algorithm for correctness.
   - Very important in the development of a program, but often forgotten
   - Major logic errors can be detected and corrected at an early stage
   - Go through the algorithm step-by-step with test data to ensure the program will actually do what it is supposed to do.

Steps in program development

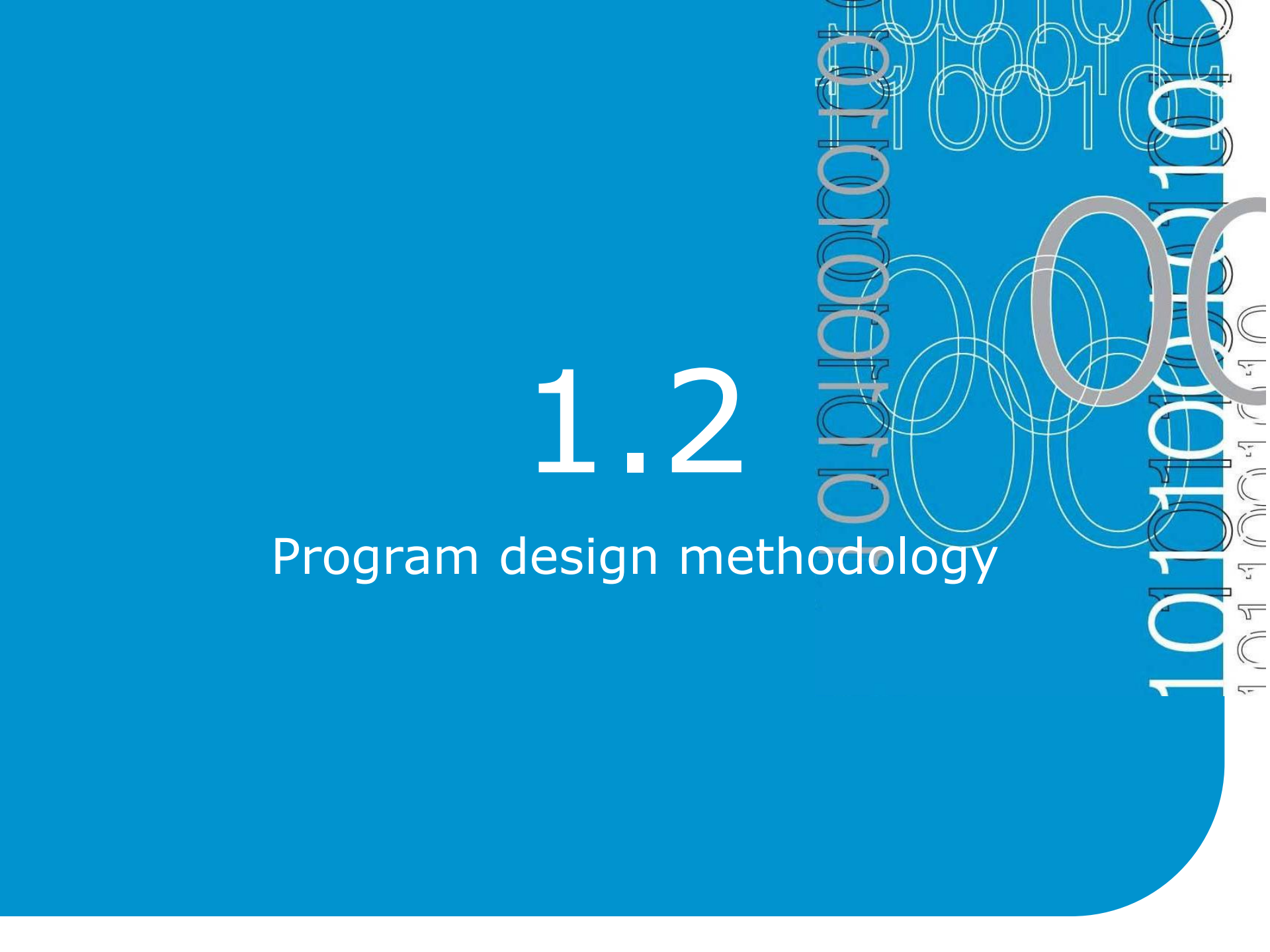
5. Code the algorithm into a specific programming language.
   - Start to code the program into a chosen programming language after all design considerations from Steps 1–4 are met.

Steps in program development

6. Run the program on the computer.
   - This step uses a program compiler and programmer-designed test data to machine-test the code for
     - syntax errors
     - logic errors

Steps in program development

7. Document and maintain the program.
   - Is really an ongoing task from the initial definition of the problem to the final test
   - Documentation involves:
     - external documentation
     - internal documentation



1.2

Program design methodology

Program design methodology

- Three approaches to program design include:
  - procedure-driven
  - event-driven
  - data-driven

Program design methodology

- Procedure-driven program design
  - Based on the idea that the most important feature of a program is what it does
  - Data into and out of each process is considered and a strategy is developed to break each function into smaller and more specific flows of data.

Program design methodology

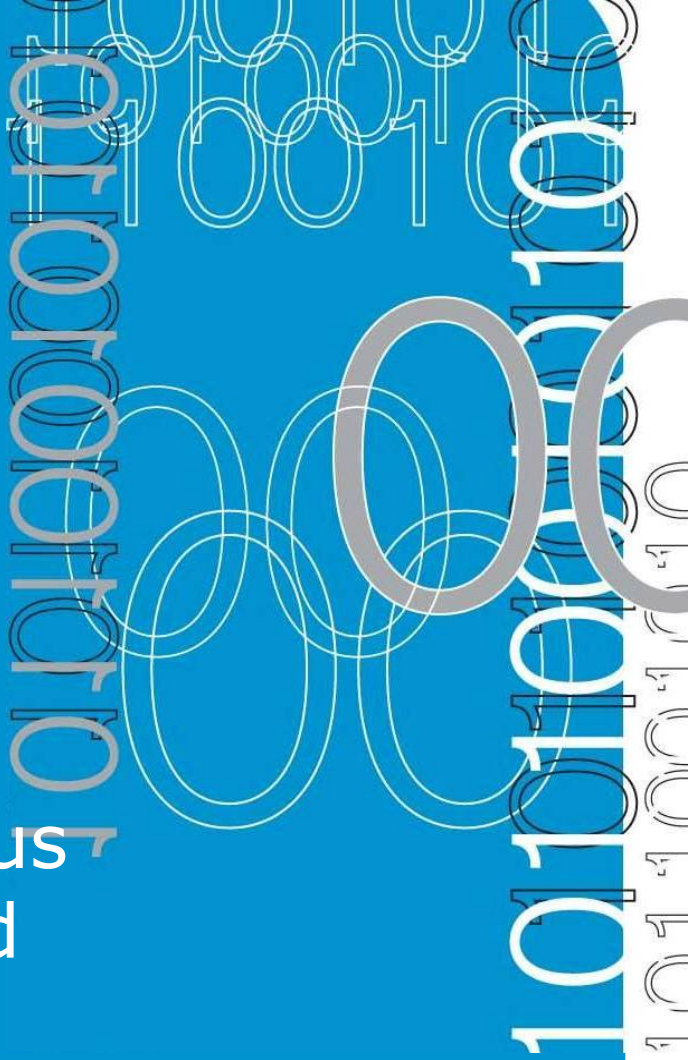
- Event-driven program design
  - Based on the idea that an event or interaction with the outside world can cause a program to change from one known state to another.

Program design methodology

- Data-driven program design
  - Based on the idea that the data in a program is more stable than the processes involved
  - Steps:
    - Analysis of data and relationships between the data
    - Required data outputs are examined in order to establish what processes are required to convert the input data to the required output

1.3

Procedural versus object-oriented programming



Procedural versus object-oriented programming

- Procedural programming approach concentrates on what a program has to do and involves identifying and organising the processes in the program solution. It is usually broken down into separate tasks, which include:
  - Top-down development
  - Modular design
  - Object-oriented programming

Procedural versus object-oriented programming

- Top-down development:
  - General solution to a problem is outlined
  - This is then broken down into more detailed steps until the most detailed levels have been completed
  - Finally, programmer starts to code
  - Results in a systematic approach to a program design

Procedural versus object-oriented programming

- Modular design:
  - Grouping task together
  - Connected directly to top-down development
  - Assists in the reading and understanding of the program

Procedural versus object-oriented programming

- Object-oriented programming
  - Based on breaking down the problem, but the primary focus is on the things that make up the program
  - Breaks the program into a set of separate objects that perform actions and relate to each other

1.4

An introduction to algorithms and pseudocode

An introduction to algorithms and pseudocode

- What is an algorithm?
  - Lists the steps involved in accomplishing a task (like a recipe)
  - Defined in programming terms as ‘a set of detailed and ordered instructions developed to describe the processes necessary to produce the desired output from a given input’

An introduction to algorithms and pseudocode

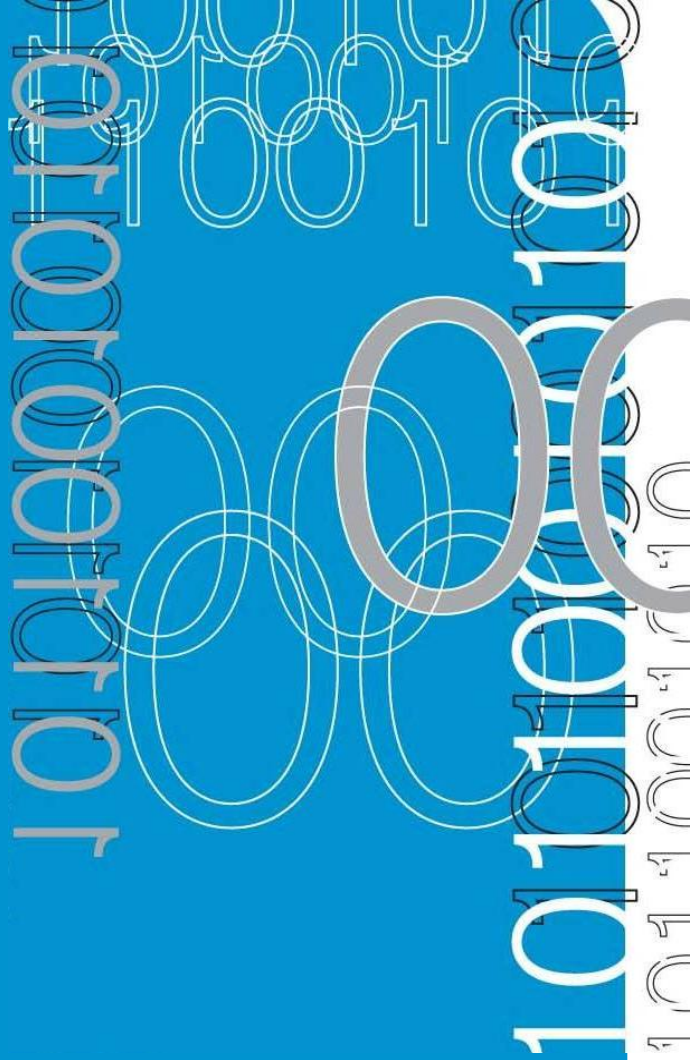
- What is an algorithm?
  - An algorithm must:
    - Be lucid, precise and unambiguous
    - Give the correct solution in all cases
    - Eventually end

An introduction to algorithms and pseudocode

- What is pseudocode?
  - Structured English (formalised and abbreviated to look like high-level computer language)

1.5

Program data



Program data

- Variable, constants and literals
  - A variable is a value stored in memory cells that may change or vary as the program executes.
  - A constant is a data item with a name and a value that remains the same during the execution of the program.
  - A literal is a constant whose name is the written representation of its value.

Program data

- Data types can be
  - Elementary data items
    - Contains a single variable that is always treated as a unit (classified into data types)

Program data

- Data types can be
  - Data structures
    - An aggregate of other data items. The data items that it contains are its components.
    - Data is grouped together in a particular way, which reflects the situation with which the program is concerned.
    - Most common are: record, file, array and string

Program data

- A popular method of storing information is to enter and store data on a file
- Advantages:
  - Different programs can access the same data
  - Data can be entered and reused several times
  - Data can be easily updated and maintained
  - The accuracy of the data is easier to enforce

Program data

- Data should always undergo a validation check before it is processed by a program.
- Examples:
  - Correct type
  - Correct range
  - Correct length
  - Completeness
  - Correct date

Summary

- Seven steps in program development are:
  1. Define the problem.
  2. Outline the solution.
  3. Develop the outline into an algorithm.
  4. Test the algorithm for correctness.
  5. Code the algorithm into a specific programming language.
  6. Run the program on the computer.
  7. Document and maintain the program.

Summary

- Three different program designs were introduced:
  - procedure-driven
  - event-driven
  - data-driven
- Definition of algorithm: a set of detailed, unambiguous and ordered instructions developed to describe the processes necessary to produce the desired output from the given input

Summary

- Definition of pseudocode: an English language-like way of representing the algorithm.
- Data variables, constants and literals were defined.
- Elementary data variables, constants and literals were defined.
- Elementary data items, data structures, files and data validation were introduced.

Objectives

- To introduce common words, keywords and meaningful names when writing pseudocode
- To define the three basic control structures as set out in the Structure Theorem
- To illustrate the three basic control structures using pseudocode

2.1

How to write pseudocode

How to write pseudocode

- There are six basic computer operations:
  1. A computer can receive information
  2. A computer can put out information
  3. A computer can perform arithmetic
  4. A computer can assign a value to a variable or memory location
  5. A computer can compare two variables and select one of two alternate actions
  6. A computer can repeat a group of actions

How to write pseudocode

1. A computer can receive information
   - The verbs Read and Get are used in pseudocode when a computer is required to receive information.
   - Read is used when the algorithm is to receive input from a record on a file.
   - Get is used when the algorithm is to receive input from the keyboard.

How to write pseudocode

2. A computer can put out information
   - The verbs Print, Write, Put, Output or Display are used in pseudocode when a computer is required to supply information or output to a device.
   - Print is used when the output is to be sent to a printer.
   - Put, Output or Display are used when the output is to be written to the screen.
   - Prompt and Get are also used in pseudocode to retrieve information.

How to write pseudocode

3. A computer can perform arithmetic
   - A mathematical calculation using either mathematical symbols or the words for those symbols. For example:
     - Add number to total OR
     - Total = Total + number
   - The following symbols can be written in pseudocode
     - **+ Add**
     - **- Subtract**
     - *** Multiply**
     - **/ Divide**
     - **() for Parentheses**

How to write pseudocode

- Orders of operation
  - Applies to pseudocode and to most computer languages
  - First operation carried out will be any calculations contained with parentheses

How to write pseudocode

4. A computer can assign a value to a variable or memory location
   - Three cases of writing pseudocode to assign a value to a variable:
     1. The verbs Initialise or Set are used to give data an initial value in pseudocode.
     2. Symbols '-' or '←' are used to assign a value as a result of some processing.
     3. The verbs Save or Store are used to keep a variable for later use.

How to write pseudocode

5. A computer can compare two variables and select one of two alternate actions.
   - To represent this operation in pseudocode, special keywords are used: IF, THEN and ELSE
   - The comparison of data is established in the IF clause
   - The choice of alternatives is determined by the THEN or ELSE options

How to write pseudocode

6. A computer can repeat a group of actions
   - When there is a sequence of processing steps that need to be repeated, two special keywords are used, DOWHILE and ENDDO
   - The condition for the repetition of a group of actions is established in the DOWHILE clause
   - The keyword ENDDO acts as a delimiter. As soon as the condition for the repetition is found false, control passes to the next statement after the ENDDO

2.2

Meaningful names

Meaningful names

- When designing an algorithm, a programmer must introduce some unique names which represents variables or objects in the problem.
- Names should be meaningful.
- Names should be transparent to adequately describe variables (Number1, number2, etc.).

Meaningful names

- Underscore is used when using more than one word (sales_tax or word_count).
- Most programming language does not tolerate a space in a variable as space would signal the end of a variable name.
- Another method is to use capital letters as a word separator (salesTax or wordCount).

2.3

The Structure Theorem

The Structure Theorem

- There are three basic control structures
  1. Sequence
  2. Selection
  3. Repetition

The Structure Theorem

1. Sequence
   - Straightforward execution of one processing step after another
   - Represents the first four basic computer operations
     1. Receive information
     2. Put out information
     3. Perform arithmetic
     4. Assign values

The Structure Theorem

- A typical sequence statement in an algorithm might read:

```
Add 1 to pageCount
Print heading line1
Print heading line2
Set lineCount to zero
Read customer record
```

- These instructions illustrate the sequence control structure as a straightforward list of steps written one after the other, in a top-to-bottom fashion

The Structure Theorem

2. Selection
   - Presentation of a condition and the choice between two actions, the choice depending on whether the condition is true or false
   - Represents the decision-making abilities of the computer
   - Illustrates the fifth basic computer operation – compare two variables and select one of two alternate actions

The Structure Theorem

- In pseudocode, selection is represented by the keywords IF, THEN, ELSE and ENDIF

```
IF condition p is true THEN
        statement(s) in true case
ELSE
        statement(s) in false case
ENDIF
```

- If condition p is true, then the statement in true case will be executed, and the statement in the false case will be skipped (vice versa)

The Structure Theorem

3. Repetition
   - Presentation of a set of instruction to be performed repeatedly, as long as the condition is true
   - Block statement is executed again and again until a terminating condition occurs
   - Illustrates the sixth basic computer operation – to repeat a group of actions.

The Structure Theorem

- Written in pseudocode as:

```
DOWHILE condition p is true
        statement block
ENDDO
```

- DOWHILE is a leading decision loop – condition is tested before any statements are executed
- ENDDO triggers a return of control to the retesting of the condition
- Condition is true, statements are repeated until condition is found false

Summary

- Six basic computer operations were listed:
  1. Receive information
  2. Put out information
  3. Perform arithmetic
  4. Assign a value to a variable
  5. Decide between two alternative actions
  6. Repeat a group of actions

Summary

- Structure theorem was introduced. The three basic control structures are:
  1. sequence
  2. selection
  3. repetition
- Each control structure was associated with the each of the six basic computer operations.

End of Lecture

Chapter 3

Developing an algorithm

Objectives

- To introduce methods of analysing a problem and developing a solution
- To develop simple algorithms using the sequence control structure
- To introduce methods of manually checking the developed solution

3.1

Defining the problem

Defining the problem

- First step in the development of a computer program is defining the problem
- Carefully reading and rereading the problem until it is completely understood
- Additional information will need to be sought to help resolve and deficiencies in the problem specification

Defining the problem

- Problem should be divided into three separate components:
  1. Input: a list of source data provided to the problem
  2. Output: a list of the outputs required
  3. Processing: a list of actions needed to produce the required outputs

Defining the problem

- When reading a problem statement, the input and output components are easily identified due to the use of descriptive words such as nouns and adjectives
- Processing component is also identified easily as the problem statement usually describes the processing steps as actions, using verbs and adverbs

Defining the problem

- Analyse the actual words used in the specification when dividing a problem into three separate components; divide them into those that are descriptive and those that imply action
- In some programming problems, the inputs, processes and output is not clearly defined. In such cases, it is best to concentrate on the outputs required

3.2

Designing a solution algorithm

Designing a solution algorithm

- Most challenging task in the life cycle of a program
- First attempt at designing a solution algorithm usually does not result in a finished product
- Pseudocode is useful in the trial-and-error process where it is relatively easy to add, alter or delete instructions

Designing a solution algorithm

- Reminder – if the algorithm is not correct, the program will never be correct
- It is important not to start coding until necessary steps of defining the problem and designing the solution algorithm have been completed

3.3

Checking the solution algorithm

Checking the solution algorithm

- After a solution algorithm has been established, it must be tested for correctness
- It is necessary because most major logic errors occur during the development of the algorithm (not detected then these errors can be passed on to the program)
- Easier to detect errors in pseudocode than in the corresponding program

Checking the solution algorithm

- Desk checking involves tracing through the logic of the algorithm with some chosen test data
- Walk through the logic of the algorithm exactly as a computer would, keeping track of all major variables values on a sheet of paper
- Helps detect errors early and allows the user to become familiar with the way the program runs

Checking the solution algorithm

- Selecting test data
  - Investigate program specification and choose simple test cases based on the requirement of the specification, not the algorithm
  - By doing this, the programmer will be able to concentrate on 'what' the program is supposed to do, not 'how'
  - To desk check the algorithm, only a few simple test cases that will follow the major parts of the algorithm logic is needed

Checking the solution algorithm

- Steps in desk checking an algorithm
  1. Choose simple input test cases that are valid
  2. Establish the expected result for each test case
  3. Make a table on a piece of paper of the relevant variable names within the algorithm
  4. Walk the first test case through the algorithm
  5. Repeat the walk-through process using other test data
  6. Check the expected result established in Step 2 matches the actual in Step 5

Summary

- A programmer must fully understand a problem before attempting to find a solution.
- The method suggested was to analyse the actual words used in the specification with the aim of dividing the problem into three separate components: input, output and processing.

Summary

- After initial analysis of the problem, the programmer must attempt to find a solution and express this solution as an algorithm.
- To do this, the programmer must use the defining diagram, the correct pseudocode statement and the three basic control structures.
- Check the algorithm for correctness by tracing through the algorithm step by step.

End of Lecture