Cambridge Assessment International Education

Pseudocode Guide for Teachers

Cambridge International AS & A Level

Information Technology 9626

Use this guide for exams from 2025.

Version 0.2

Cambridge Pathway

Quality management

Cambridge International is committed to providing exceptional quality. In line with this commitment, our quality management system for the provision of international qualifications and education programmes for students aged 5 to 19 is independently certified as meeting the internationally recognised standard, ISO 9001:2015. Learn more at **www.cambridgeinternational.org/ISO9001**

© Cambridge University Press & Assessment December 2022

Cambridge Assessment International Education is part of Cambridge University Press & Assessment. Cambridge University Press & Assessment is a department of the University of Cambridge.

Cambridge University Press & Assessment retains the copyright on all its publications. Registered centres are permitted to copy material from this booklet for their own internal use. However, we cannot give permission to centres to photocopy any material that is acknowledged to a third party even for internal use within a centre.

Contents

Introduction

How should teachers use this guide?

We advise teachers to follow this guide in their teaching and make sure that learners are familiar with the style presented here. This will enable learners to understand any pseudocode presented in examination papers more easily. It will also give them a structure to follow so that they can present their algorithms more clearly in pseudocode when required.

Teachers should be aware that learners are not required to follow this guide in their examination answers or any other material they present for assessment. By definition, pseudocode is not a programming language with a defined, mandatory syntax. Any pseudocode (as long as it is not actual programming language code) presented by candidates will only be assessed for the logic of the solution presented – where the logic is understood by the examiner and correctly solves the problem addressed, the candidate will be given credit, regardless of whether the candidate has followed the style presented here. Using a recommended style will, however, enable the candidate to communicate their solution to the examiner more effectively.

1 Pseudocode

1.1 Case

Keywords are in uppercase, e.g. `IF`, `REPEAT`, `PROCEDURE`. Different keywords are explained in later sections of this guide.

Identifiers are in mixed case (sometimes referred to as camelCase) with upper case letters indicating the beginning of new words, for example `NumberOfPlayers`.

Example – meta-variables

```
REPEAT
   <Statement(s)>
UNTIL <condition>
```

1.2 Lines and line numbering

Where it is necessary to number the lines of pseudocode so that they can be referred to, line numbers are presented to the left of the pseudocode with sufficient space to indicate clearly that they are not part of the pseudocode statements.

Line numbers are consecutive, unless numbers are skipped to indicate that part of the code is missing. This will also be clearly stated.

Each line representing a statement is numbered. However, when a statement runs over one line of text, the continuation lines are not numbered.

1.3 Comments

Comments are preceded by two forward slashes //. The comment continues until the end of the line. For multi-line comments, each line is preceded by //.

Normally the comment is on a separate line before, and at the same level of indentation as, the code it refers to. Occasionally, however, a short comment that refers to a single line may be at the end of the line to which it refers.

Example – comments

```
// this swaps values of X and Y
   Temp ← X   // temporarily store X
   X ← Y
   Y ← Temp
```

2 Variables, constants and data types

2.1 Identifiers

Identifiers (the names given to variables, constants, procedures and functions) are in mix case. They can only contain letters (`A–Z`, `a–z`), digits (`0–9`) and the underscore character (_). They must start with a letter and not a digit. Accented letters should not be used.

As in programming, it is good practice to use identifiers that describe the variable, procedure or function they refer to. Single letters may be used.

Keywords identified elsewhere in this guide should never be used as variable names.

Identifiers should be considered case insensitive, for example, `Countdown` and `CountDown` should not be used as separate variables.

2.2 Assignments

The assignment operator is ← .

Assignments should be made in the following format:

```
<identifier> ← <value>
```

The identifier must refer to a variable. The value may be any expression that evaluates to a value of the same data type as the variable.

Example – assignments

```
Counter ← 0
Counter ← Counter + 1
Wages ← NumberOfHours * HourlyRate
```

3 Common operations

3.1 Input and output

Values are input using the INPUT command as follows:

```
INPUT <identifier>
```

The identifier should be a variable name.

Values are output using the PRINT command as follows:

```
PRINT <value(s)>
```

The value, or values, as there could be several values, separated by commas, will be identifiers or could be text enclosed in quotes.

Example – `INPUT` and `PRINT` statements

```
INPUT Answer
PRINT Score
PRINT "You have ", Lives, " lives left"
```

3.2 Arithmetic operations

Standard arithmetic operator symbols are used:

+ Addition

– Subtraction

* Multiplication

/ Division

Multiplication and division have higher precedence over addition and subtraction (this is the normal mathematical convention). However, it is good practice to make the order of operations in complex expressions explicit by using parentheses.

3.3 Relational operations

The following symbols are used for relational operators (also known as comparison operators):

> Greater than

< Less than

>= Greater than or equal to

<= Less than or equal to

= Equal to

<> Not equal to

4 Selection

4.1 IF statements

`IF` statements may or may not have an `ELSE` clause.

`IF` statements without an else clause are written as follows:

```
IF <condition>
   THEN
      <statement(s)>
ENDIF
```

`IF` statements with an else clause are written as follows:

```
IF <condition>
   THEN
      <statement(s)>
   ELSE
      <statement(s)>
ENDIF
```

When `IF` statements are nested, the nesting should continue with consistent indentation.

Example – nested `IF` statements

```
IF ChallengerScore > ChampionScore
   THEN
      IF ChallengerScore > HighestScore
         THEN
            PRINT ChallengerName, " is champion and highest scorer"
         ELSE
            PRINT Player1Name, " is the new champion"
      ENDIF
   ELSE
      PRINT ChampionName, " is still the champion"
      IF ChampionScore > HighestScore
         THEN
            PRINT ChampionName, " is also the highest scorer"
      ENDIF
ENDIF
```

4.2 CASE statements

CASE statements allow one out of several branches of code to be executed, depending on the value of the identifier.

CASE statements are written as follows:

```
CASE OF <identifier>
   <value 1> : <statement1>
               <statement2>
               ...
   <value 2> : <statement1>
               <statement2>
               ...
   ...
ENDCASE
```

An OTHERWISE clause can be the last case:

```
CASE OF <identifier>
   <value 1> : <statement1>
               <statement2>
               ...
   <value 2> : <statement1>
               <statement2>
               ...
   OTHERWISE : <statement1>
               <statement2>
               ...
ENDCASE
```

Each value may be represented by a range, for example:

```
<value1> TO <value2> : <statement1>
                <statement2>
                ...
```

Note that the case clauses are tested in sequence. When a case that applies is found, its statement is executed and the CASE statement is complete. Control is passed to the statement after the ENDCASE. Any remaining cases are not tested.

If present, an OTHERWISE clause must be the last case. Its statement will be executed if none of the preceding cases apply.

Example – CASE statement

```
INPUT Wagegrade
CASE OF Wagegrade
   'M' : HourlyRate ← 15
   'S' : HourlyRate ← 20
   'A' : HourlyRate ← 25
   'D' : HourlyRate ← 30
   OTHERWISE : PRINT "INVALID Wage grade"
ENDCASE
```

5 Iteration (repetition)

5.1 Post-condition (REPEAT) loops

Post-condition loops are written as follows:

```
REPEAT
   <Statement(s)>
UNTIL
<condition>
```

The statements in the loop will be executed at least once. The condition is tested after the statements are executed and if the `UNTIL` statement is true, the loop terminates, otherwise the statements are executed again.

Example – `REPEAT UNTIL` statement

```
REPEAT
   PRINT "Please enter the password"
   INPUT Password
UNTIL Password = "Secret"
```

5.2 Pre-condition (WHILE) loops

Pre-condition loops are written as follows:

```
WHILE <condition>
   <statement(s)>
ENDWHILE
```

The condition is tested before the statements, and the statements will only be executed if the condition is true. After the statements have been executed the condition is tested again. The loop terminates when the condition is not true.

The statements will not be executed if, on the first test, the condition is not true.

Example – `WHILE` loop

```
WHILE Number > 9
   Number ← Number - 9
ENDWHILE
```

6 Procedures

6.1 Defining and calling procedures

A procedure with no parameters is defined as follows:

```
PROCEDURE <identifier>
   <statement(s)>
ENDPROCEDURE
```

The procedure can have parameters or values passed to it. For example:

```
PROCEDURE <identifier> (<parameter>)
   <statement(s)>
ENDPROCEDURE
```

There can be more than one parameter.

The `<identifier>` is the identifier used to call the procedure.

Procedures defined as above should be called as follows:

```
CALL <identifier>
```

or

```
CALL <identifier> (value1, value2 etc.)
```

These calls are complete algorithm statements.

Example – use of a procedure

Defining a procedure:

```
PROCEDURE squared(number)
   Square  ←  number*number
ENDPROCEDURE
```

Algorithm calling that procedure:

```
INPUT length
CALL Squared(length)
PRINT Square
```

7 File handling

7.1 Input and output

Instead of INPUT and PRINT, in file handling operations READ and WRITE are used.

READ causes data to be read (input) from a file.
WRITE causes data to be written (output) to a file.

Sample question

Many companies use computers to process their payroll. This involves the updating of a master file.

Every week the master file has to be updated to insert (I) the details of new workers and to delete (D) the details of workers who have left the company. Records also have to be amended (A) if an existing worker's details have changed.

(a) Including the use of nested IF THEN statements, complete the following algorithm which shows the process of updating the master file.

```
Read first record from transaction file
Read first record from old master file
WHILE not end of transaction file
    IF transaction file IDnumber = master file IDnumber
        THEN
```

Example files

The explanation of how a possible algorithm solution would work is given at the end of this section. As it stands, it is quite a long description, but in order to avoid it being longer, the example files given below are shorter than those given in any examination paper.

The transaction file may look like this:

Transaction	IDnumber	Jobtitle
A	1569	Picker
D	2378	Checkout
I	3178	Cleaner

Values are read from a file using the instruction `READ` as follows:

```
READ first record from transaction file
```

This will read data from the fields `Transaction`, `IDnumber` and `Jobtitle` in the transaction file.

The master file may look like this:

IDnumber	Department
1012	Picker
1569	Checkout
2378	Checkout
2856	Driver
3276	Driver
3576	Checkout

Data from the fields `IDnumber` and `Department` can be read from the master file as follows:

```
READ first record from master file
```

These fields can then be referred to in statements such as:

```
IF transaction file IDnumber = master file IDnumber
```

When data needs to be stored in a new file, the data is said to be 'written' to the file. If a new master file is to be created, the following statement would be used if the old master file record was to be stored unchanged on the new master file:

```
WRITE master file record to new master file
```

The records from either file will be read one by one. In order to prevent the algorithm attempting to read records after the last one in a file, a check has to be made. This is done by checking that the end of the file has not yet been reached. All files have what is called an end of file marker after the final record in the file. This makes it easy to check if all the records have been read. This check is usually done using the first statement in a `WHILE…ENDWHILE` loop or using the final statement in a `REPEAT…UNTIL` loop:

```
WHILE not end of (name) file
…
…
ENDWHILE
```

OR

```
REPEAT
…
…
UNTIL end of (name) file
```

Possible solution to sample question

```
1.  READ first record in transaction file
2.  READ first record in old master file
3.  WHILE not end of transaction file
4.     IF transaction file IDnumber = master file IDnumber
5.        THEN
6.           IF Transaction = ”A”
7.              THEN
8.                 WRITE transaction file record to new master file
9.           ENDIF
10.          READ next transaction file record
11.          READ next master file record
12.       ELSE
13.          WHILE master file IDnumber < transaction file IDnumber
14.             WRITE master file record to new master file
15.             READ next master file record
16.          ENDWHILE
17.          IF Transaction = “I”
18.             THEN
19.                WRITE transaction file record to new master file
20.                READ next transaction file record
21.          ENDIF
22.    ENDIF
23. ENDWHILE
24. WRITE remaining master file records to new master file
```

It is not usual practice to number statements in an algorithm but for the purpose of explaining what this algorithm does, numbering has been used here.

Explanation of how this algorithm works:

Statements 1 and 2 read the first records from each of the files.

Statements 3 and 23 form a loop so that all the other statements within the loop are repeatedly executed until there are no more records left in the transaction file.

Statement 4 checks to see if the `IDnumber` in the first record of the transaction file is the same as the `IDnumber` in the first record of the master file. If it is, then (statement 5) statements 6 to 11 are carried out. In this example, however, 1569 in the transaction file does not match 1012 in the master file so statements 13 to 21 are carried out.

Statement 13 checks if the `IDnumber` in the master file record (first time through, this is still the first record) is less than the `IDnumber` in the transaction file record (first time through it is still the first record). 1012 is less than 1569 so statements 14 and 15 are carried out. The first record of the master file is written to a new master file. The next record (the second) is read from the master file. Statement 16 is now carried out. This is the corresponding statement to 13 which is the start of the `WHILE…ENDWHILE` loop.

The comparison is now made again but this time the `IDnumber` in the second master file record, 1569 is not less than the `IDnumber` in the first transaction file record, 1569.

This time statements 14 and 15 are not carried out and the next statement after the end of the loop, statement 17 is carried out.

Statement 17 checks to see if the Transaction is “I” (to insert a record). If it is then (statement 18), statements 19 and 20 are carried out. In this case, the Transaction for `IDnumber` 1569 is “A” (to amend a record) so statements 19 and 20 are not carried out. The `IF` condition (statements 17 to 21) is exited.

As the `ELSE` part of the first `IF` condition (statements 4 to 22) has been carried out, this `IF` is also exited.

Statement 23 takes the algorithm back to statement 3. The end of transaction file has not been reached so statement 4 is carried out.

Statement 4 is now true so statement 5 causes statement 6 to be carried out. This checks to see if the Transaction is ”A”, and this time it is, so statements 7 and 8 are carried out. The transaction file record is written to the new master file. This effectively changes the Department from Checkout to Picker (in other words, the file needed amending because the worker had changed jobs within the company).

The algorithm then moves on to statements 10 and 11 which reads the next record from the transaction file (record 2) and the master file (record 3).

Statements 12 to 21 are ignored as they follow the `ELSE` statement. The `IF` condition (statements 4 to 22) is exited.

Statement 23 takes the algorithm back to statement 3. The end of file marker has not been reached.

Statement 4 now checks to see if the `IDnumber` in (what is now) the second record of the transaction file is the same as the `IDnumber` in (what is now) the third record of the master file. In this example, 2378 in the transaction file does match 2378 statement 5, which causes statements 6 to 11 to be carried out.

Statement 6 checks to see if the Transaction is ”A”. In this case it is not and so statements 7 to 9 can be ignored.

Statements 10 and 11 are now carried out. The next records from the transaction file and the master file are read.

Because nothing has been written to the new master file and the only possible transactions could be ”A” or ”D”, the third record is effectively deleted because there is no statement writing it to the new master file. In other words, to delete a record from a master file it has to be read from the old master file but not written to the new master file. As statement 4 was true, statements 12 to 21 can be ignored. The `IF` condition is exited.

Statement 23 takes the algorithm back to statement 3. The end of file has not been reached so statement 4 is carried out.

The records being compared are now the third record in the transaction file, `IDnumber` 3178 and the fourth record in the master file, `IDnumber` 2856. They are not the same, so statements 5 to 11 are ignored. The algorithm jumps to statement 12 and carries out statements 13 to 21.

Statement 13 checks to see if the master file `IDnumber` is less than the transaction file `IDnumber` and in this case it is (2856 is lower than 3178). Because it is less, the instructions in the second `WHILE…ENDWHILE` loop are carried out.

The master file record is written to the new master file and the next master file record is read. (`IDnumber` 3276).

The `ENDWHILE` statement takes the algorithm back to the `WHILE` statement which is now no longer true (3276 is not less than 3178) so the `WHILE...ENDWHILE` loop is exited and statement 17 is carried out.

Now, the Transaction is "I" so statements 18 to 20 are carried out. The transaction file record is written to the new master file. The next record in the transaction file is, in fact, the end of file marker.

The `IF` condition (statements 17 to 21) is exited. As the `ELSE` statements in the first `IF` condition have been carried out, the `IF` condition is exited.

Statement 23 takes the algorithm back to statement 3 and now the end of the transaction file has been reached. This means the main `WHILE...ENDWHILE` loop is exited and statement 24 is executed so the master file records `IDnumbers` 3256 and 3276 are written to the new master file and the algorithm is finished.

Index of symbols and keywords

School feedback: ‘While studying Cambridge IGCSE and Cambridge International A Levels, students broaden their horizons through a global perspective and develop a lasting passion for learning.’

Feedback from: Zhai Xiaoning, Deputy Principal, The High School Affiliated to Renmin University of China

We are committed to making our documents accessible in accordance with the WCAG 2.1 Standard. We are always looking to improve the accessibility of our documents. If you find any problems or you think we are not meeting accessibility requirements, contact us at **info@cambridgeinternational.org** with the subject heading: Digital accessibility. If you need this document in a different format, contact us and supply your name, email address and requirements and we will respond within 15 working days.

Cambridge Assessment International Education, The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA, United Kingdom
t: +44 (0)1223 553554 email: info@cambridgeinternational.org www.cambridgeinternational.org

© Cambridge University Press & Assessment December 2022