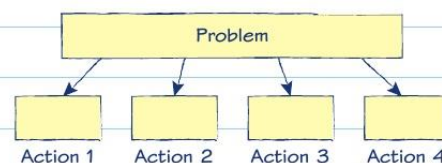


Identifying problems and processes

Computational thinking enables you to analyse a problem, break it down into smaller parts, recognise patterns within the problem and finally identify a strategy to solve it. Over the next few pages, you will revise the first stage of computational thinking – decomposition.

Decomposition – step 1

Every computer program is made up of a number of processes or actions. Before an app can be written, the problem to be solved and the processes (actions) inside the app need to be identified. The first step in decomposition involves identifying and describing the problem and the processes required to solve it.



App to collate spreadsheets

An app is being developed to bring together several spreadsheets from different members of a team into one workbook which can then be used for a mail merge.

The mail merge is to be from a worksheet in the workbook using data which is selected and copied from member worksheets.

u Photography	-----	-----	E
-----	Animal Management	Photography	C
-----	Computer Games Develop	Electrical	C
-----	Fashion&Textile design	Animal Management	N
English Literatu	-----	-----	P
-----	Beauty therapy	Hairdressing	P
-----	Hairdressing	Animal Management	C
-----	Beauty therapy	Hairdressing	E
Photography	-----	-----	P
Biology	-----	-----	P
-----	Sport	Fitness	E
u Psychology	-----	-----	P
-----	Sport	Fitness	T
-----	Computer Games Develop	Wood	P
-----	Sport	Fitness	P
Art	-----	-----	N
-----	Theatrical & media makeup	Photography	E
-----	Hairdressing	Public Services	B
-----	Engineering	Computer Games Deve	C
-----	Fitness	Sport	N
-----	-----	-----	P
Art	-----	-----	H
-----	Theatrical & media makeup	Photography	C

Only data which are names of courses are to be copied to the worksheets for mail merges. The cells with dashes in them are to be ignored.

The problem must be clearly described in language that will be familiar to the user. Later, the solution will be checked against the problem to ensure all needs have been met.

- 1 Copy and paste member worksheets into designated worksheets in the master workbook.
- 2 Run macro code to loop into each of the designated worksheets. Loop down each row of data and loop along the columns in each row.
- 3 Select every data item that is not dashes and copy.
- 4 Move to appropriate cell in merge worksheet, then paste special as value.
- 5 Complete loops when blank cells are found.
- 6 Save workbook.

Once the problem has been described, the processes needed to program the solution can be identified. These become the framework of the solution, with the detailed steps needed to implement each process added at the next stage.

Now try this

You are designing an app to allow builders to price jobs.

- (a) Write down a list of outputs needed for this app.
- (b) Write down a list of inputs needed for this app.
- (c) Choose one input and describe the actions needed to test it for validity.

Think about what data is actually needed and don't include more than that.

Breaking down problems and processes

Once the problem and processes have been identified, the next stage of decomposition is to break down the problem and processes into a sequence of steps.



App to support a competition

An app is being developed to support a charity treasure hunt. There will be four teams of up to six people. Each team will be given photographs of various places within the treasure hunt boundary and clues on sheets of paper. When calculating the scores, the following rules must be applied:

- Photo questions are each worth 1 point.
- Other questions are each worth 2 points.
- Deduct 1 point for a wrong answer.
- Blank answers are each worth 0 points.
- The points total is scaled according to how many are in the team: each member is worth 25%, and the total is then inverted. So a team of four has a scaling of 100% and a team of two has 200%.

Last year, the results were calculated in a spreadsheet. This year the organisers plan to use a smartphone app.

Steps in calculating points using app

- 1 Enter the team name with number of members in the team.
- 2 Enter number of correct photo questions.
- 3 Enter number of correct other questions.
- 4 Enter number of wrong questions.
- 5 Calculate points for correct other questions by multiplying number correct by 2.
- 6 Calculate subtotal by adding adjusted other questions to correct photo questions, then subtracting number of wrong questions.
- 7 Calculate team scaling.
- 8 Multiply subtotal by team scaling.
- 9 Repeat steps 1 to 8 for all teams.
- 10 Sort by totals to show winning team.

Team	Subtotal	Scaling	Total
A (2)	49	200%	98
B (3)	58	133%	77
C (6)	90	67%	60
D (4)	37	100%	37

Now try this

A local independent corner store would like to check its stock using a mobile phone app.

- (a) What do you think would be needed to make this practical?
- (b) What steps are needed to make the app work using the phone's camera to scan barcodes on stock items?

Think specifically about how barcodes can be used.

Communicating problems and processes

On this page, you will revise the final stage of decomposition – how to describe and communicate the key features of problems and processes to clients and other programmers.

Communicating algorithms

Describing problems and processes as a set of structured steps – an algorithm – will enable clients and programmers to understand how a proposed solution will work. At this stage, mistakes in the understanding of the problem or design flaws may become apparent before the project moves to the coding phase.

Links Revise algorithm design on page 8.



Restaurant tablet

A restaurant is planning to use a tablet for each table so diners can browse the menu using video footage of the ingredients and dishes, and read reviews from other diners before choosing their meal. The tablet will display the final bill, at which point diners can log into their PayPal account or make a payment using card or cash to the waiter. Clients may then enter feedback about their meal.

Using pseudocode

Pseudocode can be used to explain to clients and other programmers how code will work.

This pseudocode shows the process by which a customer pays and is given an option to rate their experience at the restaurant.

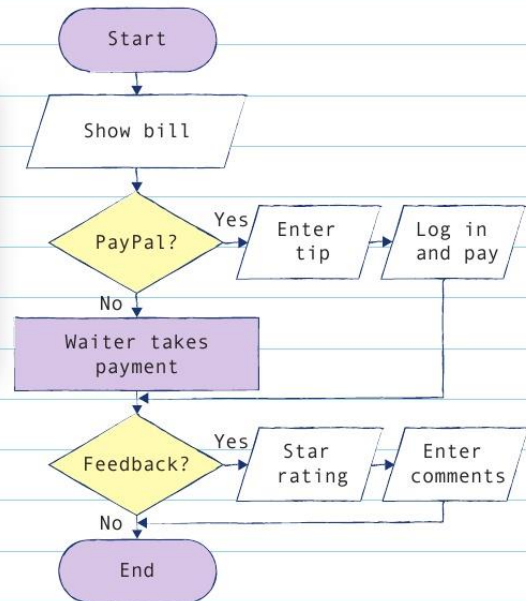
```

If PayPal
    Enter tip amount
    Log in to PayPal
If not PayPal
    Waiter takes payment
User selects whether they want to leave feedback
If yes
    Enter star rating
    Write comments
Show thank you screen on the tablet
    
```

Links Revise how to produce, apply and interpret pseudocode on pages 9–10.

Using a flowchart

You could also use a flowchart to show the algorithms required to demonstrate the processes.



Now try this

An independent jeweller is setting up a website as a retail outlet. It would like to offer a loyalty discount for customers who purchased within the last year as well as giving a reduction of 10% on the sale for more than one item and a choice of postage rates of next day or economy. The jeweller is VAT registered, so VAT needs to be added to the sale after any discounts and postage.

Describe the processes that would be needed to calculate a quote for a customer visiting the website.

What actions are needed to produce the quote? Each action will be a process.

Pattern recognition

Once the problem and processes have been described, the next step involves pattern recognition where you look for repeating features within problems and between problems. This will enable you to create code that can be reused in other apps.

Common elements and individual differences

- Identifying common elements or features in problems needing coded solutions or within systems requiring maintenance can result in producing program code that can be re-used in other apps.
- Identifying any differences and individual elements within problems that can utilise common code need interpreting so library code can be adapted by using appropriate parameters or branching within subroutines.

Code libraries

Code libraries are used by many organisations to improve the effectiveness of programming teams by keeping copies of program segments that are easy to find and to re-use.

Documentation is an essential element of a code library as this is needed to clearly identify code segments with how they can be used as reliable building blocks for new apps.

Debugging time can be reduced by using library code as these program segments will have already been extensively tested and signed off as fit for purpose.

Parameters

Parameters are vital for much re-usable code to control the values that can be passed into a subroutine and so make the workings of this code reliable and predictable.

This code uses 0, 99, 2, 5 as fixed numbers, making it very inflexible as the code would need editing to be used elsewhere.

```
Sub ColumnSort(ByRef SA (,))
Dim Z, FirstRow, Passes, Item As Integer
Dim Temp As String
FirstRow = 0
Passes = FirstRow
While Passes <= 99
Item = FirstRow
While Item <= 99
If SA(2, Item) > SA(2, Item + 1) Then
For Z = 0 To 5
Temp = SA(Z, Item)
SA(Z, Item) = SA(Z, Item + 1)
SA(Z, Item + 1) = Temp
Next Z
End If
Item = Item + 1
End While
Passes = Passes + 1
End While
End Sub
```

An example of code that is hard to re-use.

This code uses parameters NoCols, FirstRow, LastRow, Col instead of fixed numbers, making the code much more re-usable.

```
Sub ColumnSort(ByRef SA(,), NoCols, FirstRow, LastRow, Col)
Dim Z, Passes, Item As Integer
Dim Temp As String
Passes = FirstRow
While Passes <= LastRow - 1
Item = FirstRow
While Item <= LastRow - 1
If SA(Col, Item) > SA(Col, Item + 1) Then
For Z = 0 To NoCols
Temp = SA(Z, Item)
SA(Z, Item) = SA(Z, Item + 1)
SA(Z, Item + 1) = Temp
Next Z
End If
Item = Item + 1
End While
Passes = Passes + 1
End While
End Sub
```

An example of the same code tweaked to make it re-usable.

Now try this

Write down all the benefits and disadvantages of reusing code within an organisation. Do the positives outweigh the negatives?

Use a single line for each item so it's easy to see whether the positives outweigh the negatives.

Describing patterns and making predictions

Once you have identified repeating features, you will need to describe the patterns. You can then make predictions based on these patterns which will enable you to design program algorithms.



Cleaning dates data

Research was carried out to find computer games titles that had been on sale during the last three decades. Over 10000 items of data were downloaded from the internet. Some dates were not recognised as such by the spreadsheet. The number of these entries made manual editing a poor option due to the time it would take and the errors that might be introduced into the data set by so much repetitive work.

```
000000002016-06-21-0000Jun 21, 2016
000000002016-08-23-0000Aug 23, 2016
000000002016-08-23-0000August 23, 2016
000000002016-09-10-0000Sep 10, 2016
1994-12-09JP
1994-12-22JP
1995-01
1995-01-27JP
1995NA
1995NA
```

- Each date was in one of these forms:
- 00000000 then date (yyyy-mm-dd) then -0000 then the date as text
 - date (yyyy-mm-dd) then two letters
 - date (yyyy-mm)
 - date (yyyy) then two letters.

Cleaning the dates data

Search and replace could be used to clean some of the data (for example, removing 00000000). Code could edit the data into a consistent yyyy-mm-dd format, which could then be used as dates in the spreadsheet.

Code was written to loop down the data, copying each item into a variable that was then edited into the yyyy-mm-dd form, according to the type of form it started with.

Pseudocode 1

```
Select top cell of the dates column
Start loop
Copy the cell into a variable, CellContent
If left 8 characters of CellContent = 00000000
    CellContent = mid(CellContent, 9,10)
If length of CellContent =12
    CellContent = left(CellContent, 10)
```

Loops

Loops are used to repeat code that uses patterns in the data for processing, such as deleting parts of data items that are not wanted.

```
If length of CellContent =7
    CellContent = CellContent + "-01"
If length of CellContent =6
    CellContent = left(CellContent,4) + "-01-01"
Set active cell to CellContent
Move down a cell
Loop if active cell not empty
```



For more on loops, see page 19.

Now try this

- 1 Create a spreadsheet to generate the first 20 numbers in the Fibonacci sequence.
- 2 Produce an algorithm that calculates the n th term of the Fibonacci sequence.

Each term in the Fibonacci sequence is the sum of the previous two terms: 1, 1, 2, 3, 5, 8, 13.

Pattern generalisation and abstraction

After pattern recognition, the next step is to generalise and abstract these patterns to identify all the information necessary to solve a problem. To help you do this, you need to revise variables, constants, key and repeated processes, inputs and outputs.

Representing a problem as code

Identifying information that is necessary to solve an identified problem is an essential part of the programming life cycle. Parts of a problem or system can be represented in code as variables, constants, key processes, repeated processes, inputs and outputs.

	Definition
Variables	Values in a problem or system that may change Usually input by the user or may result from calculation
Constants	Values in a problem or system that remain fixed while the code runs
Key processes	Processes that are essential to understanding of a problem or how a system works
Repeated processes	Processes that occur multiple times within a problem
Inputs	Values read or entered into the system
Outputs	Information presented to the user



Workout app

You recently saw a television programme which suggested that every opportunity to exercise should be taken as 'every little helps!'

As you spend a lot of time using a computer, you think that an app to help encourage a work out whilst using a computer might be useful.

The mouse could be moved to the corners of the screen and clicked, exercising the lower arm, wrists and fingers. These actions could be repeated with the other side of the body.

Even the toes and feet could be exercised by placing the keyboard on the floor and alternatively tapping the space bar and numeric keypad to ensure the foot has some movement.

Key and repeated processes

Mouse inputs are to click onto one of 4 target images placed at the four corners of the form. Clicking on the correct target image will increment (add 1) to the variable, TapCount.

Keyboard inputs are to tap the spacebar or one of the number keys. Tapping any of these will increment (add 1) to the variable, TapCount.

```

Start button mouse click
  Initialise TapCount, StartTime to 0
  Initialise Target, to random between 1-4
  Set image(Target) to active
Target image mouse click
  IF Target matches image
    Increment TapCount
    Play success sound
  ELSE
    Play fail sound
    Call Update statistics
Key press event
  IF space or number pressed
    Increment TapCount
    Play success sound
  ELSE
    Play fail sound
  Call Update statistics
Update statistics subroutine
OUTPUT TapCount
FOR Countdown = 5 TO 1 STEP -1
  Display Countdown
  
```

A start button can start the workout by clearing the variables, TapCount, StartTime, to zero.

The screen will show the target images and workout statistics. Target images can have three variants for active, inactive and correct click. Workout statistics can show time taken, number of correct taps/clicks, number of incorrect taps/clicks, average speed and accuracy percentage.

Speakers can make a sound each time a correct click or key press is made or a different sound if a wrong click or key press is made.

Now try this

Write down two key processes and two repeated processes for a website shopping cart.

Think about the actions that take place in the website shopping cart.

Representing the new system

The last element of pattern generalisation and abstraction is to represent the new system using variables, constants, key processes, repeated processes, inputs and outputs. Filtering and ignoring any information not needed to solve the problem will enable you to focus on the actual problem.



Links For more information on variables, constants, key processes, repeated processes, inputs and outputs, see page 6.

Filtering information

Before writing a program, think carefully about what is actually needed to help solve the problems you are asked to code.

In a database, for example, it is very easy to add fields to a table so there is a place for every possible aspect of the data subject. A better approach is to look at the information that is required from the system, which can then be matched to the data needed to populate the reports and screens outputting from the system.



Health club members list

A system is being written to handle the members list for a health club. It will hold all the information needed for the club's day-to-day operations.

The system should be quick and easy to use as well as using validation techniques to reduce errors typed into the system.

Members could be issued cards which allow scanning into the system by barcode, swiping or NFC (contactless near-field communication).

Reports can be used to extract data from the system onto paper and data can be exported for use in mail merges.

Printed outputs

Reports from the health club members database could include:

- schedule showing the activities booked for that day, week or month
- members list summary
- member details with all the information about an individual member
- members activity log detailing the activities undertaken over a period of time
- members payments due statement with what is currently owed to the club
- booking receipt to confirm an activity has been reserved.

Tables

The database could include the following tables:

- members
- bookings
- activities
- payments.

Forms

These forms will allow easy navigation of the database to make it more user friendly:

- main menu to click buttons navigating to other parts of the system
- members to add, edit or delete a member
- bookings to add, edit or delete a booking
- payments to record a payment
- reports to choose a report for printing.

Now try this

Create a data dictionary identifying the fields needed for the tables in a database to keep track of a health club members list.

The tables are listed in bullets on this page. What fields would each table need?

Algorithm design

The final stage in computational thinking is to design the algorithm using a step-by-step strategy to solve the problem. This will enable you to clearly understand how the program will work.

Designing an algorithm

- 1 Define the overall purpose of the program.
- 2 Divide into the processes needed.
- 3 Plan the steps needed for each process.
- 4 Check the algorithm against the original need to confirm it will be fit for purpose.

Algorithm design

There are often many algorithms in a program which can be at different levels of detail. Designing a program can start with an overall algorithm to summarise how the system works, with other algorithms providing detail needed to design smaller sections of code.

Links For more information on standard algorithms, see pages 26–29.



Stock control system

The owner of a second-hand furniture shop is considering writing a stock control program because she quite enjoys coding and wants to have control over how the app looks and behaves. The app is to run on a PC in the shop as a restricted version so customers can find out if there is anything in the back stock room that interests them.

New item pseudocode

This algorithm enters a new item of stock, checks all the data present and then saves to a stock data file.

```
If any field has not been completed
  Display message
  Show * next to every field not completed
  Place cursor in first field not completed
When all fields completed
  Generate stock number
  Copy fields to Stock file
  Save Stock file
  Clear fields on the New item form
```

Stock search pseudocode

This algorithm searches for an item in a stock data file and shows the results on-screen.

```
Click on search button
If search textbox is empty and furniture checked
  Loop around stock array
  Display all furniture
If search not empty and furniture checked
  Loop around stock array
  Display all furniture matching textbox
If search empty and other checked
  Loop around stock array
  Display all non-furniture items
If search not empty and other checked
  Loop around stock array
  Display other items matching textbox
Place cursor into search textbox
```

Sale pseudocode

This algorithm allows the user to record details of the sale of an item and then save to the data file.

```
Select item sold
Select customer
If customer not known
  Open New customer form
  Enter new customer
  Click on Confirm button
  Copy fields to Customer array
  Save Customer file
  Close form
Show customer information
Click confirm sale
Save Stock, Customer and Sales files
Print receipt
Clear fields on the Sales form
```

Now try this

A car electronic cruise control keeps the vehicle at a constant speed by using the accelerator, gear change and brake, until the driver cancels cruise control by using the brake.

Write pseudocode to design a cruise control system that also links into the sat-nav to prevent the vehicle from exceeding speed limits.

What inputs and outputs are needed for the control device?

Structured English (pseudocode)

There are two main methods you can use to plan program algorithms – pseudocode (structured English) and flow charts. On this page, you will revise commonly used pseudocode terms and how to apply them. Pseudocode can be converted to a programming language to implement:

```
REPEAT UNTIL the end of file
  READ into Sectors(Y)
  Increment Y
Set LastSector to Y - 1
Close the data file
```



```
Do
  Input(1, Sectors(Y))
  Y = Y + 1
Loop Until EOF(1)
Lastsector = Y - 1
FileClose(1)
```

Representing operations

- **BEGIN...END** can be used for any code which you want to keep separate or simply to show where your algorithm starts and finishes.
- **INPUT/OUTPUT** are for any part of the algorithm that allows data in or out such as typing into a textbox or displaying a result.
- **PRINT** is used when a hard copy is produced.
- **READ/WRITE** are for when data are read into the algorithm from a file or written out to a file.

Representing decisions

- **IF...THEN...ELSE...ELSEIF (ELIF)** are used for branches in the algorithm.
- Simple branches use **IF...THEN** to define a test condition and action for condition is met which are usually indented or you could use **BEGIN...END** for them.
- **ELSE** is used when actions are required when an **IF...THEN** condition is not met.
- **ELSEIF (or ELIF in some programming languages)** is for actions to be carried out if the previous **IF...THEN** condition is not met and a further test needs to be made.
- **WHEN** is used to represent select case structures with several branches possible based upon the contents of a variable.

Representing repetition

Each of these are written as a single pseudocode line to define the loop followed by the repeated code indented in the code.

- **FOR** is the unconditional **FOR...NEXT** loop with the pseudocode line showing how many times the loop iterates.
- **REPEAT UNTIL** is a conditional loop with the pseudocode line defining what ends the loop.
- **WHILE/WHILE NOT** are conditional loops with code defining what allows the iteration.

Dos and don'ts

- 👍 Use program command words to identify branch and loop structures.
- 👍 Use indents to show what's included in a structure.
- 👍 Summarise sections of code.

- 👎 Don't write actual code which is ready to run.
- 👎 Don't produce pseudocode in your program editor.
- 👎 Don't include too much detail about how code will do an action such as swapping items.

Now try this

Produce pseudocode for spreadsheet code to copy rows in a worksheet to one of three other worksheets based upon contents of first cell in each row. A fourth worksheet is used for copies of rows where first cell does not match. This needs to be able to handle any number of rows, starting in a cell named 'FirstCell'.

Read the requirement carefully then consider how you would explain the algorithm in simple words.

Interpreting pseudocode

Pseudocode is used to plan program algorithms. It enables the programmer to visualise how a program will work and to see improvements to the logical structures and processes after reading it. On this page, you will revise how to interpret and develop pseudocode.



Zilch

Zilch is a game played with six dice where the players each take turns throwing the dice to earn points. A target score is set.

The game is won by the player who goes over the target score after the same number of turns as the other players.

When a player has a turn they keep on throwing until either they 'stick' to keep their points or throw a non-scoring combination of dice - 'Zilch' - when their points for the turn are zero.

Zilch dice points rules

There are six dice with several possible points schemes in use. We shall use the scoring below:

1, 2, 3, 4, 5, 6	3000 pts
Three pairs	1500 pts
Three the same	dice number × 100 pts
Dice showing 5	50 pts
Dice showing 1	100 pts

Interpreting and developing code

The process calculating the outcomes of each stage of the game will produce points earned and dice left for the next throw. This will be large and complex, so needs to be broken down into sub-processes, making them easier to focus on and write.

How would you do this if playing with real dice? The first process is to find out if the highest score is thrown, then next highest and so on. Each of these algorithms will be a section of pseudocode.

Preparation for identifying the highest score can take place inside this loop by counting how many of each number has been thrown in the Totals() array. This loop can also show the dice number on the screen.

The structure of this pseudocode can be evaluated against the requirement to identify a throw of 1, 2, 3, 4, 5, 6 using dry runs.

The code here is reasonably effective. Less effective code could use another FOR loop to count how many of each number was thrown. Less effective code might test for a number being thrown more than once, rather than 0.

The highest score is 1, 2, 3, 4, 5, 6 with the method shown here using an array, Totals(), to find out if each number has been used once. Before the check, each item in Totals() is set to 0 using a FOR loop.

```

Set Score to 0
FOR X = 1 TO 6
  Set Totals(X) to 0
FOR X = 1 TO 6
  Set Throw to random number between 1-6
  Set Dice(X) to Throw
  Increment Totals(Throw)
  Show Dice(X) on the form with its number
Set Winner to True
FOR X = 1 TO 6
  IF Totals(X) = 0 THEN set Winner to False
IF Winner
  Add 3000 to Score
  Show Score on the form
END subroutine
  
```

A FOR loop throws the dice using the variable, Throw, to hold the number for each dice.

The variable, Winner, is set to true then a FOR loop iterates through the Totals() array, changing Winner to false if any of the numbers were not thrown.



Now try this

Produce a description of how this pseudocode calculates a score in Zilch:






The sequence is important. Start with the first line of the pseudocode and interpret the meaning. Remember, indents show how much code is inside a structure such as a FOR loop.

```

FOR X = 1 TO 6
  IF Dice(X) = 1
    Add 100 to Score
    Decrement DiceLeft
  IF Dice(X) = 5
    Add 50 to Score
    Decrement DiceLeft
  Show Score on the form
  
```

Flow charts

Flow charts provide a pictorial complement to pseudocode, helping you to plan algorithms. British Computer Society (BCS) symbols are commonly used in flow charts.

Flow chart shape	Description
	Process used for anything in code that cannot be represented by any of the other symbols.
	Decision shows where there is a choice of two paths with the condition that needs to be met written inside the symbol.
	Input/output shows every place where data or events come into or leave the algorithm.
	Connectors are used to reduce the need to draw lines across the flow chart.
	Start/end symbols show the entry and exit points in the algorithm.

Alarm flow chart

This flow chart illustrates how an alarm works on a mobile phone.

The flow chart has to begin and finish with start/end symbols. The rest of it needs to show the routes that are possible in the code.

The alarm is set with an input from the user.

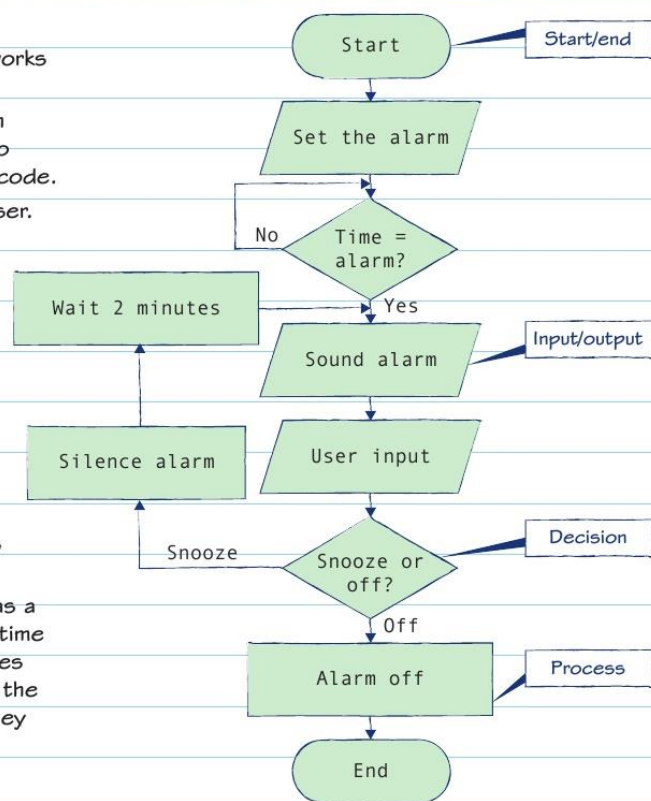
A decision is used to show how the system regularly checks the actual time to the alarm. When they match, the yes branch is taken so the alarm is sounded.

The user can now input into the system to either turn the alarm off or snooze.

If snooze, the app enters a process to wait a short time before the alarm sounds again.

If the user turns the alarm off, a process disables the alarm and the app ends.

Waiting is shown in this flow chart both as a process (wait 2 minutes) and as a loop (time = alarm?). These are both valid techniques for showing the delay with the author of the flow chart able to choose the method they prefer to show this delay.



Now try this

An independent jeweller is setting up a website as a retail outlet. It would like to offer a loyalty discount for customers who purchased within the last year as well as giving a reduction of 10% on the sale for more than one item and a choice of postage rates of next day or economy. The jeweller is VAT registered, so VAT needs to be added to the sale after any discounts and postage.

Produce a flow chart to illustrate this algorithm.

There should be a symbol in the flow chart for each line of your pseudocode. Make sure you use the correct symbols.

Handling data within a program

Programming paradigms can be used to build computer code to handle data within a program. On this page, you will revise common data-handling techniques and structures provided within programming languages to process data.

Reading a data file

Code to read a data file usually needs an indefinite loop to repeat reading each line from the disk until it reaches the end of file.

A variable, `EmployersFile`, has been set to name the data file before the code here opens it for input. A variable, `Y`, is set to zero before the `Do` loop to keep track of each line read in from the

`EmployersFile` and which item in the array, `Employers()`, is used to store the input data.

A nested loop uses the variable, `X`, to keep track of the data for each employee and which item in the array is used to store the input data. This loop uses a `UCase()` function to force each data item to upper case.

After `EmployersFile` is closed variable, `Y`, is used to set variable, `LastEmployer`, so the program knows the subscript number of the last employee record in the array, `Employers()`.

```
FileOpen(1, EmployersFile, OpenMode.Input)
Y = 0
Do
  Y = Y + 1
  For X = 0 To 9
    Input(1, Employers(X, Y))
    TempEmployers(X, Y) = UCase(TempEmployers(X, Y))
  Next X
Loop Until (EOF(1))
FileClose(1)
LastEmployer = Y
```

Writing a data file

Code to write a data file can use a definite loop to repeat writing each line to the disk as the code knows how many records are in the array.

A variable, `NameOfFile`, is set to name the data file before the code here opens it for output.

This code uses a variable, `LineOfPrint`, which is built up into each line to be written to the disk with a comma between each data item.

Two `FOR...NEXT` loops are used to write to disk.

The first loop produces column headings for when the data file is opened into Excel with the name of the array, `Allocations(6-200)`, in the first column and numbers from the loop variable, `X`, for which array item is in the other columns.

The second loop writes the data to disk. After this loop, the data file is closed.

```
FileOpen(2, NameOfFile, OpenMode.Output)
LineOfPrint = "Allocations(6-200)"
PrintLine(2, LineOfPrint)
LineOfPrint = " ,"
Xvalue = 6
For X = 0 To Xvalue
  LineOfPrint = LineOfPrint & X & ","
Next X
PrintLine(2, LineOfPrint)
For Y = 0 To LastAllocation
  LineOfPrint = Y & ","
  For X = 0 To Xvalue
    LineOfPrint = LineOfPrint & Allocations(X, Y) & ","
  Next
  PrintLine(2, LineOfPrint)
Next Y
FileClose(2)
```

Now try this

Produce a program to read in a data file, make changes to the data and write it back to the disk. The data file can be created using Excel and saved as CSV (comma separated variables). Use a calculation in the spreadsheet to produce a reference number in the first column. The program you write can add 'REF' to these numbers before writing them back to disk. You can open the new data file in Excel to confirm the reference numbers have been set by your code.

Include a comma between each item when writing to disk.

Constants and variables

On this page, you will revise the data types you can use to define constants and variables.

Constants and variables

Constants and variables are very similar, with both naming a place in memory where data can be held. A constant does not change when code runs, the contents of a variable is usually changed by calculations or user input.

Arrays

An array is a variable which can contain many different values, each of these identified by the subscript (number) inside brackets at the end of the array name. A lot of code uses arrays to hold data records for the program.



For more on arrays, see page 23.

Text variables and constants

These can be combined (concatenated), searched or part of the string can be selected and used, such as the first three characters.

- **Alphanumeric strings** are used to hold combinations of letters from the alphabet and numbers, such as AB3076.
- A **character** is a single letter or number such as A, B, 6. A string is one or more characters. In a program, these variables can be used for any combination of words, spaces or numbers such as an address or a name.
- **Strings** can hold alphanumeric characters as well as other characters including escape codes such as CrLf (carriage return/line feed).

Numeric variables and constants

- **Floating point (real)** variables are used to contain numbers which may have a fractional part. These variables can hold a range of values which depends upon the type used. A single has a range of $-3.4028235E+38$ to $3.4028235E+38$, using 4 bytes of memory. A double is $\pm 1.79769313486231570E+308$ and uses 8 bytes of memory.
- **Integer** variables and constants are used to contain whole numbers. These variables can hold a range of values, which depends upon the type of integer used. A short integer has a range of -32768 to 32767 , using 2 bytes of memory. A long integer has a range of -2147483648 to 2147483647 and uses 4 bytes of memory.

Date/time

Declaring a variable as a date can save the programmer a lot of effort as there are functions available to calculate dates, such as DateAdd, and these variables can show the date in whatever format is required for the app.

A date variable can also be used for time. The actual content of date variable is a number with the whole part the date (number of days since 1 January 1900) and the fractional part the time, e.g. 6am is .25, midday is .5 and so on.

Boolean

A Boolean variable has only two possible values – true or false.

It is a good data type for use in a conditional statement, such as IF Found THEN, where Found is the Boolean variable.

Boolean variables can also be used to represent **objects**, such as option buttons in code.

Now try this

Write code which uses six different data types to hold information entered by the user. Process each of the entries using a method appropriate to the data type, for example, Boolean, to make a decision, using appropriate output to show the results of your processing.

Decide upon the data types with what you'll do with them before you start coding.

Managing variables

You need to revise the difference between local and global variables and when to use them, as well as the use of naming conventions to give meaningful names to objects in your code.

Managing variables

Managing variables helps to get the best performance from an app in terms of reliability, although there can be a very small reduction in speed due to the creations and releasings of **local variables**.

The minor speed hit is more than compensated for by the extra reliability due to much more control over where variables exists in the code. Good program design is very clear on where a variable is used or changed and so if another part of the code tries to use such a variable an error is generated to alert the programmer.

Global and local variables

The scope of a variable defines which parts of the code can see or use it.

A global variable exists everywhere in the code and only ceases when the app closes.

A local variable exists inside a subroutine or function subprogram whilst that code is running then ceases when the subprogram ends.

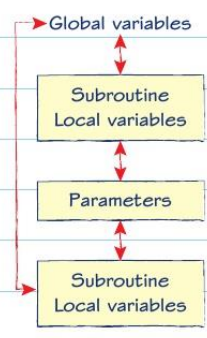
If a subroutine calls another subroutine, any local variables in the calling subprogram would not be seen by the called subroutine unless passed in as a parameter.

Parameters

A parameter is an argument in brackets after the name of a subroutine or function code passing a value into this code.

A parameter is a local variable to the subroutine or function unless it is defined by reference, in which case it is the same variable as was used by the code calling the subroutine or function.

The default for a parameter is by value, which means that what's inside a variable is passed into the subroutine or function and so does not affect this variable elsewhere.



Global can be used anywhere; local and parameters are private to the subroutine.

Naming conventions

The programmer has a lot of choice of the names of variables used in code, although there are a number of reserved words, such as open, which cannot be used as variables because they are part of the programming language.

A good variable name helps to document the code because it describes what the variable contains. Capitalisation can be used to help see words used in a variable name, e.g. CarColour.

Bad names are anything meaningless or which mislead about the use of the variable.

Variable names – dos and don'ts

- | | |
|--------------|--------------------------------|
| 👍 OKtoGo | 👎 oktogo (poor capitalisation) |
| 👍 VATdue | 👎 Var1 (meaningless) |
| 👍 NameOfFile | 👎 Axx (meaningless) |

Now try this

Create a poster showing how the scope of variables affects where they can be used in a program.

You can use circles to show each scope.

Arithmetic operations

Programming paradigms can be used to implement arithmetic operations, which include mathematical functions such as + and *, relational operators such as = and <, Boolean operators such as NOT, as well as date and time.

Mathematical operators

Mathematical operators are plus (+), minus (-), divide (/ or DIV) and multiply (*). Remember that the computer will always use BIDMAS (brackets, indices, divide, multiply, add, subtract) for the order in which a calculation is worked out.

The following calculation needs to set Pay by working out the HourRate + Supplement before multiplying by Hours, but gives a wrong result:

Pay = HourRate + Supplement * Hours

This is because brackets should be used to calculate the addition before multiplying:

Pay = (HourRate + Supplement) * Hours

Relational operators

Relational operators are frequently used in code, especially for conditions which control a branch into a choice of coding routes.

In these examples:

Pay has been set to 3.9

Cost has been set to 4

Equals	Pay = Cost	False
Less than	Pay < Cost	True
More than	Pay > Cost	False
Not equal to	Pay <> Cost	True
Less than or equal to	Pay <= Cost	True
More than or equal to	Pay >= Cost	False

Modulo

When a number is divided into another, the remainder (rem) is called the modulo or modulus, (MOD), which is often useful in calculations carried out by code needing the number remaining after division.

Modulo operator examples

10 % 3 returns 1 in Python code.
 7 mod 4 returns 3 in VB.NET code.
 = MOD(4, 3) returns 1 in an Excel cell.
 23 rem 4 returns 3 in Prolog code.

Boolean operators

Boolean operators can be complex calculations but always end with a result of True or False.

In these examples:

Car has been set to True

Diesel has been set to False

Opposite (NOT)	NOT Diesel	True
All of them (AND)	Car AND Diesel	False
Any of them (OR)	Car OR Diesel	True

Date/time operators

Usually a date in program code is held internally as a whole number (the day count from 1/1/1900) and time as the fractional part of a number, e.g. .75 is 6pm, so 6pm on 17 October 2017 is held as 43025.75, so simple arithmetic can often be used. Excel® has a known bug which calculates 1900 as a leap year.

Other programming languages make it very difficult for the programmer to reach the underlying numbers. It is much easier and practical to use the date and time functions provided.

Now try this

Create an Excel spreadsheet to show expressions illustrating mathematical, relational, Boolean and date/time operators. Copy and paste another version of each of your examples so a printed copy shows both the calculation workings and the result.

Use a single quote (') at the start of each of the copied examples so the workings print.

Arithmetic functions

Arithmetic functions enable you to code arithmetical operations – random, range, round, truncation – in a program. The Excel spreadsheet is used to demonstrate these arithmetic functions on this page.

Arithmetic functions

random()	Generates a random number.
range()	Creates an array of elements using the range of values in the brackets.
round()	Rounds a number up or down to the nearest whole number.
truncation()	Rounds a number down to the number of decimal places in the brackets.

Using the range() function

One argument will create a range of integer numbers from 0 to one before the argument, e.g. range(5) creates 0, 1, 2, 3, 4.

Two arguments create a range of integer numbers from first argument to one before the last, e.g. range(2,6) creates 2, 3, 4, 5.

Three arguments create a range of integer numbers with the last argument defining how much each item increments, e.g. range(1,12,3) creates 1, 4, 7, 10.

In code, **10 in range(1,4)** will return false.

ROUND() and TRUNCATE() functions

These are both used to specify the number of decimal places showing for a number.

The round() function will adjust a number to fit with the least significant digit rounded up or down.

The truncate() function simply removes any digits that do not fit.

Round and Trunc

	A	B
1	17.256	
2	17.26	17.25
3	=ROUND(A1,2)	=TRUNC(A1,2)

RAND and RANDBETWEEN

	A	B
1	0.322574	=RAND()
2	88	=RANDBETWEEN(1,100)

Using the random() function

The random() function will usually generate a random number larger than 0 and less than 1.

Some programming languages accept an argument inside the brackets to define the largest random number that can be produced.

Excel offers the RANDBETWEEN() function which accepts two arguments to define the scope of random numbers that are generated.

Now try this

Create an Excel spreadsheet to generate test data where the A column contains random numbers between 3 and 50, the B column a random date up to a year before today (assuming 365 days in the year) and the C column a random letter between A and Z.

Use the NOW() function as part of your calculation for the date.
Use the CHAR() function as part of your calculation for the letter.

String handling and general functions

String handling and other built-in general functions convert between different types of number and strings and perform general operations such as dealing with data files and printing.

String conversions

Converting to numeric allows code to use numbers stored as strings in calculations. CInt() converts to integer, CDbl() to double data type for floating point (float) numbers. CStr() can convert a number into string if there is need for searching or extracting part of the number.



Manipulating strings

- **Concatenation** is joining together two or more strings. The '+' character is used for concatenation by C, Java, Python, VB.NET among others whilst the '&' character is unique to Visual Basic. With numbers, '+' performs addition, but it concatenates strings.
- **Length** is how many characters are in a string. Many languages have a len() function to return this number.
- **Position** is where a character or group of characters are in a string. VB.NET uses the IndexOf method, Python the find method.

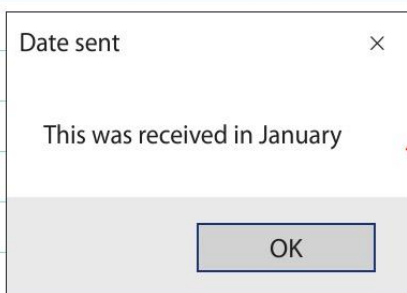
Making it work

The VB.NET code below uses concatenation to join text with month name extracted from date variable, DoR, after converting into a string variable, strDoR. DoR is formatted to "09 January 2017" (long date) before conversion, so number of characters in the month can be calculated as length of date string minus 8 (2 digits day, 4 year, 2 spaces around month).

Month position is calculated from 1 more than index of first space plus 2 (index starts at 0).

When this code runs, MsgText will contain "This was received in January"

```
DoR = "9/1/2017"
strDoR = CStr(Format(DoR, "long date"))
MonthLen = Len(strDoR) - 8
DoRpos = strDoR.IndexOf(" ") + 2
DoRmonth = Mid(strDoR, DoRpos, MonthLen)
MsgText = "This was received in "
MsgText = MsgText & DoRmonth
```



Input and open functions

Input lets users enter into a variable. This Python code shows a prompt of 'How many?', storing the response in Quantity:

```
Quantity = input("How many?")
```

Open connects code to a data file with an argument defining type of access, e.g. read, write. This Python example shows a file, Data.csv, being opened to read the data:

```
DataFile = open("Data.csv", "r" )
```

Range and print

Range is a function to return a range object. This Excel code example shows a calculation, =Rand(), being entered into a range of cells:

```
Range("A2:D12").Formula = "=Rand()"
```

Print sends text to screen or other output such as a data file. This Python code writes the contents of a variable, DataVar to a file, data.txt, opened in write mode as DataFile:

```
DataFile = open("data.txt", "w")
print(DataVar, file=DataFile)
```

Now try this

Create code to extract and use part of a date converted to string in a short sentence as described in the Making it work box above.

Use some form of console or message box output to check what is in the variables as you develop this code.

Validating data

Programming paradigms can be used to build effective validation techniques into code, improving the validity of inputs with post-check actions aiding further accuracy.

Validation check techniques

Checking data as it is entered for validity is a basic technique used in many ways to check the **data type** and **range** with any **constraints** before the code attempts to process the entry.

This screening helps to reduce errors and gives the user the opportunity to correct mistyping at the time of entering the data.

Data types and boolean

- Checking for the correct data type is basic validation preventing a lot of data entry errors, e.g. by rejecting text when a number is needed.
- Boolean logic can be applied to a data entry where an input could use a choice of validation rules, e.g. a vehicle registration could be in the form of XX99XXX or X999XXX.

Range

Many validation checks ensure inputs are within a range of values, such as age to make sure someone is not too young.

An age can be entered into a textbox with simple validation to ensure a number has been entered within an acceptable range, such as between 18 and 25.

A date of birth is much better data as this would still be useful for years after the data entry as an up-to-date age can be calculated from the current date obtained from the computer clock.

11 February 1998						
February 1998						
Mon	Tue	Wed	Thu	Fri	Sat	Sun
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	1
2	3	4	5	6	7	8

Constraints

Validation using constraints is essential for data entry such as a reference number with a clear structure. A reference number should be fixed with a set number and combination of letters and digits, e.g. STROO234, which are very straightforward to check. In this example the first three characters would be letters, the next five characters constrained to numbers and the overall number of characters must be eight.

Post-check actions

An app should include post-check actions which provide feedback to the user on why a validation check has failed and what the user needs to do to correct their entry:

- **Enforcement action** usually clears the bad data from the screen.
- **Advisory action** usually keeps the data but also sends a warning message.
- **Verification action** asks the user to confirm their data is correct.

Now try this

What validation can be applied to a data entry requiring a UK postcode?

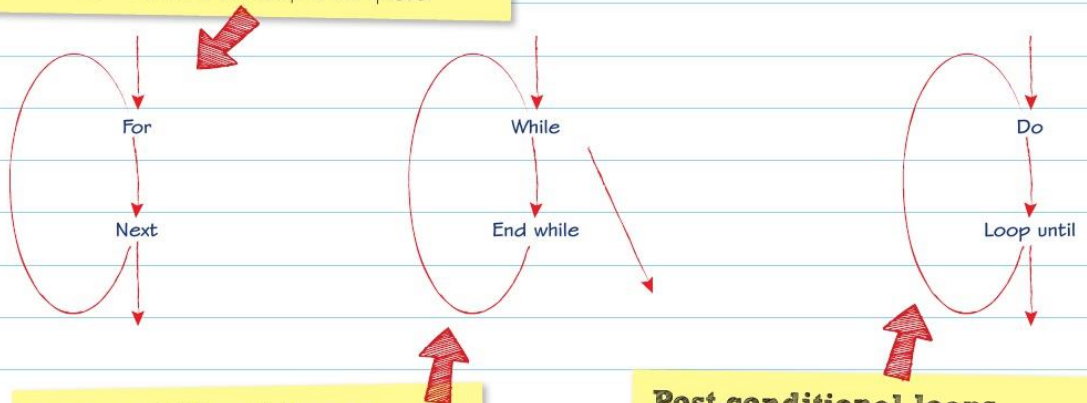
What positions do the letters and numbers occupy in a postcode? Are there any further techniques available?

Loops

Control structures include loops, also known as iterations, which repeat code as many times as needed. This page revises how to improve the effectiveness of code iterations by appropriate use of REPEAT, FOR, WHILE structures and any mechanisms needed to break out of them.

Unconditional loops

The classic unconditional loop is FOR...NEXT where a loop variable is used to keep count of the number of iterations with the NEXT line in this structure used to determine when the loop is complete.



Pre-conditional loops

Conditional loops will continue until an event occurs or a condition is met. The condition can be at the start, e.g. WHILE, which is known as a pre-conditioned loop, so the code inside the loop will not be run at all if the condition is not met when this structure is executed.

Post-conditional loops

If the condition is at the end, e.g. REPEAT UNTIL or LOOP UNTIL, it is a post-conditional loop so code inside the loop will run at least once, even if the condition is not met as the test is after the body of the loop.

These structures offer the programmer more control over how the loop will work.

Using loops

FOR...NEXT	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> looping through arrays <input checked="" type="checkbox"/> generating test data
WHILE	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> reading in data files
REPEAT UNTIL	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> checking for user attempts, e.g. passwords

Breaking out of a loop

When running a loop, the programming environment creates a structure which needs to end properly or there might be problems if the code runs for a long time.

Programming languages include commands such as BREAK, EXIT FOR or EXIT DO to finish a loop early.

Most code should not need to exit, as a conditional loop should respond to such situations. An unconditional loop requiring an early exit should probably be conditional.

Now try this

Produce a guide on appropriate uses for each loop type. Include examples of how the condition test for a conditional loop can best be used at the start or at the end of the loop.

Think of a situation where iteration code should not be run if a condition is not met.

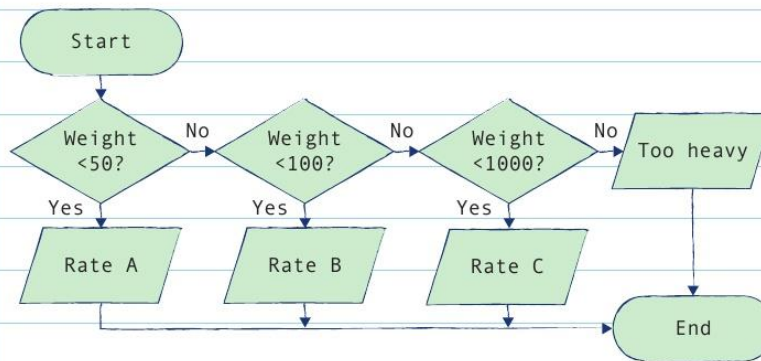
Branches

Branches allow you to make decisions within an algorithm. On this page, you will revise IF...THEN... ELSE...ELSEIF selections.

Branching with IF

The IF control structure allows codes to divide into separate pathways, selecting between two or more routes through the program. This structure starts with the IF...THEN line of code where a condition is evaluated as true or false. Code immediately after the IF...THEN line is run if the condition is true as far as the next part of this structure, which could be:

- ELSEIF to set another condition
- ELSE for code if the condition(s) not met
- ENDIF to complete the structure.



Postage rates

An app could be written to allocate a postage rate according to the weight of a shipment:

Weight	Rate
Below 50 g	A
50 g or more and below 100 g	B
100 g or more and below 1000 g	C
1000 g or more	Too heavy

The app will allow the user to type a weight into a text box, txtWeight, then show the appropriate rate on-screen.

The IF condition (number typed into WEIGHT by the user), shows Rate A if less than 50.

Care needs to be taken with conditions. The conditions here are carefully sequenced with first condition, (<50), so the next condition, (<100), is from 50 up to and not quite 100.

```

IF WEIGHT < 50 THEN
    SET POSTAGE LABEL TO "Rate A"
ELSEIF WEIGHT < 100 THEN
    SET POSTAGE LABEL TO "Rate B"
ELSEIF WEIGHT < 1000 THEN
    SET POSTAGE LABEL TO "Rate C"
ELSE
    SET POSTAGE LABEL TO "Too heavy"
ENDIF
  
```

```

If txtWeight.Text < 50 Then
    lblPostage.Text = "Rate A"
ElseIf txtWeight.Text < 100 Then
    lblPostage.Text = "Rate B"
ElseIf txtWeight.Text < 1000 Then
    lblPostage.Text = "Rate C"
Else
    lblPostage.Text = "Too heavy"
End If
  
```

ELSEIF statements respond to other weights with ELSE line running code not met by any other condition showing "Too heavy".

Now try this

Write a program which accepts (and validates) user input of a whole number between 0 and 48 to represent the points achieved for a test. Your program will use a select case structure to show 'Fail' (0-17), 'Pass' (18-25), 'Merit' (26-41), 'Distinction' (42-47) or 'Distinction*' (48) according to the input value.

Be very careful to code for the grade boundaries and use test data to ensure they are met.

Had a look Nearly there Nailed it!

Function calls

Programming paradigm control structures include function calls. A function is a sub-program that returns a value when it is called. This page revises how to define a function and declare any arguments, as well as how to call the function.

Functions

The **definition** needs the keyword **FUNCTION** followed by its name. **Arguments** can be used in a similar way to sub-routines as parameters inside brackets after the function name.

There needs to be a line of code inside the function where the return value is defined.

To call a function, either a variable is set to the function or the function name is used in a place where the value it returns can be used.

Function example code

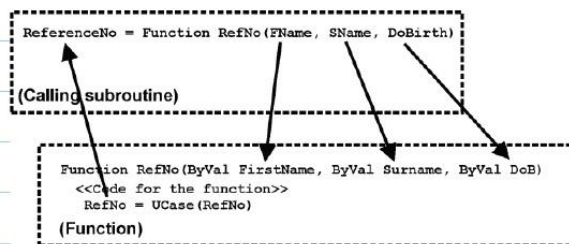
The function code below takes parameters and processes them into a reference number from the first two characters of FirstName, Surname, DoB followed by last two characters of FirstName, Surname (DoB needs to be converted into a string before the first two numbers can be used). RefNo is converted into upper case before being returned by this function.

```
Function RefNo(ByVal FirstName, ByVal Surname, ByVal DoB)
    RefNo = FirstName.substring(0, 2)
    RefNo = RefNo + Surname.substring(0, 2)
    RefNo = RefNo + CStr(DoB).Substring(0, 2)
    RefNo = RefNo + FirstName.substring(Len(FirstName) -2, 2)
    RefNo = RefNo + Surname.substring(Len(Surname) -2, 2)
    RefNo = UCase(RefNo)
End Function
```

These parameters are ByVal meaning that their values can be used inside the function as local variables and will not change anything else in the program.

The following line of code calls this function:

```
Reference = RefNo(txtFname.Text, txtSurname.Text, dtpDoB.Text)
```



Parameters

In the code above, FirstName, Surname and DoB are all parameters passed in by the values of text boxes and a date picker on the form. These are all used as local variables by the function.

This shows how the parameters are passed into the function which then returns the value in RefNo.

Now try this

Write a program or create a spreadsheet which has a function that creates a reference number from first name, last name and date of birth entered into a form or spreadsheet cells.

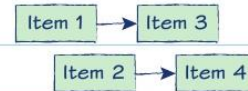
Prepare some test data to confirm your function works as expected.

Lists

Data structures enable you to store and process data in a computer program. A linked list is a data structure using pointers to link to the next item. A data set can have several lists.

Linked lists

These are used where a program needs to be able to follow a sequence between items of data which may be of different types.



Linked lists

The example below has two pointers before each item, first location, second job role. The end of each list is -1 (11111111 in binary).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	07	08	D	A	V	I	D	OE	1E	H	E	L	E	N	2A	16
1	A	B	D	I	L	1D	24	T	A	N	G	E	L	32	2B	M
2	U	I	N	44	3E	S	T	E	V	E	3D	33	J	O	S	H
3	U	A	49	4A	E	L	I	Z	A	B	E	T	H	-1	45	H
4	A	R	R	Y	-1	-1	J	O	E	-1	-1	B	R	I	A	N

Location start pointers are 00 (Bristol), 15 (London), 23 (York) with these teams: Bristol (David, Helen, Abdil, Joshua, Harry), London (Tangel, Muin, Elizabeth, Brian), York (Steve, Joe).

Job role start pointers are 01 (Sales), 0F (Support) with Sales (David, Helen, Muin, Joshua, Elizabeth, Brian), Support (Abdil, Tangel, Steve, Harry, Joe).

Code to define a list

This code creates a list, adds some items then loops through the list to show the items.

```

Dim lstNames As New List(Of String)
lstNames.Add("Mo")
lstNames.Add("Sarah")
lstNames.Add("Sam")

lstNames.Sort()

For Each item In lstNames
    MsgBox(item)
Next
  
```

How the code works

The code uses the sort method to change the order of data items inside the list into an alphabetical sequence.

An iteration loops through the items in the list, using a message box to show each.

The variable, `lstNames`, is defined as a list of the string data type. Three items are added to the list, "Mo", "Sarah", "Sam".

Now try this

Create some test data consisting of six sales people, each working from one of two offices and whether they have a driving licence. Copy this data onto squared paper using pointers for the office and driving licence.

You can print squared paper from Excel using borders and row/widths adjusted to create square cells.

Had a look Nearly there Nailed it!

Arrays

The array is a data structure widely used in code as a variable containing several data items.

Variables and arrays

A variable is a place in memory used by code to hold an item of data. Variables are named (declared) in code by the programmer, usually defined for a particular type of data.

An array is a variable with many places for data so different items can be held. Brackets at end of the array name enclose subscript(s), used to identify the current item in the array.

1D array	2D array			
97	22	44	41	63
79	35	11	20	6
68	96	13	61	90
11	49	31	36	36
39	99			59
7	26			85
70	17	59	20	50
51	77	49	1	88

One-dimensional (1D) arrays

A one-dimensional array has a single subscript inside the brackets and can be thought of as a single list of items.

An array can store only one type of data but the data can be of any type including integers, characters and strings.

Arrays can work very well with loops. A loop variable can be used as the subscript so each item in the array can be used in comparisons or other operations.

TestData(0)	23.8739745010
TestData(1)	24.0826482235
TestData(2)	24.6732391543
TestData(3)	25.0122885583
TestData(4)	25.3637706381
TestData(5)	25.8537892214
TestData(6)	25.8946307915
TestData(7)	26.7077571746

The data above shows an example of how an array could be used to hold data streamed into the computer from a test device.

Two-dimensional (2D) arrays

A two-dimensional array is multi-dimensional as it has two subscripts inside the brackets and can be thought of as able to hold a table of data. The example data, right, shows how an array could be used to look up postage rates.

PostageRate(2,1) holds £7.45, the rate for 100g with £1000 compensation (cover). The subscript used for rows and subscript used for columns is the choice of the programmer and makes no difference as long as the program code is consistent in how these are used.

```
Cover = 2
Weight = 4
lblRate.Text = PostageRate(Cover, Weight)
```

The 0 subscript in both dimensions is used to hold a title for that set of data, e.g. row 3 holds the rates for up to 1kg.

Postage rates	0	1	2	3
0		£500	£1000	£2500
1	100g	£6.45	£7.45	£9.45
2	500g	£7.25	£8.25	£10.25
3	1kg	£8.55	£9.55	£11.55
4	2kg	£11.00	£12.00	£14.00
5	10kg	£26.60	£27.60	£29.60
6	20kg	£41.20	£42.20	£44.20

Now try this

Produce code that uses a 2D array to hold a set of postage rates for different weight ranges and insured amounts. The user should be able to enter a weight, select the insurance and see the postage cost.

Use the data shown on this page to populate your array.

Records

A record is a data structure used to hold an item such as a product or person, often in a database table. The record is made from fields, each of which is defined to an appropriate data type. For example, a product description would be text, quantity as number and so on.

Fields

Fields are columns in a table object shown in the datasheet view, with each holding an item of data such as a date for each record.

LRef	Date out	Date in	GameRef	MemberRef	Returned?
1	08/11/2016	17/11/2016	PC0287	BE02	<input checked="" type="checkbox"/>
2	09/11/2016		PC1945	NE03	<input type="checkbox"/>
3	09/11/2016		WI0452	MO12	<input type="checkbox"/>

These fields are set up in the design view of a database. The screenshot, right, shows some of the data types available for fields including AutoNumber to increment an integer with each new record, Date/Time to show a date-picker and hold dates, Short Text for most data entries and Yes/No.

Field Name	Data Type
LRef	AutoNumber
Date out	Date/Time
Date in	Date/Time
GameRef	Short Text
MemberRef	Short Text
Returned?	Yes/No



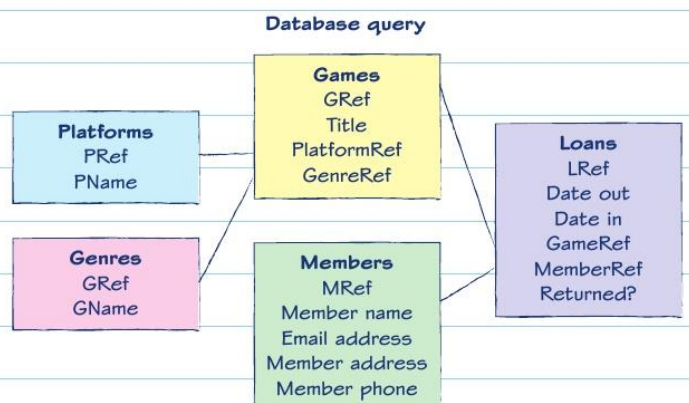
Console games club objects

A club has been formed by a group of friends to loan each other games for their consoles. The number in the group has expanded to the point where it has been agreed to create a database to keep track of who has what. The diagram below represents how the various tables of a database and their fields can be linked using a query.

Using records in a computer program

A database query can join tables together into a data set which consists of any mix of the table fields and can contain records selected by criteria to be used by forms and reports.

A computer program can carry out similar operations on data using arrays or data objects to hold the data using XML (eXtensible Markup Language) as data source document formats.



Now try this

Create a database for the console games club described on this page.

Use the table and field names shown in the ERD on this page.

Had a look Nearly there Nailed it!

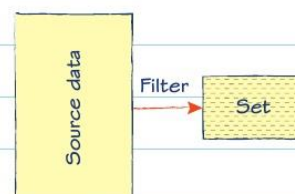
Sets

A set is a collection of records in a database or other data source that could be made from some of a table, filtering or a query data set.

Sets of data

A set consists of some data that has been brought in from a structure such as a table in a database, an array in a program or records that match a criterion.

The set will meet a particular need, such as clients with a policy due for renewal, students who failed to submit work on time or any other useful collection of data.



Queries and data sets

A database uses queries to produce data sets. A data set is often temporary and used for a purpose such as providing the data for a report.

If a query is used for a report, the data set is generated when the report runs but not kept, so if the same report is run again the query will generate a fresh data set based on data currently held in the records.

Database queries can bring in data from several tables and apply criteria to these records to produce a data set that exactly matches what is required for a report.

Data sets like this exist on many car trading websites, with the website users being able to filter results by such categories 'make' and 'price'. A visitor to the website would be able to see which vehicles are within their budget.

Set of cars selected by brand, sorted by price

Bugatti EB110 Dauer	£894,950
2002 11,200 miles Manual Petrol	
Bugatti Veyron 8.0 2dr	£935,000
2008 15,000 miles Automatic Petrol	
Bugatti EB110 Dauer	£950,000
2003 2,500 miles Manual Petrol	
Bugatti Veyron 8.0 2dr	£975,000
2010 13,500 miles Automatic Petrol	
Bugatti Veyron 8.0 2dr	£1,200,000
2011 22,750 miles Manual Petrol	
Bugatti Veyron 8.0 2dr	£1,275,000
2012 11,000 miles Automatic Petrol	

Examples of data sets

Title	Genre	Developer(s)
Afro Samurai	Action	Namco Bandai Games
Air Conflicts: Secret Wars	Action	Games Farm
Air Conflicts: Vietnam	Action	Kalypso Media
Amazing Island	Action	Ancient
Anarcute	Action	AnarTeam
Animaniacs: The Great Edgar Hunt	Action	Warthog Games
Aquaman: Battle for Atlantis	Action	Lucky Chicken Games
Arslan: The Warriors of Legend	Action	Koei Tecmo
Arslan: The Warriors of Legend	Action	Koei Tecmo
Asterix & Obelix XXLPAL	Action	Étranges Libellules
Attack on Titan	Action	Omega Force
Avatar: The Last Airbender	Action	THQ

Sets of data can be produced by databases and by apps processing some or all the data present into just that which is wanted.

This data set is from a database of games available for consoles and computers, selecting the data for action games. This would be useful for a customer looking to purchase such a product.

Now try this

What examples of sets can you identify in the learner data held by your centre?

Consider the information that is kept about you and your achievements.

Bubble sort

Sorting can be an essential technique when processing data into information. The most commonly used sorting algorithms used to sequence data are the bubble sort, quick sort and insertion sort. On this page, you will revise the bubble sort, which is a simple, basic algorithm.

Benefits	Limitations
<ul style="list-style-type: none"> 👍 Simple to understand 👍 Easy to implement 👍 Only needs a small amount of RAM 👍 Stable and reliable 	<ul style="list-style-type: none"> 👎 Slow 👎 Very poor with large data sets 👎 Old algorithm

How it works

The bubble sort works by starting with the first item which is passed up the data set by swapping with the next item, until a larger item is met, then the larger item is passed up and so on. At the end of the first pass, the highest item will be at the end.

This process keeps on repeating until all the data is sorted into sequence.



Bubble sort algorithm

DO...LOOP WHILE repeats the code until there is nothing left to sort using Switch which is set to false at the start of the iteration and to true when an item is sorted into a new position.

```

Do
  Switch = False
  For L1 = First To Last-1
    If A(L1)>A(L1 + 1) Then
      Tmp = A(L1)
      A(L1) = A(L1 + 1)
      A(L1 + 1) = Tmp
      Switch = True
    End If
  Next L1
Loop While Switch
  
```

The FOR...NEXT loop starts at the beginning of the data, ending at the item before last. Each time through this loop the current item is compared to the next and if the current item is larger, they swap. During the first iteration the largest item, 75 below, is moved to the end, like a bubble rising to the top in a glass. Further repeats of the FOR...NEXT loop inside the DO...LOOP WHILE structure bubble each next highest items, 27, 16, to their correct places.

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)
27	16	75	12	31	32
16	27	75	12	31	32
16	27	12	75	31	32
16	27	12	31	75	32
16	27	12	31	32	75
16	12	27	31	32	75
12	16	27	31	32	75

Now try this

- (a) Enter six values in the first row of a spreadsheet.
- (b) Copy this data into the second row and reorder it to show one pass of the bubble sort algorithm.
- (c) Repeat this process until all the values are sorted.

Use the pseudocode for guidance.

Had a look Nearly there Nailed it!

Quick sort

The quick sort is a popular sorting algorithm used to sequence data.

Benefits	Limitations
👍 Good at sorting large data sets	👎 Not efficient at sorting very jumbled data sets
👍 Good at mostly sorting data	👎 Difficult to write
👍 Widely used and popular	👎 Not perfectly stable because equal items might not stay in same sequence
👍 Concise code using sophisticated recursions	
👍 Does not need much temporary memory	
👍 Can work well with parallel processing	

How it works

The algorithm selects a pivot in the data, then loops until items from left of the data set are all below the pivot and items from right all above the pivot. The pivot item can then be copied to its place before recursing the sort subroutine to do it all again for another pivot.

This is like sorting a pile of paper into two heaps, one heap less than an arbitrary value (pivot), the other more. Each of these heaps can then be further divided and sorted then put back together into a sorted whole.



Quick sort algorithm

Low is the lowest item in the array, High the highest item.

The WHILE High>Low loop iterates until the sort is completed.

WHILE L1<L2 AND A(L1)<=Pivot loop finds where from the left of the data set Pivot should be. The item here is copied to the low place and replaced by the Pivot.

```

Sub QuickSort(Low, High)
  While High > Low
    L1 = Low
    L2 = High
    Pivot = A(Low)
    While L1 < L2
      While A(L2) > Pivot
        L2 = L2 - 1
      End While
      A(L1) = A(L2)
      While L1 < L2 And A(L1) <= Pivot
        L1 = L1 + 1
      End While
      A(L2) = A(L1)
    End While
    A(L1) = Pivot
    Call QuickSort(Low, L1 - 1)
    Low = L1 + 1
  End While
End Sub
  
```

Pivot is set to a point in the data.

The WHILE A(L2)>Pivot loop finds where from the right of the data set Pivot should be. The low item is copied to this place.

The WHILE L1<L2 loop is used to iterate until the sort is completed.

The recursion takes place with the quicksort sub-routine called from itself to sort what is left in the data set. The data examples shown here highlight in green the range of the data set that are sorted with each quicksort recursion with the Pivot value for each recursion in the right-most column of this data. The first pivot, 93, is placed at the end as there are no greater values. When 42 is the pivot, 55 is moved next to 93 ready for 42 to be copied to its place. When 21 is pivot, 13 is moved so 21 can be copied to its place completing the sort.

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	Pivot
93	18	55	13	21	42	93
42	18	55	13	21	93	42
21	18	55	13	21	93	42
21	18	55	13	55	93	42
21	18	13	13	55	93	42
21	18	13	42	55	93	21
13	18	21	42	55	93	21

Now try this

- Enter six values in the first row of a spreadsheet.
- Copy this data into the second row and re-order it to show one pass of the quick sort.
- Repeat this process until all the values are sorted.

Use the pseudocode for guidance.

Insertion sort

One of the most common algorithms used to sequence data is the insertion sort.

Benefits	Limitations
<ul style="list-style-type: none"> 👍 Algorithm well understood 👍 Simple to code 👍 Sort stable so existing matches kept in same sequence 👍 Memory efficient 👍 Good with sequential data 👍 Good when data almost completely sorted 	<ul style="list-style-type: none"> 👎 Can have performance issues when used with slow backing storage 👎 Poor performance with large lists

How it works

The insertion sort has been known and understood for decades, modestly improving the bubble sort with better performance.

Code starts from the low end, checking each pair of items to ensure the later item is larger than the previous one.

When an item is found to be out of place, all the items after the low end are moved back so the found item can be inserted at the beginning of this data set.

The process then re-starts to find the next item that is in the wrong place so it can be inserted into its correct position.



Insertion sort algorithm

The FOR...NEXT using L1 as loop variable iterates for each item in the data set with Tmp holding the current item.

```

For L1 = 1 To Last
  Tmp = A(L1)
  For L2 = L1 To 1 Step -1
    If A(L2 - 1) > Tmp Then
      A(L2) = A(L2 - 1)
    Else
      Exit For
    End If
  Next L2
  A(L2) = Tmp
Next L1
    
```

The FOR...NEXT using L2 as loop variable starts with an IF statement to find when an item in the array is larger than the next item at which point it iterates from the current L1 item to the start moving data items back one place then Tmp is put into where the lowest of the data items was moved.

The lowest item of each iteration is inserted into the correct place with items shuffled up to make space for the sorted item. In the example below, 8 is inserted into the first position with items after moved to their next place up to where the 8 was. The other insertions were 39 then 40.

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)
9	16	68	8	39	40
8	9	16	68	39	40
8	9	16	39	68	40
8	9	16	39	40	68

Now try this

- (a) Enter six values in the first row of a spreadsheet.
- (b) Copy this data into the second row and re-order it to show one pass of the insertion sort algorithm.
- (c) Repeat this process until all the values are sorted.

Use the pseudocode for guidance.

Searching

Searching is a typical use for a computer system holding massive amounts of data which can be accurately checked for any items that match the search criteria. There are two basic types of searching algorithm: serial/linear and binary. On this page, you will also revise two other common algorithms, to count occurrences and for input validation.

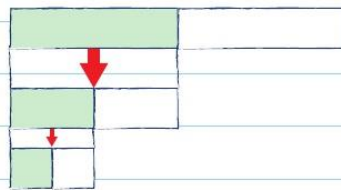
Serial/linear search

This is a very simple algorithm, which starts at the beginning of the data, then looks at each item until a match is found. This is a good way to **count the occurrences** that match criteria in a data set as well as the best search option for data that has not been sorted into a sequence.

- ✓ best for unstructured data
- ✓ very simple to program
- ✓ works with any data set
- ✓ easily counts number of occurrences
- ✓ works with any combination of criteria



Serial searches each item until found.



Binary keeps dividing data until found.

Binary search

This algorithm can be used with data which has a sequenced order. The code starts by looking at the mid-point of the data to decide if the item matching the search criteria is before or after this item. The mid-point of the half of data containing the item is then looked at to decide if the item matching the search criteria is before or after. This is repeated until the item matching the search criteria is found.

- ✓ best algorithm for sorted data set
- ✓ very quick
- ✓ works with very large data sets
- ✓ programming algorithms are well known

Count occurrences

This is almost the same algorithm as for a linear search, except that it counts how many are found, rather than returning items matching search criteria.

Input validation

Validation of search criteria is essential to reduce wasted search time from bad data entries.

Now try this

- 1 (a) Create a spreadsheet model to calculate the lowest and highest number of comparisons needed to find an item using a linear and also using a binary search.
(b) How many comparisons are needed for 100/10 000/1 000 000-sized data sets using these methods?
- 2 Is there an obvious choice of algorithm for these searches? Why?

Remember the binary search starts half way through the data set, then keeps on halving until only the search item remains.

Using stacks and queues

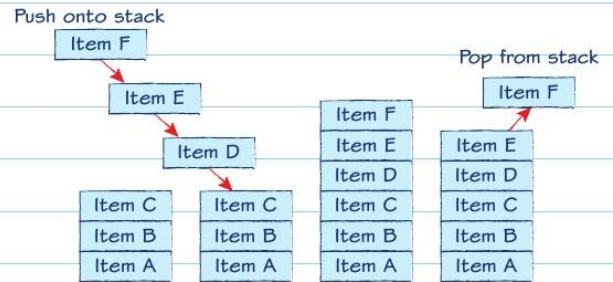
Stacks and queues are different techniques used to keep data items temporarily when a program is running. A stack brings back the last item (LIFO) whereas a queue retrieves the first item (FIFO).

Stacks

A stack is a last in first out (**LIFO**) structure, rather like a pile of plates which are stacked with the top plate (the last one added to the stack) being the first to be removed from the stack.

Items are pushed onto the stack to store and popped off the stack when retrieved.

Stacks are used many thousands of times every second when code runs on a computer for subroutine return addresses.



Queues

A queue is a first in first out (**FIFO**) structure, similar to a tunnel or conveyor belt with the first item written (added) to the queue being the first to be read from the queue.

Queues are used in many places where items are buffered on a computer system to wait for processing.

Examples of stacks

- Subroutines and functions use a stack to remember the address of the line of code that called them, which is used as the return address for when the subroutine or function completes.
- Spreadsheets use a stack to keep the part answers when calculations are worked out following BIDMAS. For example, a multiplication part answer would be pushed onto the stack before being popped off to add to another part answer.

Examples of queues

- Operating systems use queues for printing, with the first sent to the printer being the first to get printed.
- Computers use a keyboard buffer between the keyboard and the operating system so if the keyboard is used very quickly, the keys are read into the computer in the order they were typed.

Now try this

Create a paper-based story board and/or an animation to illustrate how program sub-routines use stacks for their return addresses.

You can use an example of a subroutine calling another subroutine.

Had a look Nearly there Nailed it!

Procedural programming structure

Procedural programming is the traditional type of programming language used to produce code that starts from one point then follows routes through the code, calling procedures and functions as needed, to help structure the code.

Procedural programming structures

Structure	Definition
Statement	Line of code
Block	Collection of statements
Procedure (sub-routine)	Named sub-program that can be called from other places in the code that returns when completed to where it was called from
Function	A procedure that returns a value

Using procedural programming structure

Good practice	Poor practice
<ul style="list-style-type: none"> Use sensible procedure and function names making self-documenting code. Use arguments to pass values into procedures and functions. 	<ul style="list-style-type: none"> Copy and paste the same code into several places in the program. Use procedures to work out values that are returned in global variables.

Evaluating code

The code shown below has examples of both good and bad programming practice.

```

For Y = 0 To 20
    SB(Y) = ""
Next Y

'Read Dfile into P() array
    lblProgress.Text="Reading data"
FileOpen(1,Dfile,OpenMode.Input)
Y = 1
Do
    Input(1, S(0, Y))
    P(0, Y) = UCase(S(0, Y))
    If Not (EOF(1)) Then
        Input(1, P(1, Y))
    End If
    Y = Y + 1
Loop Until EOF(1)
    
```

- The FOR Y loop used to set initial values for array is good. Poor practice would be 21 lines of code to do this instead of the loop structure.
- SB(), P() are poor naming, as they are meaningless. Better practice would be names such as ShoppingBasket(), Products().
- Y is a good name for the loop variable as it is used to loop around the array Y subscript.
- The comment ('Read Dfile...') is good practice as this helps to document the code.
- Setting label, lblProgress.Text, to let the user know what the code is doing is good practice.
- Using a variable, Dfile, for the data file is good practice as this an appropriate name. Using a variable makes the code much more readable than the full file and folder names.
- Using the Ucase() function is good practice to make the data consistent in the program.
- IF NOT branch is poor practice as it handles an unexpected end of data file showing the data structures have not been fully planned.

Now try this

Interpret the code given and explain the product of this function. Suggest ways in which it could be improved.

```

Function Letters(ByVal Word1)
    Dim X As Integer
    Dim Word2 As String
    Word2 = ""
    For X = 1 To Len(Word1)
        Word2= Mid(Word1, X, 1) & Word2
    Next
    Letters = Word2
End Function
    
```

You can use a FOR loop to iterate the number of characters (LEN) in the word.

Procedural programming control structures

Within procedural programming, control structures define the sequences and routes available to run through code with branching to respond to when the conditions are met and iteration to repeat sections of code.

Control structure	When to use it
Sequence	Everywhere apart from lines of code used to define part of a different control structure
Condition	Whenever a decision needs to be made on which path is to be taken through the code
Iteration	When a section of code needs to be re-run

Sequential code

```
Call InitialiseVariables()
StudentID File =
"Choices Batch D.csv"
Employer File =
"Employer Batch D.csv"
Sessions Lot = PMslots
BatchLetter = "D"
Call ReadData()
```

This is the default structure, simply lines of code which follow each other. The code starts to run with the first line in the program, then the second line and so on.

Conditional code

```
If chkUseReserve.Checked = True Then
    lblReserveTrigger.Visible = True
    txtReserveTrigger.Visible = True
Else
    lblReserveTrigger.Visible = False
    txtReserveTrigger.Visible = False
End If
```

Conditional code takes one of two routes through the code with the condition used to determine which of the routes is taken.

A condition is anything that works out as true or false. In the code example here, the condition is whether the checkbox object, `chkUseReserve`, has been selected.

Iterative code

```
For Z = 0 To NoCols
    Temp = Stock(Z, I)
    Stock(Z, I) = Stock(Z, I + 1)
    Stock(Z, I + 1) = Temp
Next Z
```

Iterative code is often called a loop as it repeats a section of code by looping around it.

The code example here shows a FOR...NEXT loop which starts at 0 and continues for as many times as are in the variable, `NoCols`.

The loop variable, `Z`, starts at 0 and increments with each loop, so `Z` will contain 1 on the next iteration, then 2 and so on until `Z` reaches the value held in `NoCols` when the loop completes.

Types of conditional code

- IF...THEN...ELSE is a simple structure which offers two routes, according to whether the condition between IF...THEN is true or false.
- SELECT CASE is a more complex structure offering as many branches as the program needs. This structure lists as many conditions as needed with the code to be run for each.

Types of iterative code

- Definite loops have a set number of iterations, such as FOR...NEXT where the number of loops is defined in the FOR line.
- Indefinite loops repeat until a condition is met, such as DO...LOOP UNTIL. The condition can be at the end of the structure so the loop code runs at least once. If the condition is at the start the loop code will not run at all if the condition is false when the program runs.

Now try this

Produce code using appropriate procedural structures to accept ten numbers, loop through to add them together, find the mean average and to then find further averages of those values above or below the mean. Step through your code to see variable contents as the averages are calculated.

Put a breakpoint in your code to pause so you can see how they change using an appropriate tool, such as a watch window or hovering pointer over variables in the code.

Had a look Nearly there Nailed it!

Object-oriented programming structure

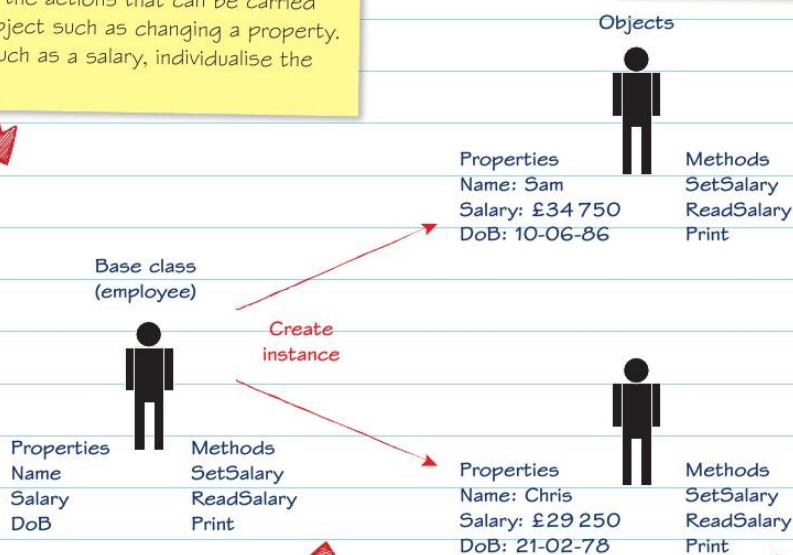
Object-oriented programming is a type of programming language using objects with properties to define what they are and methods to define what can be done with them.

Classes

A class is the starting point for any object, giving the object its characteristics. You can think of a class as the template defining the methods and properties for a new object.

- Methods are the actions that can be carried out on the object such as changing a property.
- Properties, such as a salary, individualise the object.

An example of object-orientated programming structure.



Instances

A new object will be an instance of a class. There can be many instances of a class, each of which is an object. In the example on this page the employee class can be used to create several instances, each of which is an employee object.

Objects

The objects can be used in your code according to their methods and properties. Objects such as buttons and text boxes are also available in modern visual languages that you can use to construct the interface. Dragging a new button from a toolbox onto a form creates a new instance of the button class.

Now try this

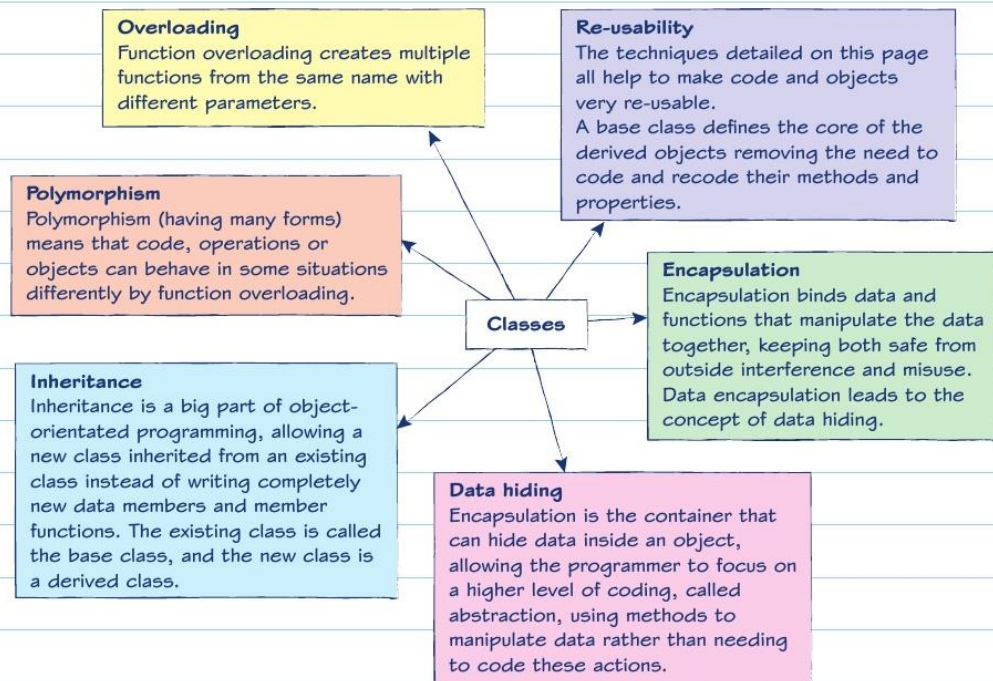
Complete the code or produce your own code to define an invoice class to be used by a form including a method definition for output and data for Name, Amount, Date, Customer.

```
class Invoice
{
public:
}
```

You will need to create a new instance of the class.

Object-oriented programming features

Object-oriented programming can be used with a range of techniques – inheritance, encapsulation, polymorphism and overloading, data hiding and reusability – to provide a lot of flexibility on how an object can behave.



Common object-oriented programming languages include:

- C++
- Microsoft® C#
- Oracle® JavaScript
- PHP
- Ada.

Now try this

Modify the DrawSquare method to enable a rectangle to be drawn if two parameters are provided. Overload this class to accept just one length.

```
Public Class DrawShape
    Public Sub Shape(sender, X, Y, ShL, ShH)
        Dim myBrush As New System.Drawing.SolidBrush(System.Drawing.Color.Red)
        Dim formGraphics As System.Drawing.Graphics
        formGraphics = sender.CreateGraphics()
        formGraphics.FillRectangle(myBrush, New Rectangle(X, Y, ShL, ShH))
        myBrush.Dispose()
        formGraphics.Dispose()
    End Sub
End Class
```

Had a look Nearly there Nailed it!

Event-driven programming structure

Event-driven programming produces code that responds to events such as a mouse being clicked on a button or another object.

Event-driven programs and sub-routines

- Modern event-driven programs mostly use forms for the user interface, with objects on the forms such as buttons which generate events that start code in the sub-routine which responds to the event.
- Sub-routines can be written as code modules by the programmer to structure their code. Sub-routine modules are also generated for responding to each event a programmer needs to code, usually by clicking on an object then selecting the appropriate event.

Main loop

Programmers do not usually code for the main loop, as this is provided by the programming environment to continuously look for an event to occur, at which point the event handler sub-routine code is called to respond to the event. This loop recognises which event to call the appropriate code, for example the mouse click event handler for a button which has just received a click from the user.

Event-driven programming offers a wide and varied choice of events so the coder can choose the exact event that is required.

Callback functions

The purpose of a callback function is to carry out a task in the background and then to report back to the program when the task is completed.

This could be compared with shopping. A normal function is like going to the supermarket, filling the trolley and paying at the checkout, which needs your full attention. A callback function can be likened to online shopping where the order can be quickly placed and you can do other tasks until the order arrives.

Event handler code

The sub-routine below is code to handle the `CheckedChanged` event of a checkbox object, `chkUseReserve`. This code would usually be triggered when the mouse clicks on the object to check or uncheck, but would also be invoked if the object checked property was changed from another sub-routine.

The code uses an IF selection structure to branch through one of two different routes according to whether this checkbox has changed to checked or unchecked.

```
Private Sub chkUseReserve_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles _chkUseReserve.CheckedChanged
```

```
    If chkUseReserve.Checked = True Then
        lblReserveTrigger.Visible = True
        txtReserveTrigger.Visible = True
    Else
        lblReserveTrigger.Visible = False
        txtReserveTrigger.Visible = False
    End If
End Sub
```

This code responds to the `CheckedChanged` event, so it runs when the checkbox is set or cleared.

Now try this

Write code for appropriate events to show a 'Rollover' effect on an object on a form, by changing the background colour and counting how many times the mouse moves over the object.

Create your objects first then select the events you want to code.

Event-driven programming features

Event-driven programs are used extensively in modern apps, which depend upon mouse clicks or a touch screen. Events can also be triggered by objects such as a timer to produce regular events that occur alongside other events such as a mouse click.

Service-orientated processing

Service-orientated processing is the breaking down of complex problems into a collection of separate (but potentially linked) processes, each providing a specific service for client applications.

These programs often work in the background. For example, a database can run as a service on a server listening for requests and providing responses.

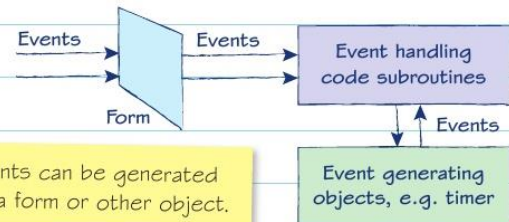
Service-orientated processes can establish connections with each other using messages passing across a network.

These processes respond to events to produce the actions needed to provide the required service.

Events, handlers and loops

An event can be almost anything that happens to an app when it runs. As well as obvious events such as a mouse click, there can be subtle events such as key down, key up, key press to offer the programmer a lot of control over which trigger function to use to start the event handler running.

Event loops keep spinning around, waiting for an event to occur so they can call the appropriate **event handler** to run code.

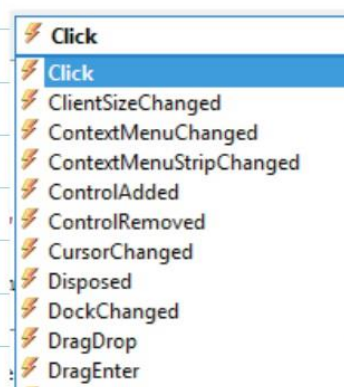


Time-driven events

A timer object can be used to generate time-driven events which occur at regular intervals.

The coding behind a clock, stopwatch or similar will use a timer object to generate the regular events needed to update the display.

A timer object can be defined in code with how long between the events it generates.



Trigger functions

A trigger function is there to respond to events which the code encounters, triggering an event handler code sub-routine.

The programmer has a wide choice of which of these to select from inside the programming environment.

The screenshot, left, shows just a few of the trigger events available for an object on a form in a VB.NET program. When the programmer clicks on one of these triggers, the programming environment starts up a new sub-routine to handle that event.

The coder can then enter the code they want to run for this event trigger.

Now try this

Use a timer object and code to demonstrate time-driven events and trigger functions by showing labels on a form counting seconds and minutes.

Use meaningful names for your objects.

Coding for the web: Characteristics

Mark-up and web languages are used to create websites and to provide extra functionality such as live calculations. HTML is the mark-up standard.

Web mark-up languages

There is a choice of web mark-up languages to produce code for a website to run in your browser, each with its own characteristics, features, **performance** and power.

A modern web mark-up language needs to meet the standards of HTML5 as this is the expected environment a browser will use to display and run the webpage.

Platform independence and security

A web mark-up language needs to be stable and to run on:

- different browsers such as Safari, Chrome
- different hardware such as mobile phones, tablets, computers and smart televisions.

Web mark-up languages should be able handle secure transactions such as payments.



Case study

Web language popularities

The IEEE Spectrum Survey recently completed a survey using the following sources to rank the popularity of web programming languages:

- search results in Google
- data from Google Trends
- tweets sent on Twitter
- GitHub repositories
- StackOverflow questions
- Reddit posts
- Hacker News posts
- demand for jobs on the Career Builder job site
- demand for jobs on the Dice job site
- IEEE Xplore journal articles.

Popular programming languages

- Definite loops have a set number of iterations, such as FOR...NEXT where the number of loops is defined in the FOR line.
- Java
- C
- C++
- Python
- C#
- PHP
- JavaScript.

Power

The power of HTML has increased with each revision as more features are added to improve what can be done inside the browser and to help performance and platform independence.

HTML5 provides improvements including:

- editable content on a webpage
- validation for email addresses
- better form features
- better audio support.

Protocols

HTTP (hyper text transfer protocol) has been one of the major enablers of web browsing, providing a request/response relationship between the webpage in the browser and the website providing the page.

HTTP carries overheads such as the need for HTTP to poll the server to get new updated data, which slows the communications.

Newer technologies such as WebSocket offer a persistent connection where the client and server can send data to each other at any time.

Now try this

Research a web language to produce a summary of your choice.

You could use Java for this exercise.

Coding for the web: Uses

Code can be written using web languages to respond to the user in ways such as calculating a shopping cart total. Such code can utilise client-side processing and **scripting** to run on the user computer or can be **server-side** with a request to the server for data.

Client-side processing

This code can run in the browser to show whether the user has visited the website before by opening a cookie then showing a prompt for the user to enter their name if the cookie is not present, setting the cookie after the user name is entered so it is there for the next visit.

```
function Visitor(info){
    var vName = GetCookie('vName')
    if (vName == null) {
        vName = prompt("Please enter your name");
        SetCookie ('vName', vName, exp);
    }
    return vName;
}
```

```
var vSQL = opDB.createStatement();
vSQL.executeQuery("SELECT * FROM dbTable;");
var SQLresult = vSQL.getResultSet();
while (SQLresult.next()) {
    results.push(new DataModel({
        Ref: SQLresult.getInt("Ref"),
        Username: SQLresult.getString("Username"),
        Description: SQLresult.getString("Description")
    }));
}
```

Server-side processing

The code fragment to the left can run on a server to select all the records from a MySQL database table named dbTable, then loop around them to display in the browser screen.

The main web platform choice for the hosting servers is between Linux and Windows. These are both good and reliable systems, so most of the issues and implications of implementing are down to how the server-side code is set up, programmed and tested.

Server-side and client-side advantages

Server side	Client side
Good for:	
✓ MySQL queries to retrieve data	✓ interacts directly with objects, for example buttons on user browser window
✓ saving user data	✓ fast response times
✓ validating users.	

Implementing code on a web platform

Most of the **issues** around code on a web platform are about compatibility with the browser and that security settings may prevent code from running on the user computer.

The main **implication** is that a reliable internet connection needs to be present for the browser to connect to the web.

Now try this

Produce an example each of where client-side and server-side processing are used, with the implications of implementing such code on these web platforms.

eBay could be used for the server-side processing example.

Had a look Nearly there Nailed it!

Translation issues

Translating code between programming languages is sometimes needed when implementing a new system. The implications and impacts on users, organisations and developers need to be anticipated and planned for to minimise disruption.

Translating between languages

Writing code always needs the programming language to translate this high-level code into low-level code the computer can understand.

Translating a high-level language into another high-level language needs a program called a translator (source-to-source compiler). These are mostly used to move code from old to newer hardware between languages such as FORTRAN-to-Ada.

Reasons for translating code

- Replacement of old with new hardware which will not support legacy code.
- New software environment used with the organisation.
- Need to keep to new standards brought in by the organisation.

The drawbacks of translating code outweigh the benefits.

Benefits
Can save time
Can save cost from not needing much programmer involvement
May be able to use a lot of the original documentation from the original app
May be able to re-use previous test plan and data

Drawbacks
Time to test the new translations work correctly
Distancing of translated version from original program design and implementation
Readability of code reduced because humans write code that makes sense using names of variables
Translated code may need a lot of editing to make it understandable for future maintenance
Original code results from a lot of thinking and planning, may use structures too subtle for software utility to recognise, leading to errors
Will need thorough testing to ensure that everything still works as intended and that no logic or processing errors have occurred

Implications for users, organisation and developers		
Users	Organisation	Developers
Users familiar with the older version of an app may find that a new version changes the user interface in subtle and annoying ways, such as order of selected objects on a form using tab, keyboard shortcuts not working, or forms not holding the focus as expected.	There will be a cost and time required to test the translated app. There might be a knock-on impact if subtle errors in output data are not detected during testing and go live.	Developers will almost certainly find the new code less comprehensible than the old and harder to maintain.

Now try this

Explain why an organisation might need to translate code between programming languages.

What change in the systems would bring about this need?

Translation alternatives

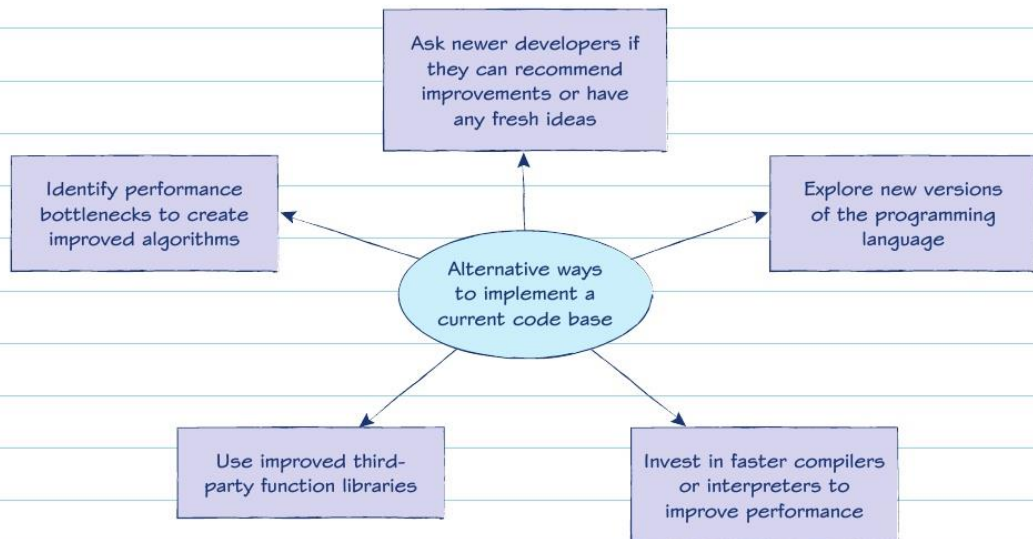
On this page, you will revise alternative methods of translating code.

Translating code

There are many problems in producing a utility to convert code from one language to another. Each language has a large collection of low-level code used to make the program source code written by programmers into the machine code that can actually run on the computer. This makes it very difficult for conversion software to move code that both runs and can be maintained by programmers.

Google Web Toolkit (GWT) and Hiphop

- The GWT utility generates JavaScript from Java and features a test browser where you can test how the Java code executes with a step-through debugger.
- Facebook produced its HipHop Virtual Machine (HHVM) to compile PHP into C++ to improve how quickly webpages can run code.



Implications of translating code

Whatever programming paradigm being used, the ability to debug programs, identify errors and fix them remains essential. Finding errors in computer-generated code can be very difficult.

The short-term gain from moving the code can easily be lost if the new solution is difficult to understand and edit. Any problems that are encountered will need to be resolved by editing code and testing the modified solution.

Now try this

Produce an advert for a named translation tool.

Try to find some webpages reviewing the tool.

Had a look Nearly there Nailed it!

Your Unit 1 exam

Your Unit 1 exam will be set by Pearson and could cover any of the essential content in the unit. You can revise the unit content in this Revision Guide. This skills section is designed to **revise skills** that might be needed in your exam.

Exam skills

These are some of the exam skills you'll need:

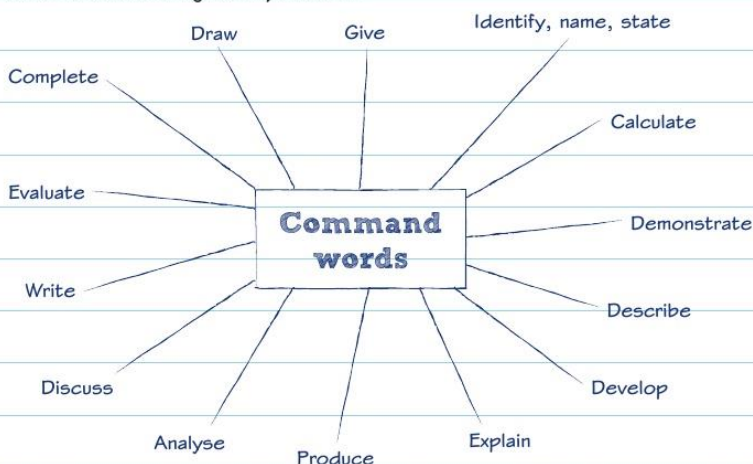
- | | |
|---|--|
| ✓ Understanding the question
Revise this on page 42 | ✓ Longer-answer questions
Revise this on page 46 |
| ✓ Short-answer questions
Revise this on page 43 | ✓ Analyse data and information
Revise this on page 47 |
| ✓ Performing calculations
Revise this on page 44 | ✓ Predicting outcomes
Revise this on page 48 |
| ✓ Drawing diagrams or flow charts
Revise this on page 45 | ✓ Evaluate questions
Revise this on page 49 |

Question types

Command words are the key terms used in questions, for example 'identify', 'explain', 'draw'. They identify the approach you should take to answering the question.

Question types

Command words are the key terms used in questions, for example 'identify', 'explain', 'draw'. They identify the approach you should take to answering the question.



Exam checklist

Before the exam, make sure you have

- ✓ a black pen you like and a spare
- ✓ a pencil, sharpener and eraser for drawing diagrams and flow charts
- ✓ a calculator in case you are asked to perform calculations
- ✓ double checked the time and date of your exam.

Check the Pearson website

This section is designed to demonstrate the skills that might be needed in your assessed task. The details of your actual assessed task may change from year to year so always make sure you are up to date. Check the Pearson website for the most up-to-date **Sample Assessment Material** to get an idea of the structure of your assessed task and what this requires of you.

Now try this

Create a revision timetable around your exam schedule.

Understanding the question

Understanding what a question is asking you to do will help you to take the right approach to your answer. Here are some skills you could use to enable you to focus on the purpose of the question.

Worked example

Building For Excellence (BFE) is a group of local builders wanting a mobile phone app to give an instant price for a job, which can then be texted to the client. The app should be simple and easy to use.

The client requirements for the app are given in the information.

Describe the problem, inputs, processing and outputs for the BFE app.

4 marks

Sample response extract

The app is to run on mobile phones, be easy to use and allow BFE builders to give instant prices for jobs. The details of the price can then be sent to the client as a text. Inputs will be name of client, mobile number of client, hours needed and materials. Processing to multiply hours by hourly rate, multiply distance by travel rate, add costs. Output to show costs of hours, travel, total cost, text to client.

Read the scenario and tasks very carefully to help you understand the type of answer you are expected to write.

Identify and highlight the *command word* in the question. This indicates the skills you will need to demonstrate in your answer. In this revision question, **describe** indicates you will need to outline the key features of four items.

Look at the marks available for a question which should help you to understand the level of detail required. Questions that ask you to demonstrate or apply your knowledge and understanding will have fewer marks than an **evaluate** question which will require a longer answer and may have a maximum of 12 marks.

If you have time, re-read your answers to check they make sense and answer the question.

Worked example

Draw a screen design for the app and label the main features.

4 marks

Sample response extract

The command word **draw** asks you to show your understanding through a diagram or flow chart. In this revision question, the task also requires you to label the diagram, so be sure you include annotations.

Make sure you carefully read the client requirements to include inputs for all the data that will be required for the app.

Make sure every output identified in the client requirements is included in your design.



Revise pages 1, 2 and 3 on decomposition

Now try this

Write pseudocode for the calculations needed by this app.

Short-answer questions

Short-answer questions, with command words such as explain, identify, state, want you to provide factual information. The answers normally only require one or two words or short sentences.

Worked example

State two reasons why flow charts are used.

2 marks

This is a **state** question, so your answer will not need much detail.

Sample response extract

- 1 To explain how an algorithm works
- 2 To help in identifying errors in code


Worked example


Explain how each of these flow chart symbols is used.


3 marks

This is an **explain** question, so your answer should communicate the usage of each symbol.

Describe each symbol.

 (a) _____

 (b) _____

 (c) _____

This question is simply asking you to **identify** the items. Don't worry about the pros and cons.

Worked example

Identify an appropriate item of information that could be held in each of these types of variable:

- (a) Alphanumeric string
- (b) Character
- (c) Floating point (real)

4 marks

Sample response extract

- (a) A terminator used to show each place where the program starts or ends
- (b) Input/output used to show points in the algorithm where data is input or output
- (c) Decision used to show the places in the algorithm where the program takes one of two branches according to the decision shown in this symbol

Sample response extract

- (a) Anything which could be a mix of letters, numbers, spaces and general typing such as an address, e.g. 12 Station Road
- (b) Similar to alphanumeric string, but can only be a single character such as a variable used to hold gender, e.g. M
- (c) Numbers with fractional parts such as a variable used to hold the average value of a data set, e.g. 23.453

The learner has clearly explained each of the three flow chart symbols. This answer includes enough detail and keeps to the point.

Make sure you include an example for each.

 **Links** To revise types of data, see page 13.

Now try this

Identify appropriate items of information that could be held in each of these types of variable:

- (a) date/time
- (b) integer
- (c) Boolean.

Performing calculations

You may need to show you have the skills to perform calculations involving either numbers or string handling.

Worked example

A program calculates cost of paint to decorate a room. Paint is available in 5-litre cans (£10), 2.5-litre cans (£6) and 1-litre cans (£3). A litre of paint covers 2 m². Variables are; Cans1l, Cans25l, Cans5l, Cost, Coverage, PaintCost, PaintNeeded, WallArea, WallHeight, WallLength.

- What will WallArea hold if WallLength=12, WallHeight=3 (metres)?
- What will PaintNeeded hold?
- What will Cans1l, Cans25l, Cans5l hold?
- What will be the cost of paint for this job?

8 marks

Sample response extract

- WallArea = WallLength * WallHeight, so will hold 36
- PaintNeeded = WallArea / Coverage so will hold 18
- Cans5l = integer of PaintNeeded/5 so will hold 3
PaintNeeded = PaintNeeded - (Cans5l * 5)
Cans25l = integer of PaintNeeded/2.5 so will hold 1
PaintNeeded = PaintNeeded - (Cans25l * 2.5)
Cans1l = round up of PaintNeeded so will hold 1
- Cost = (Cans5l * 10) + (Cans25l * 6) + (Cans1l * 3) so will hold 39

Each part of the answer covers two points. The learner has also shown their workings.

Worked example

A game uses variables to hold strength (as a percentage), time (seconds), kills (integer) about the player with the score calculated as $1000 * \text{strength} * \text{kills} / \text{time}$.

- How does time affect the score?
- Identify three combinations scoring 40.

8 marks

Sample response extract

- A shorter time results in a higher score. This is because the time is divided into the other variables.
- Three combinations scoring 40:
Strength=50% Kills=8 Time=100
 $50\% * 8 = 4$, $4 / 100 = 0.04$, $1000 * 0.04 = 40$
Strength=50% Kills=16 Time=200
 $50\% * 16 = 8$, $8 / 200 = 0.04$, $1000 * 0.04 = 40$
Strength=100% Kills=8 Time=200
 $100\% * 8 = 8$, $8 / 200 = 0.04$, $1000 * 0.04 = 40$

Use a pencil and paper to define how the score is calculated; strength * kills / time with some example numbers. Use numbers that are easy to multiply and divide to get a 'feel' for how the calculation works.

When answering a **calculate** question, remember to show your workings as part of your answer. Even if your answer is not correct, you may get credit for your method.

Now try this

A multi-level game is being developed with the score calculated from the number of steps taken in each level added to a bonus awarded for completing the level. The points for each step in level 1 is 10, each subsequent level has 5 added to the previous step points so level 2 has 15 points per step, level 3 has 20 points per step and so on. The bonus for completing level 1 is 50, with the bonus doubling for subsequent levels, so level 2 has a bonus of 100 points, level 3 has 200 and so on.

What will be the score from 50 steps in level 1, 48 steps in level 2, 58 steps in level 3, 52 steps in level 4 and 12 steps in level 5?

Include the bonus for each level.

Had a look Nearly there Nailed it!

Drawing diagrams or flow charts

You may be asked to show that you can draw or complete a diagram or flow chart.

Worked example

This is an example of working code for an insertion sort:

```

For L1 = 1 To Last
  Tmp = A(L1)
  For L2 = L1 To 1 Step -1
    If A(L2 - 1) > Tmp Then
      A(L2) = A(L2 - 1)
    Else
      Exit For
    End If
  Next L2
  A(L2) = Tmp
Next L1
    
```

Draw a flow chart for this algorithm using standard BCS symbols.

5 marks

Make sure your flow chart accurately shows how this algorithm works.

You may find it helpful to produce a quick rough sketch to clarify your thinking.

Remember to use BCS symbols in your flow chart.

You will need to include every line of code in your flow chart.

Links For more on BCS flow chart symbols, see page 11.

Try to get most of the data flows in your flow chart to go down or right. Use arrows to confirm each direction.

Draw your diagram as clearly and neatly as possible.

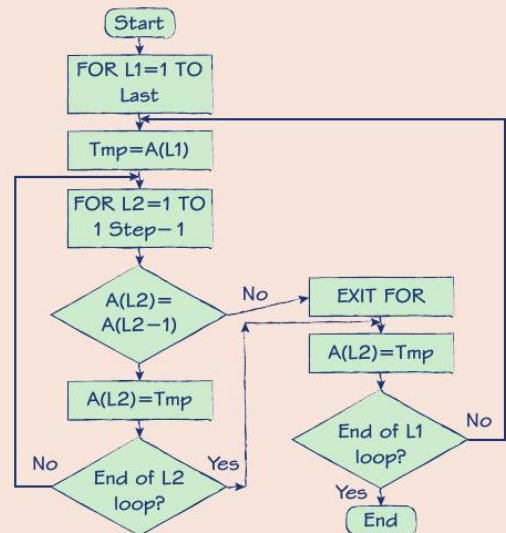
Decision box must show the correct logic.

Yes and No routes must be correctly labelled.

The data flows should be down or right unless arrows show otherwise.

Links To revise flow charts, see page 11.
To revise insertion sort, see page 28.

Sample response extract



Now try this

Write pseudocode, then draw a flow chart for a linear search algorithm. You should be able to devise your own algorithm which accepts the search criterion then loops through a data set until the criterion is found with an output to report on finding the item or that the criterion was not present in the data set.

Longer-answer questions

Some questions may require longer, more in-depth answers. You will need to take a slightly different approach to these questions.

Worked example

Programming languages use stacks to remember return addresses from sub-routine calls.

Analyse with reference to a diagram how a programming environment passes program execution to a sub-routine and the data structure used to return to the point in the code where the sub-routine was called when the sub-routine completes.

10 marks

Sample response extract

Program execution simply runs through the code line by line unless a control structure causes a different route through the code. A sub-routine is a named sub-program that can be called from anywhere (within scope) of the program so the execution jumps to the sub-routine then returns to the next line of code after the call.

When program code is executed a specialised register (program counter, PC) inside the CPU holds the address of the next instruction in memory. When an instruction is brought into the CPU, usually, the length of the instruction is added to the PC so it now points to the next code to be run from memory.

Calling a sub-routine needs the PC to 'save' the address that is there for the next instruction so it can be brought back when the sub-routine is exited. After this address is saved, the address of the first line of the sub-routine can be put into the PC so the program can execute this sub-program.

Saving the address from the PC needs to be flexible, so if there is recursion, or other sub-routine(s) are called whilst the sub-routine is executed, the place to return to is brought back in the correct order.

For this reason, a stack is used to save the return addresses. This is a FILO (first in last out) structure so each return address can be pushed onto the stack when the sub-routine is called then popped off when completed to the PC so execution can continue from where the last call was.

The diagram shows a stack with four return addresses. The first time the program called a sub-routine was return address 1. This sub-routine was still running when another sub-routine was called, causing return address 2 to be pushed onto the stack. While this sub-routine was running it recursed to call itself two times, so return addresses 3 and 4 were pushed on.

When the sub-routines exit, these return addresses can be popped off the top of the stack so each returns to the point where it was called.

The stack is a great structure for these operations as the sub-routines can return in reverse order.

Take a little time to make sure you fully understand what the question is asking you to do. Note down the points you want to cover on a piece of paper.

For this **analyse** question you need to be clear on:

- program execution, including sub-routines and data structures
- the uses of stack-based memory allocation.

Start your answer with a clear introduction which makes reference to the question. Here the learner explains what happens when code is executed and defines a sub-routine.

Try to plan the structure of your answer in a logical way rather than just adding points as you remember them.

Use accurate technical language wherever possible.

Your diagram might look like this:

Return 4
Return 3
Return 2
Return 1



To revise data structures, see page 22.

Now try this

Discuss how a function can be written by a coder into their program. Include what is needed to define the function, declare arguments, return the value and call the function.

Analyse data and information

Analyse questions want you to examine a topic in detail and look at the relationship between the parts. This might involve discussing the advantages and disadvantages of possible solutions and predicting outcomes, but you may not be expected to give a conclusion.

Worked example

Restique Relics (RR), a successful restoration and reclaimed building antiques retailer, is considering raising its online presence. It already advertises selected individual items on online market places. RR plans to set up a website so potential clients can see all of their current products with a shopping cart and checkout.

Discuss the ways that client-side and server-side scripting could be used to implement a website for Restique Relics. **6 marks**

Sample response extract

When you visit a website there are many ways that scripting can be used to enhance the experience. [1] Server-side scripting runs on the server computer that you are visiting and client-side scripting runs in the browser of the computer you are using. [1]

The main use for server-side scripting is to search the data sets held on the server and to return the results to you. [1] RR would use MySQL to allow visitors to its website to see the products and to search for specific items or features.

The shopping cart will be controlled by scripting. This could be on the server or the client, but the better choice would be the server. [1] This is so the information about products selected, client, payments, etc. can be kept safely under the control of the website. This would provide a more reliable solution, e.g. two people would not be able to buy the same product.

Client-side scripting would be good to respond to the browser to make best use of the features provided. [1] Cookies can be used to record visits and to welcome clients back.

The best websites use a mix of server- and client-side scripting to produce a quick and accurate browsing experience. [1]

Analyse questions require you to carefully consider the scenario. You need to clearly explain the points you are making and always link them to the scenario.

For this question, you need to be clear on the uses of both:

- server-side scripting
- client-side scripting.

Try to write answers that use fluent and accurate technical vocabulary.

Start your answer with a short introductory sentence, linked to the question and scenario.

Remember to include a clear definition of what you are going to be writing about, in this case server-side scripting and client-side scripting.

Explain the main uses of server-side scripting and client-side scripting.

Your answer should consider the advantages/disadvantages of the solution.

This is a balanced consideration of the issues, using logical chains of reasoning showing a full awareness of their relative importance to the scenario.

The elements of the question are carefully considered with arguments clearly linked to the given scenario.

Links To revise how websites use code, see pages 37 and 38.

Now try this

Discuss the benefits and drawbacks of translating code between programming languages.

Start by defining what is meant by translating code between programming languages in your answer and why this might be needed. Remember to identify and explain both benefits and drawbacks.

Links To revise translating code between programming languages, see pages 39–40.

Predicting outcomes

Here are some examples of skills which will help you to predict outcomes from a given computing solution.

Worked example

A program has a form with three text boxes, txtNumber1-#3, six labels, lblResult1-#6 and a button to run the code shown below:

```
Dim tNumber1
Dim tNumber2
Dim tNumber3

tNumber1 = CSng(txtNumber1.Text)
tNumber2 = CSng(txtNumber2.Text)
tNumber3 = CSng(txtNumber3.Text)

<#1>lblResult1.Text = tNumber1 + tNumber2 * tNumber3
lblResult2.Text = (tNumber1 + tNumber2) * tNumber3
lblResult3.Text = tNumber2 / tNumber3
lblResult4.Text = tNumber1 + tNumber2 Mod tNumber3
lblResult5.Text = tNumber1 <#2>< tNumber3 Mod tNumber2
lblResult6.Text = tNumber2 <2>> tNumber1 * tNumber3
```

For this question, you will need to follow the program execution and predict the outcome. Write down the input numbers, carefully read each line of code and perform the calculation described.

Complete the table with the outcomes you predict for the six labels for each row of inputs into the text boxes.

8 marks

Remember the BIDMAS rule for calculations.

Sample response extract

	lblResult1	lblResult2	lblResult3	lblResult4	lblResult5	lblResult6
txtNumber1	2					
txtNumber2	5	22	28	1.25	3	TRUE
txtNumber3	4					FALSE
txtNumber1	4					
txtNumber2	1	7	15	0.33333	5	FALSE
txtNumber3	3					FALSE
txtNumber1	3					
txtNumber2	5	43	64	0.625	8	FALSE
txtNumber3	8					FALSE
txtNumber1	3					
txtNumber2	8	19	22	4	3	FALSE
txtNumber3	2					TRUE

Notice logical operators. These will produce true or false values.

You may find it helpful to write out your calculations when working out the answer. However, this revision question requires you only to note down what would be displayed on the labels – so you should show only what will appear as a result of running the program.

Now try this

Create a spreadsheet or write a program with these calculations. Add another four calculations of your choice using any combination of mathematical (+-*/), relational (= <> <= >=) and Boolean (NOT, AND, OR) operators. Work out the answers for some test data and check them with the results from the app you've produced.

Had a look Nearly there Nailed it!

'Evaluate' questions

Questions that ask you to evaluate want you to show you can consider all sides of an argument in order to provide a well-supported judgement on a topic or problem. This may include writing a supported conclusion or a recommendation for the technologies, procedures and outcomes of a computerised solution.

Worked example

A dedicated computer system uses memory to hold numerical data.

Evaluate integer and floating point (real) data types usage by program code and how they are held in memory.

8 marks

Sample response extract

Computers use a range of structures to hold data inside memory using just two digits, 0 or 1, which are only good for yes/no, but when 8 of these BITS are grouped into a byte, 256 combinations are available.

Integers are whole numbers, held in memory as simple binary numbers occupying 2 or 4 bytes. This choice of short (range of -32768 to 32767) or long integer (-2147483648 to 2147483647) allows less memory for whole numbers expected to be in the short range.

Floating point numbers resolve issues integers have with numbers which are very large or have a fractional part. These are also known as real because they are the closest to numbers in the real world.

Floating point uses the same principles as standard form with a mantissa and an exponent. The mantissa is the high part of a number normalised with the decimal point moved to the largest digit and the rest of the number rounded to a set number of places, for example, in denary 24457892.25 becomes 2.45, with the exponent holding how many places the point moved, so this standard form is 2.45×10^7 or 2.45E07.

There is a rounding issue as the number is lost when the mantissa is formed. The original number cannot be fully recalculated. In this example, 2.45×10^7 could only be brought back as 24500000.

Both standard form and floating point are very space-efficient techniques for holding very large numbers.

Floating point works in the same way, except that it is all in binary. 24457892.25 is a large binary number, 01011101010011001010100111.01, which in floating point would have 0.1011101 as an 8 bit mantissa and an exponent of 00011001 as binary point moves 25 times from the low end of the original binary number.

Integers are best for whole numbers in a known range, with floating point for very large or small numbers and those which have fractional parts.

For this **evaluate** question, you need to be clear on the concepts of:

- binary
- floating point numbers
- integers.

Remember to start your answer with an introduction, linked to the question and scenario.

Use your technical knowledge of the area and combine it with examples to help support your explanations and evaluations.

If possible, use a calculator to help get your number work correct.

Try to give good examples with accurate workings. Here there are examples of the ranges of different data types and binary representations of numbers.

Now try this

Evaluate the uses of one- and multi-dimensional arrays with examples. Plan your answer using bullet points.