Chapter 4

Selection control structures

Objectives

- To elaborate on the uses of simple selection, multiple selection and nested selection in algorithms
- To introduce the case construct in pseudocode
- To develop algorithms using variations of the selection control structure

4.1

The selection control structure

The selection control structure

- The condition in the IF statement is based on a comparison of two items and it is expressed with the following relational operations:
  - <   less than
  - >   greater than
  - =   equals to
  - <=   less than or equal to
  - >=   greater than on equal to
  - <>   not equal to

The selection control structure

- There are a number of variation of the selection structure as follows:
  1. Simple selection (simple IF statement)
  2. Simple selection with null false branch (null ELSE statement)
  3. Combined selection (combined IF statement)
  4. Nested selection (nested IF statement)

The selection control structure

1. Simple selection (simple IF statement)
   - Simple selection occurs when a choice is made between two alternate paths depending on the result of a condition being true or false
   - Keywords: IF, THEN, ELSE, ENDIF

The selection control structure

- Example:

```
IF account_balance < $300 THEN
  service_change = $5.00
ELSE
  service_charge = $2.00
ENDIF
```

The selection control structure

2. Simple selection with null false branch (null ELSE statement)
   - The null ELSE structure is a variation of the simple IF structure
   - It is used when a task is performed only when a particular condition is true
   - No processing will take place if condition is false

The selection control structure

- Example

```
IF student_attendance = part_time THEN
        add 1 to part_time_count
ENDIF
```

- In this case the part_time_count will be altered only if he student’s attendance pattern is part-time

The selection control structure

3. Combined selection (combined IF statement)
   - IF statement with AND or OR connector
   - Example IF, AND connector

```
IF student_attendance = part_time
AND student_gender = female THEN
     add 1 to female_part_time_count
ENDIF
```

   - This example, student record will undergo two test. Only those students who are female and who attend part-time will be selected; counter go up
   - If either condition is to be found to be false, the counter will not change

The selection control structure

- Example IF, OR connector

```
IF student_attendance = part_time
OR student_gender = female THEN
        add 1 to female_part_time_count
ENDIF
```

- Counter will only increase if
  - The student is part-time regardless of gender
  - The student is female regardless of attendance pattern

The selection control structure

- More than two condition can be linked together with the AND or OR operators.

```
IF (record_code = '23'
OR update_code = delete)
AND account_balance = zero THEN
  delete customer record
ENDIF
```

- Remark → parentheses must be used to avoid ambiguity

The selection control structure

- The NOT operator can be used for the logical negation of a condition

```
IF NOT (record_code = '23') THEN
  update customer record
ENDIF
```

- Note that the AND and OR operators can also be used with the NOT operator, but great care must be taken and parentheses used to avoid ambiguity

The selection control structure

4. Nested selection (nested if statement)
   - Can be classified as
     - Linear nested IF statements
     - Non-linear nested IF statements
   - Linear nested IF statement is used when a field is being tested for various values and a different action is to be taken for each value

The selection control structure

- It is called Linear due to each ELSE immediately follows the IF condition to which it corresponds

```
IF record_code = 'A' THEN
  increment counter_A
ELSE
  IF record_code = 'B' THEN
           increment counter_B
  ELSE
           increment error_counter
  ENDIF
ENDIF
```

- Note there are an equal number of IF, ELSE and ENDIF and indentation makes it easier to read and understand

The selection control structure

- Non-linear nested IF occurs when a number of different conditions needs to be satisfied before a particular action can occur
- It is termed non-linear because the ELSE statement may be separated from the IF statement with which it is paired

The selection control structure

```
IF student_attendance = part_time THEN
    IF student_gender = female THEN
      IF student_age >21 THEN
                  add 1 to mature_female_pt_students
      ELSE
                  add 1 to young_female_pt_students
      ENDIF
    ELSE
      add 1 to male_pt_students
    ENDIF
ELSE
    add 1 to full_time_students
ENDIF
```

– Note: Equal number of IF, ELSE and ENDIF

4.2

Algorithms using selection

Algorithm using selection

- Example that use the selection control structure. The problem will be defined, solution algorithm will be developed and the algorithm will be manually tested
- Example: Design an algorithm that will <u>prompt</u> a terminal operator for three characters, <u>accept</u> those characters as input, <u>sor</u>t them into ascending sequence and <u>output</u> them onto screen

Algorithm using selection

A. Defining the diagram

Input	**Processing**	**Output**
char_1 char_2 char_3	Prompt for characters Accept 3 characters Sort three characters Output three characters	char_1 char_2 char_3

Algorithm using selection

B. Solution algorithm

- Solution algorithm requires a series of IF statements to sort the three characters into ascending sequence

```
        Read_three_characters
1              Prompt the operator for char_1, char_2, char_3
2              Get char_1, char_2, char_3
3              IF char_1 > char_2 THEN
                      temp = char_1
                      char_1 = char_2
                      char_2 = temp
               ENDIF
4              IF char_1 > char_3 THEN
                      temp = char_2
                      char_2 = char_3
                      char_3 = temp
               ENDIF
5              IF char_1 > char_2 THEN
                      temp = char_1
                      char_1 = char_2
                      char_2 = temp
               ENDIF
6              Output to the screen char_1, char_2, char_3
        END
```

Algorithm using selection

- Logic of the algorithm is concerned with the sorting of the three characters into alphabetic sequence

C. Desk checking

- Set of valid characters will be used to check the algorithm; the characters k, b and g will be used

Algorithm using selection

- Input data

	Data Set
char_1	k
char_2	b
char_3	g

- Expected results

	Data Set
char_1	b
char_2	g
char_3	k

Algorithm using selection

- Desk check table

Statement no	char_1	char_2	char_3	temp
First pass				
1,2	k	b	g	
3	b	k		k
4		g	k	k
5				
6	output	output	Output	

4.3

The case structure

The case structure

- Case control structure is another means of expressing linear nested IF statements in a simpler and more concise form.
- Used in pseudocode for two reasons:
  - It can be directly translated into many high-level languages.
  - It makes the pseudocode easier to write and understand.

The case structure

- The linear nested IF structure can be replaced with a case control structure
- This simplifies the basic selection control structure and extends it from a choice between two values to a choice of multiple values
- In pseudocode, the keywords CASE OF and ENDCASE serve to identify the structure with the multiple values indented

Summary

- Description and pseudocode examples were given for simple selection, null ELSE, combined IF and nested IF statements.
- Case structure was introduced as a means of expressing a linear nested IF statement in a simpler and more concise form.

Chapter 5

Repetition control structures

Objectives

- To develop algorithm that use the DOWHILE and REPEAT… UNTIL control structures
- To introduce a pseudocode structure for counted repetition loops
- To develop algorithms using variations of the repetition construct

5.1

Repetition using the DOWHILE structure

Repetition using the DOWHILE structure

- Most programs require the same logic to be repeated for several sets of data
- The most efficient way to deal with this situation is to establish a looping structure in the algorithm that will cause the processing logic to be repeated a number of times

Repetition using the DOWHILE structure

- Three different ways that a set of instructions can be repeated:
  - At the beginning of the loop (leading decision loop)
  - At the end of the loop (trailing decision loop)
  - A counted number of times (counted loop)

Repetition using the DOWHILE structure

- Leading decision loop
  - DOWHILE construct is a leading decision loop – that is the condition is tested before any statements are executed.
  - Format is:

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

Repetition using the DOWHILE structure

– The following processing takes place:
  1. Logical condition p is tested
  2. If condition p is found to be true, the statements within the statement block are executed once. The delimiter ENDDO then triggers a return of control to the retesting of condition p
  3. If condition p is still true, the statements are executed again, and so the repetition process continues until the condition is found to be false
  4. If condition p is found to be false, no further processing takes place within this loop

Repetition using the DOWHILE structure

- There are two important consideration to be aware of before designing a DOWHILE loop
  - Testing of the condition is at the beginning of the loop
  - The only way to terminate the loop is to render the DOWHILE condition false

Repetition using the DOWHILE structure

- Using DOWHILE to repeat a set of instructions a known number of times
  - When a set of instruction is repeated a specific number of times, a counter can be used in pseudocode, which is initialised before the DOWHILE statement and incremented just before the ENDDO statement

Repetition using the DOWHILE structure

- Using DOWHILE to repeat a set of instruction an unknown number of times
  - When a trailer record or sentinel exists
    - Special record or value placed at the end of valid data to signify the end of that data
  - When a trailer record or sentinel does not exist
    - End-of-file (EOF) marker is added when the file is created as the last character in the file

5.2

Repetition using the REPEAT...UNTIL structure

Repetition using the REPEAT...UNTIL structure

- Trailing decision loop
  - REPEAT...UNTIL structure is similar to the DOWHILE structure (group of statements are repeated in accordance with specific condition)
  - REPEAT...UNTIL structure tests condition at the end of the loop
  - This means that the statement within the loop will be executed once before the condition is tested

Repetition using the REPEAT…UNTIL structure

- If the condition is false, the statement will then be repeated UNTIL the condition becomes true
- REPEAT…UNTIL is a trailing decision loop (the statement are executed once before the condition is tested)
- Two consideration to be aware of when using REPEAT…UNTIL
  - Loops are executed when the condition is false
  - This structure will always be executed at least once

5.3

Counted repetition

Counted repetition

- Counted loop
  - Counted repetition occurs only once when the exact number of loop iterations is known in advance
  - Simple keyword DO is use as follows:

```
DO loop_index = initial_value to
final_value
    statement block
ENDDO
```

Counted repetition

- The DO loop does more than repeat the statement block.
  1. Initialise the loop_index to the required intial_value
  2. Increment the loop_index by 1 for each pass through the loop
  3. Test the value of loop_index at the beginning of each loop to ensure that it is within the stated range of values
  4. Terminate the loop when the loop_index has exceeded the specified final_value

Counted repetition

- In other words, a counted repetition construct will perform the initialising, incrementing and testing of the loop counter automatically
- It will also terminate the loop once the require number of repetition has been executed

Summary

- This chapter covered the repetition control structure in detail
- Description and pseudocode were given for:
  - Leading decision loops (DOWHILLE)
  - Trailing decision loops (REPEAT…UNTIL)
  - Counted loops (DO)

Summary

- Most of the solution algorithms that used the DOWHILE structure had the same general pattern. This pattern consisted of:
  1. some initial processing before the loop
  2. some processing for each record within the loop
  3. some final processing once the loop has been exited.

Summary

- Expressed as a solution algorithm, this basic pattern was developed as a general solution:

```
Process_sequential_file
        Initial processing
        Read first record
        DOWHILE more records exist
                Process this record
                Read next record
        ENDDO
        Final processing
END
```

Chapter 6

Pseudocode algorithms using sequence, selection and repetition

Objectives

- To develop solution algorithms to eight typical programming problems using sequence, selection and repetition constructs

6.1

Eight solution algorithms

Eight solution algorithms

- Each programming problem will be defined, the control structures required will be determined and a solution algorithm will be devised

Eight solution algorithms

1. Defining the problem
   – Divide the problem into its three components
     - Input
     - Output
     - Processing
   – The processing component should list the task to be performed – what need to be done, <u>NOT</u> how. Underline the verbs in each problem to help identify the actions to be performed

Eight solution algorithms

2. The control structures required
   - Once the problem has been defined, write down the control structures (sequence, selection and repetition) that may be needed, as well as any extra variables that the solution may require

Eight solution algorithms

3. The solution algorithm
   - Devise a solution algorithm and represent it using pseudocode
4. Desk checking
   - Desk check each of the algorithm with two or more test cases

Example 6.1 Process number pairs

Design an algorithm that will prompt for and receive pairs of numbers from an operator at a terminal and display their sum, product and average on the screen. If the calculated sum is over 200, an asterisk is to be displayed beside the sum. The program is to terminate when a pair of zero values is entered.

Example 6.1 Process number pairs

A. Defining Diagram

Input	Processing	Output
number1 number2	Prompt for numbers Get numbers Calculate sum Calculate product Calculate average Display sum, product, average Display '*'	Sum Product Average '*'

Example 6.1 Process number pairs

B. Control structures required
   1. A DOWHILE loop to control the repetition
   2. An IF statement to determine if an asterisk is to be displayed
   3. Note the use of the NOT operand with the AND logical operator

Example 6.1 Process number pairs

C. Solution algorithm

```
Process_numbers_pairs
          Set sum to zero
          Prompt for number1, number2
          Get number1, number2
          DOWHILE NOT (number1 = 0 AND number2 = 0)
                    sum = number1 + number2
                    product = number1 * number2
                    average = sum / 2
                    IF sum > 200 THEN
                              Display sum, '*', product, average
```

Example 6.1 Process number pairs

```
                ELSE
                        Display sum, product, average
                ENDIF
                Prompt for number1, number2
                Get number1, number2
        ENDDO
END
```

Summary

- This chapter developed solutions to eight programming problems. The approach to the problems followed the same pattern:
  1. The problem was defined using a defining diagram.
  2. The control structures required were written down, along with any extra variables required.
  3. The solution algorithm was produced, using pseudocode and the three basic control structures: sequence, selection and repetition.

Summary

- It was noted that the solution algorithms followed the same basic pattern, although the statements within the pattern were different

```
Process_sequential_file
        Initial processing
        Read first record
        DOWHILE more records exist
                Process this record
                Read next record
        ENDDO
        Final processing
END
```

End of Lecture