4.3: An introduction to Algorithms and Pseudocode

IT1406 - Introduction to Programming

Level I - Semester 1

4.3. An introduction to Algorithms and Pseudocode

- A program must be systematically and properly designed before coding begins.
- This design process results in the construction of an algorithm.

4.3. An introduction to Algorithms and Pseudocode

What is an algorithm?

- An algorithm is like a recipe: it lists the steps involved in accomplishing a
- task.
- It can be defined in programming terms as a set of detailed, unambiguous and ordered instructions developed to describe the processes necessary to produce the desired output from a given input.
- The algorithm is written in simple English and is not a formal document. However, to be useful, there are some principles that should be adhered to.
- An algorithm must:
    - be lucid, precise and unambiguous
    - give the correct solution in all cases
    - eventually end.

4.3. An introduction to Algorithms and Pseudocode

- **What is an algorithm? (cont.)**
- For example, if you want to instruct someone to add up a list of prices on a pocket calculator, you might write an algorithm such as the following:

```
Turn on calculator
Clear calculator
Repeat the following instructions
    Key in dollar amount
    Key in decimal point (.)
    Key in cents amount
    Press addition (+) key
Until all prices have been entered
Write down total price
Turn off calculator
```

4.3. An introduction to Algorithms and Pseudocode

What is an algorithm? (cont.)

- Notice that in this algorithm the first two steps are performed once, before the repetitive process of entering the prices.
- After all the prices have been entered and summed, the total price can be written down and the calculator turned off.
- These final two activities are also performed only once.
- This algorithm satisfies the desired list of properties: it lists all the steps in the correct order from top to bottom in a definite and unambiguous fashion until a correct solution is reached.
- Notice that the steps to be repeated (entering and summing the prices) are indented, both to separate them from those steps performed only once and to emphasize the repetitive nature of their action.
- It is important to use indentation when writing solution algorithms because it helps to differentiate between the different control structures.

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- In 4.1 described seven steps in the development of a computer program.
- The very first step, and one of the most important, is defining the problem.
- This involves carefully reading and rereading the problem until you understand completely what is required. Quite often, additional information will need to be sought to help resolve any ambiguities or deficiencies in the problem specifications.
- To help with this initial analysis, the problem should be divided into three separate components:

  1 *Input:* a list of the 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.

4.3. An introduction to Algorithms and Pseudocode

- **Defining the problem**
- When reading the problem statement, the input and output components are easily identified, because they use descriptive words such as nouns and adjectives.
- The processing component is also identified easily.
- The problem statement usually describes the processing steps as actions, using verbs and adverbs.
- When dividing a problem into its three different components, analyse the actual words used in the specification, and divide them into those that are descriptive and those that imply actions.
- It may help to underline the nouns, adjectives and verbs used in the specification.

4.3. An introduction to Algorithms and Pseudocode

- **Defining the problem**
- In some programming problems, the inputs, processes and outputs may not be clearly defined.
- In such cases, it is best to concentrate on the outputs required.
- Doing this will then determine the inputs, and the way will be set for determining the processing steps required to produce the desired output.
- At this stage, the processing section should be a list of what actions need to be performed, not how they will be accomplished.
- Do not attempt to find a solution until the problem has been completely defined.

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- Let's look at a simple example.

Add three numbers

- A program is required to read three numbers, add them together and print their total.
- Tackle this problem in two stages.
- First, underline the nouns and adjectives used in the specification.
- This will establish the input and output components, as well as any objects that are required.

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- With the nouns and adjectives underlined, our example would look like this:

A program is required to read <u>three numbers</u>, add them together and print their <u>total</u>.

- By looking at the underlined nouns and adjectives, it is easy to see that the input for this problem is three numbers and the output is the total.
- It is helpful to write down these first two components in a simple diagram, called a defining diagram as in the next slide.

4.3. An introduction to Algorithms and Pseudocode

- **Defining the problem**

Input	Processing	Output
number1		total
number2		
number3		

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- Second, underline (in a different **colour**) the verbs and adverbs used in the specification.
- This will establish the actions required.

A program is required to read <u>three numbers</u>, add them together and print their <u>total</u>.

- By looking at those words, it can be seen that the processing verbs are '**read**', '**add together**' and '**print**'.

4.3. An introduction to Algorithms and Pseudocode

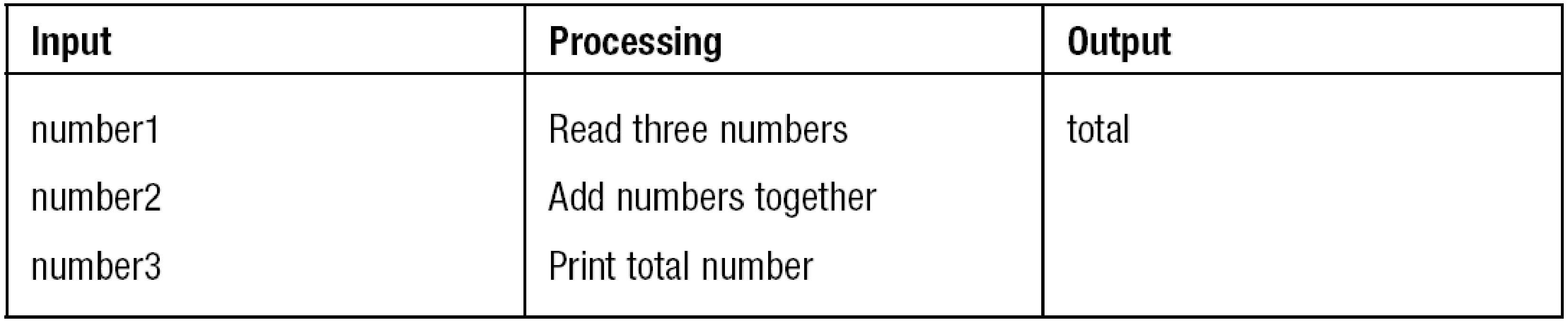
- **Defining the problem**

- By looking at the underlined words, it can be seen that the processing verbs are ‘read’, ‘add together’ and ‘print’.
- These steps can now be added to our defining diagram to make it complete.
- When writing down each processing verb, also include the objects or nouns associated with each verb.

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- The defining diagram now becomes:



Input	Processing	Output
number1	Read three numbers	total
number2	Add numbers together	
number3	Print total number	

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- Now that all the nouns and verbs in the specification have been considered and the defining diagram is complete, the problem has been properly defined.
- That is, we now understand the input to the problem, the output to be produced, and the processing steps required to convert the input to the output.

4.3. An introduction to Algorithms and Pseudocode

Defining the problem

- When it comes to writing down the processing steps in an algorithm, use words that describe the work to be done in terms of single specific tasks or functions. For example:

```
Read three numbers
add numbers together
Print total number
```

- There is a pattern in the words chosen to describe these steps.
- Each action is described as a single verb followed by a two-word object.
- Studies have shown that if you follow this convention to describe a processing step, two benefits will result.
- First, you are using a disciplined approach to defining the problem and, second, the processing is being divided into separate tasks or functions.
- This simple operation of dividing a problem into separate functions and choosing a proper name for each function will be extremely important later, when considering algorithm modules.

4.3. An introduction to Algorithms and Pseudocode

Designing a solution algorithm

- Designing a solution algorithm is the most challenging task in the life cycle of a program.
- Once the problem has been properly defined, you can start to outline your solution.
- The first attempt at designing a solution algorithm usually does not result in a finished product.
- Steps may be left out, or some that are included may later be altered or deleted.
- Pseudocode is useful in this trial-and-error process, since it is relatively easy to add, delete or alter an instruction.
- Do not hesitate to alter algorithms, or even to discard one and start again, if you are not completely satisfied with it.
- If the algorithm is not correct, the program will never be.

4.3. An introduction to Algorithms and Pseudocode

Designing a solution algorithm

- There is some argument that the work of a programmer ends with the algorithm design.
- After that, a coder or trainee programmer could take over and code the solution algorithm into a specific programming language.
- In practice, this usually does not happen. However, it is important that you do not start coding until the necessary steps of defining the problem and designing the solution algorithm have been completed.

4.3. An introduction to Algorithms and Pseudocode

Designing a solution algorithm

- Here are solution algorithms for the preceding two examples.
- All involve sequence control structures only; there are no decisions or loops, so the solution algorithms are relatively simple.

4.3. An introduction to Algorithms and Pseudocode

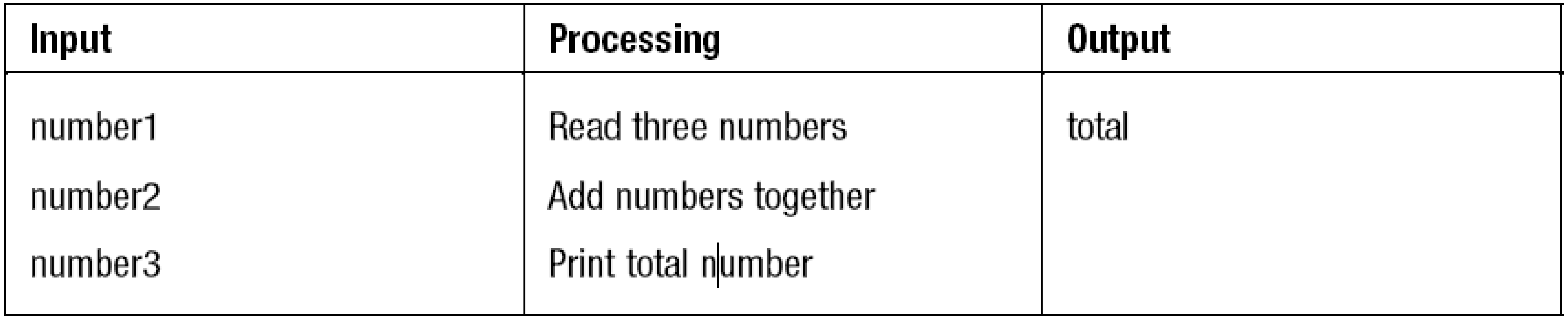
Solution algorithm for adding three numbers

- A program is required to read three numbers, add them together and print their total.

4.3. An introduction to Algorithms and Pseudocode

Solution algorithm for adding three numbers

- ***Defining diagram***



Input	Processing	Output
number1	Read three numbers	total
number2	Add numbers together	
number3	Print total number	

4.3. An introduction to Algorithms and Pseudocode

- **Solution algorithm for adding three numbers**
- ***Solution algorithm***
- The defining diagram shows what is required, and a simple calculation will establish how.
- Using pseudocode and the sequence control structure establish the solution algorithm as follows:

4.3. An introduction to Algorithms and Pseudocode

- **Solution algorithm for adding three numbers**
- ***Solution algorithm***

```
Add_three_numbers
    Read number1, number2, number3
    total = number1 + number2 + number3
    Print total
END
```

4.3. An introduction to Algorithms and Pseudocode

Solution algorithm for adding three numbers

There are a number of points to consider in this solution algorithm:

1. A name has been given to the algorithm, namely Add_three_numbers. Algorithm names should briefly describe the function of the algorithm, and are usually expressed as a single verb followed by a two-word object.

2. An END statement at the end of the algorithm indicates that the algorithm is complete.

3. All processing steps between the algorithm name and the END statement have been indented for readability.

4. Each processing step in the defining diagram relates directly to one or more statements in the algorithm.

For instance, 'Read three numbers' in the defining diagram becomes 'Read number1, number2, number3' in the algorithm; and 'Add numbers together' becomes 'total = number1 + number2 + number3'.

4.3. An introduction to Algorithms and Pseudocode

Solution algorithm for calculating average

- A program is required to prompt the terminal operator for the maximum and minimum temperature readings on a particular day, accept those readings as integers, and calculate and display to the screen the average temperature, calculated by:

(maximum temperature + minimum temperature)/2.

4.3. An introduction to Algorithms and Pseudocode

Defining Diagram

Input	Processing	Output
max_temp min_temp	Prompt for temperatures Get temperatures Calculate average temperature Display average temperature	avg_temp

4.3. An introduction to Algorithms and Pseudocode

Solution algorithm

- Using pseudocode, a simple calculation and the sequence control structure, the algorithm can be expressed as follows:

```
Find_average_temperature
    Prompt operator for max_temp, min_temp
    Get max_temp, min_temp
    avg_temp = (max_temp + min_temp)/2
    Output avg_temp to the screen
END
```

4.3. An introduction to Algorithms and Pseudocode

What is pseudocode?

- Pseudocode and flowcharts are both popular ways of representing algorithms.
- Pseudocode has been chosen as the primary method of representing an algorithm because it is easy to read and write, and allows the programmer to concentrate on the logic of the problem.
- Pseudocode is really structured English.
- It is English that has been formalised and abbreviated to look like the high-level computer languages.
- There is no standard pseudocode at present.
- Authors seem to adopt their own special techniques and sets of rules, which often resemble a particular programming language.
- Lets try to attempt to establish a standard pseudocode for use by all programmers, regardless of the programming language they choose.

4.3. An introduction to Algorithms and Pseudocode

Like many versions of pseudocode, this version has certain conventions:

1. Statements are written in simple English.

2. Each instruction is written on a separate line.

3. Keywords and indentation are used to signify particular control structures.

4. Each set of instructions is written from top to bottom, with only one entry and one exit.

5. Groups of statements may be formed into modules, and that module given a name.

4.3. An introduction to Algorithms and Pseudocode

What is pseudocode?

- When designing a solution algorithm, it is necessary to keep in mind the fact that a computer will eventually perform the set of instructions you write.
- If you use words and phrases in the pseudocode that correspond to some basic computer operations, the translation from the pseudocode algorithm to a specific programming language becomes quite simple.
- In here we establish six basic computer operations and introduces common words and keywords used to represent these operations in pseudocode.
- Each operation can be represented as a straightforward instruction in English, with keywords and indentation to signify a particular control structure.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

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 alternative actions**
6. **A computer can repeat a group of actions**

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

1. A computer can receive information

- When a computer is required to receive information or input from a particular source, whether it be a terminal, a disk or any other device, the verbs Read and Get are used in the pseudocode. Read is usually used when the algorithm is to receive input from a record on a file, while Get is used when the algorithm is to receive input from the keyboard.
- For example, typical pseudocode instructions to receive information are:
  - Read student name
  - Get system date
  - Read number_1, number_2
  - Get tax_code
- Each example uses a single verb, Read or Get, followed by one or more nouns to indicate what data is to be obtained.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

2. A computer can put out information

- When a computer is required to supply information or output to a device, the verbs Print, Write, Put, Output or Display are used in the pseudocode.
- Print is usually used when the output is to be sent to the printer, while Write is used when the output is to be written to a file.
- If the output is to be written to the screen, the words Put, Output or Display are used in the pseudocode. Typical pseudocode examples are:
  - Print ‘Program Completed’
  - Write customer record to master file
  - Put out name, address and postcode
  - Output total_tax
  - Display ‘End of data’

4.3. An introduction to Algorithms and Pseudocode

2. A computer can put out information

- Usually an output Prompt instruction is required before an input Get instruction. The Prompt verb causes a message to be sent to the screen, which requires the user to respond, usually by providing input. Examples are:
  - Prompt for student_mark
  - Get student_mark

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

3. A computer can perform arithmetic

- Most programs require the computer to perform some sort of mathematical calculation, or to apply a formula, and for these a programmer may use either actual mathematical symbols or the words for those symbols.
- For instance, the same pseudocode instruction can be expressed as either of the following:
  - add number to total
  - total = total + number
- Both expressions clearly instruct the computer to add one value to another, so either is acceptable in pseudocode.
- The equal symbol '=' is used to indicate assignment of a value as a result of some processing.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

3. A computer can perform arithmetic

- To be consistent with high-level programming languages, the following symbols can be written in pseudocode:

  + for add

  – for subtract

  * for multiply

  / for divide

  ( ) for parentheses

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

3. A computer can perform arithmetic

- The verbs Compute and Calculate are also available.
- Some examples of pseudocode instructions to perform a calculation are:
    - divide total_marks by student_count
    - sales_tax = cost_price * 0.10
    - compute C = (F – 32) * 5/9

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

3. A computer can perform arithmetic

- **Order of operations**
- When writing mathematical calculations for the computer, the standard mathematical *order of operations* applies to pseudocode and to most computer languages.
- The first operation carried out will be any calculation contained within parentheses.
- Next, any multiplication or division, as it occurs from left to right, will be performed.
- Then, any addition or subtraction, as it occurs from left to right, will be performed.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

4. A computer can assign a value to a variable or memory location

- There are three instances in which you may write pseudocode to assign a value to a variable or memory location:

1 To give data an initial value in pseudocode, the verbs Initialise or Set are used.

2 To assign a value as a result of some processing, the symbols '=' or '←' are written.

3 To keep a variable for later use, the verbs Save or Store are used.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

4. A computer can assign a value to a variable or memory location

- Some typical pseudocode examples are:

```
Initialise total_price to zero
Set student_count to 0
total_price = cost_price + sales_tax
total_price ← cost_price + sales_tax
store customer_num in last_customer_num
```

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

4. A computer can assign a value to a variable or memory location

- Note that the ‘=’ symbol is used to assign a value to a variable as a result of some processing and is not equivalent to the mathematical ‘=’ symbol.
- For this reason, some programmers prefer to use the ‘←’ symbol to represent the assign operation.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

5. A computer can compare two variables and select one of two alternative actions

- An important computer operation available to the programmer is the ability to compare two variables and then, as a result of the comparison, select one of two alternative 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, and the choice of alternatives is determined by the THEN or ELSE options.
- Only one of these alternatives will be performed.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

5. A computer can compare two variables and select one of two alternative actions

- A typical pseudocode example to illustrate this operation is:

```
IF student_attendance_status is part_time THEN
        add 1 to part_time_count
ELSE
        add 1 to full_time_count
ENDIF
```

- In this example the attendance status of the student is investigated, with the result that either the part_time_count or the full_time_count accumulator is incremented.
- Note the use of indentation to emphasise the THEN and ELSE options, and the use of the delimiter ENDIF to close the operation.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

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, DOWHILE and ENDDO, are used in pseudocode.
- The condition for the repetition of a group of actions is established in the DOWHILE clause, and the actions to be repeated are listed beneath it.

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

6. A computer can repeat a group of actions

- For example:

```
DOWHILE student_total < 50
    Read student record
    Print student name, address to report
    add 1 to student_total
ENDDO
```

4.3. An introduction to Algorithms and Pseudocode

Six basic computer operations that we consider here are:

6. A computer can repeat a group of actions

- In this example it is easy to see the statements that are to be repeated, as they immediately follow the DOWHILE statement and are indented for added emphasis.
- The condition that controls and eventually terminates the repetition is established in the DOWHILE clause, and the keyword ENDDO acts as a delimiter.
- As soon as the condition for repetition is found to be false, control passes to the next statement after the ENDDO.

4.3. An introduction to Algorithms and Pseudocode

Meaningful names

- When designing a solution algorithm, a programmer must introduce some unique names, which will be used to represent the variables or objects in the problem.
- All names should be meaningful.
- A name given to a variable is simply a method of identifying a particular storage location in the computer.
- The uniqueness of a name will differentiate this location from others.
- Often a name describes the type of data stored in a particular variable.
- For instance, a variable may be one of three simple data types: an integer, a real number or a character.
- The name itself should be transparent enough to adequately describe the variable; for example, number1, number2 and number3 are more meaningful names for three numbers than A, B and C.

4.3. An introduction to Algorithms and Pseudocode

Meaningful names

- If more than one word is used in the name of a variable, then underscores are useful as word separators, for example sales_tax and word_count.
- Most programming languages do not tolerate a space in a variable name, as a space would signal the end of the variable name and thus imply that there were two variables.
- If an underscore cannot be used, then words can be joined together with the use of a capital letter as a word separator, for example salesTax and wordCount.
- For readability, it is not advisable to string together words all in lower case. A name such as 'carregistration' is much harder to read than 'carRegistration'.