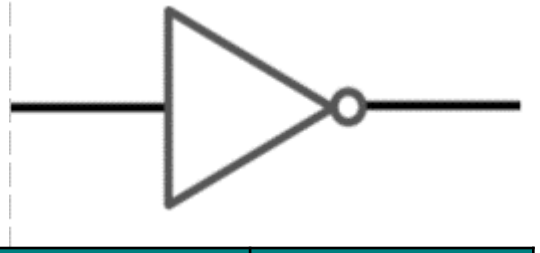
GCSE | Computational Logic

Knowledge Organiser

NOT

Takes the input and outputs the opposite(inverts)



Input	Output
0	1
1	0

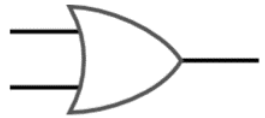
AND

For an AND gate to give output of 1, both inputs must be 1

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

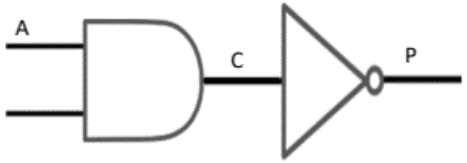
OR

For an OR gate to give output of 1, either inputs must be 1.



Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Combined



A	B	C	P
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Logical Expressions - These are very similar to algebra. The rule of BIDMASS applies. Brackets should be dealt with first. You may be asked to draw a logic gate from an expression . E.g.

p = NOT(A AND B)

P = NOT(A OR B)

P = NOT(A OR B) AND C

P = (A AND B) AND C

Truth Table – this is the table that is filled in to identify the different inputs and to determine the output of different situations

You may be expected to do the following:

- Draw a single gate based on the word
- Identify the name of a gate based on the illustration
- Fill in a truth table
- Draw a logic gate from a given expression
- Draw a logic gate for a given scenario

GCSE | Testing

Knowledge Organiser

What is testing?
Testing a phase of the software development cycle that is undertaken to ensure that there are no errors in the programming that would prevent it from working or from working correctly.

There are different types of testing:
Iterative testing – this is the type of testing that is done repeatedly throughout development. For example one part is coded then tested, then the next part coded and then tested
Final testing – this is the type of testing that is done at the end of the development when program is finished.

When testing code there are different types of error to look out for. You need to know about two types of error

- **Logical Error -** With logic errors the program will work, however an unexpected output/outcome will be given. This may be due to an incorrect logical operator being used e.g < instead of >. Or somewhere in the program a calculation is wrong that then has a knock on effect.
- **Syntax Error -** These are errors that break the "grammatical" rules of the programming language.e.g you forget to put a : on the end of an if statement. This will prevent the program from working/being translated.

Test plan – A test plan is a document that is produced to ensure that all the different paths are tested thoroughly. It is a way of ensuring all the different types of test are conducted.
See example below:

Test num	Test description	Test data	Expected outcome	Actual outcome
1	Does the game start when option 1 is pressed?	Normal – type In number 1	Game loading screen displayed	As expected
2	Is an error message displayed when invalid date is entered?	Erroneous - type in 100	Error displayed saying invalid option try again	As expected

You could be asked to complete or draw a test plan

When it comes to testing a program or filling in a test table there are 3 types of test data to considered:

- **Normal** – This is valid data that will not produce an error
- **Erroneous** – This is data that will produce an error, it is usually extreme for example if there is a menu with 1)play game 2) view scores and 3)exit. Someone typing in 393 or "jkfk" should display an error message
- **Boundary** – This is data that is on the edge of what is acceptable. Again using the example above. 1 and 3 are boundary as are 0 and 4 because they are on the edge of acceptable

Using this it will ensure all different circumstances are checked

GCSE | Defensive Design

Knowledge Organiser

What is it ?
Programmers try to protect their programs by testing them to reduce the number of errors, predicting how users might misuse their program and trying to prevent it and making sure their code is well maintained..

Input Sanitisation – removes any unwanted characters that have been entered into a program. This can be used to prevent the likes of SQL injection when users are filling in an online for
Input verification – this will ask the user to verify what they have entered. A good example is when setting up a new account and it ask the user to enter their password twice to ensure that it is correct.
Input Validation – Checks if the data meets certain criteria before passing it through the program. The following validation checks can be used:.

- Presence Check – Checks that data has actually been entered, so not blanks
- Length check – checks data meets a certain length criteria. E.g password should be 8 characters long
- Format check – Checks data is in the right format e.g. an email should always have an @ symbol in it
- Look- up table – Checks data against a list of values, these will be presented in a drop down usually
- Check digit- checks numerical data is entered correctly

Anticipating Misuse - Predict how users may misuse the code and code it to cope with unexpected or erroneous data.
Authentication – this is when a program confirms the identity of a user before giving access to the full program. This is commonly done through usernames and passwords. To ensure that know unauthorized users access the system.

Maintainability – Code that has been well maintained is easy to edit without causing any errors. When a programmer is writing their code they should make sure that it is easy for someone else to work with and also it is easy to understand when the programmer comes back to it. There are a range of steps that can be taken to improve the maintainability of code:

- **Comments** - explain the code so that other programmers can understand it.
- **Variable names –** Give variables names that match what they are storing, so it makes sense to others.
- **Sub-programs –** Use sub-programs to avoid repetitions in code and to make it more organised.
- **Indentation –** Make sure code is indented correctly so that it is easy to se the flow of the program.

GCSE | Languages and IDE's

Knowledge Organiser

High Level Language

- Eg: Python, Java etc
- Each instruction in a high level code represents many machine code instructions.
- The code will work on many different computers and processors
- Data can be stored in different structures like lists and arrays
- **The code is easy to read and understand as it uses words from the English language like WHILE , IF, FOR etc**
- The code has to be converted into machine code for the computer to understand it
- Programs will be less memory efficient as there is no control over what the CPU does

Low Level Language

- Eg: Machine code (binary) and assembly language
- Each instruction only represents one instruction of machine code
- Low level languages are written for one particular machine or processor
- To store data the programmer needs to understand
- how the CPU manages memory
- **Low level code is difficult to read and understand,, as it is written in binary and mnemonics**
- Machine code can be executed without translators
- Programs are more memory efficient as you control what the CPU does

Translators

High level languages have to be translated to machine code for the computer to understand them.

Assemblers – turn assembly language into machine code

Compilers – Translate all of the code in on go to create an executable file. A compiler can take a long time, but the final code runs quickly and gives a list of errors for the entire program.

Interpreters – Translates the code one instructions at a time. This means the program will run more slowly. No executable file is created so the code will need to be translated every time it runs. The interpreter will stop after each error which is helpful when debugging

IDE'S (integrated Development Environments

IDE's help programmers develop their code. They have a range of features to do this:

Editors – the area which the code is written in. Includes line numbers and colour coding for different features of the code (variables, comments etc)

Run Time Environment – Lets the programmer run the code quickly to test it for errors

Error Diagnostics – includes diagnostic tools to help find and solve errors

A Translator – to translate the code into machine code

Breakpoints – Stop the program on certain lines so that information up to that point can be gathered.

GCSE | Algorithms

Knowledge Organiser

Algorithm – A sequence of instructions followed to solve a problem

Abstraction – This is when unnecessary detail is removed from a problem to make it easier to solve

Decomposition– This is when instructions or broken down into smaller more manageable tasks

Searching Algorithms

Binary Search – Step 1 - find the middle item in the list

Step 2 - Ask a logical question such as is the search value bigger than the item in the middle?

Step 3 - Depending on the response the question in step 1, the items to the left or to the right will be dismissed

Step 4 - the process(step 1 to step 3) will repeat on the items remaining

Step 5 - When only one item is left the value has been found and the algorithm will stop.

Linear Search – Step 1 - starts at the beginning of the list and it will compare the search value to the item in position one.

Step 2 - If the search value is not found, it will increment by one and check the item in position two for the search value

Step 3 - this process will repeat, moving up the list by 1 each time

Step 4 - Once the search value matches the item, the algorithm will stop

Remember in your exam you must refer to the example data when explaining each of the steps. IF you are are ask to Show you draw out the steps. If you are asked to explain you write the steps along with for example….

Sorting Algorithms

Bubble Sort– Step 1 - Starts at the beginning of the list on the left, comparing the first two adjacent pairs.

Step 2 - If the item on the left is bigger than the item on the right, the positions of these will be swapped.

Step 3 - The search increments to the next adjacent pairs (so item 2 and item 3), then it will follow the process as in step 2

Step 4 - This process then repeats

Step 5 - Once the algorithm reaches the end the process will go back to the beginning.

Step 6 - The algorithm will only stop once no more swaps are required

Insertion Sort– Step 1 - Split the list into sorted and unsorted list. Sorted list on the left, unsorted list on the right

Step 2 - pull out the first item from the unsorted list and compare it with the last item in the sorted list

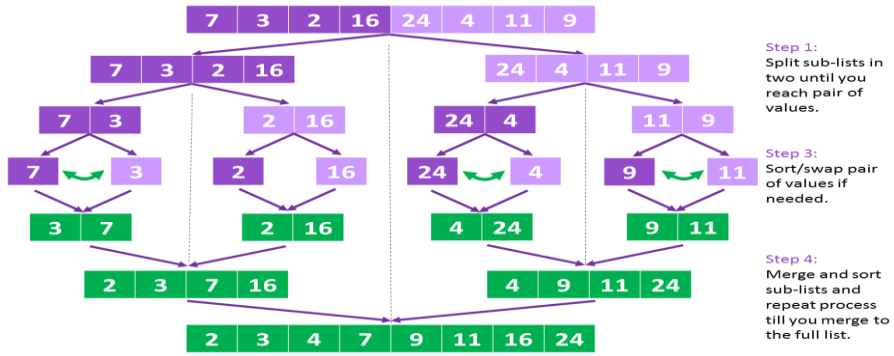
Step 3 - If item is bigger than the item to left in the sorted list it will be appending to the end

Step 4 - If the item is smaller than the item to the left in the sorted listed, it will move backwards through the sorted list

Step 5 - The process in step 4 is repeated, until the item is bigger than the item to the left, at this point it will be inserted into the sorted list.

Step 6 - Step 2 to step 5 will be repeated until there are no items left in the unsorted list.

Merge Sort



GCSE | Programming Concepts

Knowledge Organiser

Variables – A variable is a location in computer memory that stores a value that does not change
Constants – These are the same as variables, but the values **do not** change
Assign – This is when we give a variable or constant a value. We do this using a single **=** sign
e.g score = 0
Sequence – instructions followed in order

Data Types

Integer – These are whole numbers. **E.g Score = 10**
Real/float – These are numbers with a decimal value **E.g price = 2.99**
String – This is a sequence of characters inside speech marks " " **e.g school = "SEA"**
Boolean – This when there is two different options **e.g summer = TRUE**
Char – Single character **e.g option = "a"**

Casting – Sometimes we need to convert from one data type to another. For example when outputting a string along side an interger. To do this we use:
str() - converts to string, float() – converts to float, Int() –converts to integer

Selection – This is the programming technique used to ask a question/ make a decision/ to check a condition has been met. There are two types of selection that you need to recognise, trace, write and use. IF statements and switch case:

```
userType = input("Enter the user type:")
if userType == "admin":
    print("Access granted to all drives")
elif userType == "teacher":
    print("Access tostaff drive and pupil drive")
elif userType == "pupil":
    print("Access to pupil drive")
else:
    print("invalid user")
```

If – used to ask one questions
Elif(else if) – when there is more than one option
Else – when the result of an option is false

SWITCH CASE does the same as if statement but the variable being evaluated is only required next to switch. Then the case represents what it is being compared to

```
userType = input("Enter the user type:")
SWITCH userType:
    CASE "admin":
        print("Access granted to all drives")
    CASE "teacher":
        print("Access tostaff drive and pupil drive")
    CASE "pupil":
        print("Access to pupil drive")
    DEFAULT:
        print("invalid user")
ENDSWITCH
```

Input – This is when we enter data into the system. When identifying inputs you need to look for works that imply input(enter, ask, question, check)
Process – This is when an action takes places, usually a calculation or change a variable.
Output – This is when the system gives feedback from process.. When identifying inputs you need to look out for words that imply output(display, show, illustrate)

Arithmetic operators

(+) - for addition print(5 + 5) result: 10
(-) - for subtraction print(10-7) result:3
(/) – for divide print(30/10) result:3
(*) – for multiplication print(5*10) result:50
(^) – for power of print(2**4) result:16
(%) – MOD – remainder from division print(13%2) result: 1
(**//**) – **DIV** - Integer division(returns the whole before decimal) print(13//2) result: 6

GCSE | Programming Concepts 2

Knowledge Organiser

Iteration – This is the programming technique used to repeat a section of code for a specific number of times or until a specific condition has been met. The benefits of using iteration when coding is that it reduces the number of lines of code required, therefore saving the programmer time and taking up less storage on the computer. There are two types of iteration that you need to know about as outlined below:

Count-controlled loops – repeat a specific number of times

```
for i in range(1,101):
    print(i)
```

← This will print the numbers 1 to 100)

This will print out numbers - →
counting from 0 to 100 going up in 5's

```
for i in range(0,101,5):
    print(i)
```

Condition controlled loops – repeat until a condition is met

```
password = "admin12"
userPass = input("Enter your password")
while userPass != "admin123":
    print("try again")
    userPass = input("Enter your password")
print("access granted")
```

← This is a pre-test loop and will ask the user to enter their password until they enter "admin123")

This is a post test loop. It will always do one run of the repeating statements as the condition is only checked at the end→

```
password = "admin12"
Do
    userPass = input("Enter your password")
until userPass == password:
print("access granted")
```

Logical Operators

< means less than
> means greater than
>= means greater than or equal to
<= means less than or equal to
== means equal to
!= means not equal to

Boolean Operators

AND – This is the operator that is used when both conditions must be true
OR – This is the operator that is used when one condition needs to be true
NOT - This operator inverts the result of a condition for example of the condition is true it will change it to false

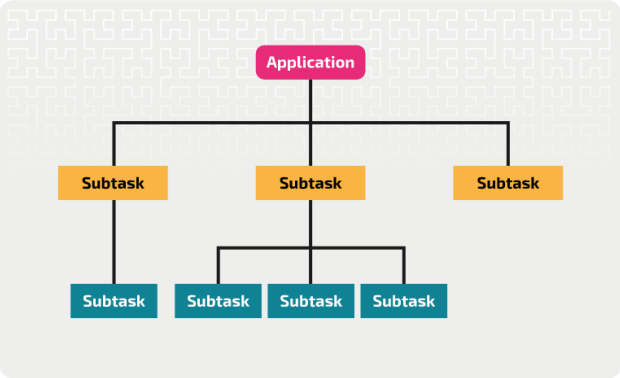
```
light = "on"
time = "night"
lamp = "off"
if light == "on" and time == "night":
        print("this will print out because both are true")
elif light == "on" or lamp == "on":
        print("this would be true because light is on ")
elif not(light == "on"):
    print("this will not print out as long as light is on")
else:
    print("example")
```

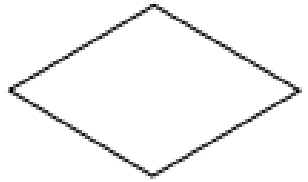
GCSE | Pseudocode and flowcharts

Knowledge Organiser

Pseudocode– A simplified way of writing a programming language, using structured English. Do not need to follow strict syntax rules.
Flowchart – A diagram used to illustrate the pathway through a program and processed. Used for design. Used to show algorithms
Trace tables – used to check the different outputs, and processes of a code based on given data. Can be used to ensure the logical of a program is correct

Structure diagram – It shows the hierarchy or structure of the different components or modules of the system and shows how they connect and interact with each other. As the level goes lower it shows the task being broken down into sub-task



	Line		Input/ Output
	Process		Decision
	Sub program		Terminal

Line – Used to show the direction in which the data is going. The only symbol which will have two arrows coming of it is the decisions. These arrows should be labeled true or false
Terminal – This is used for the start and end of flowchart
Input/output – Used to represent data being entered e.g name = input() and to display results e.g print(score)
Process – used for actions, calculations. Commonly used for setting up variables, assigning a value, updating a value
Decision – used for selection statements. To check a condition is met. Also used to represent iteration. These arrows will loop back up
Sub-program – used when calling a existing sub-program

GCSE | Advanced programming

Knowledge Organiser

String Manipulation

String length – Returns the size of a string.Counts the characters .length

Substrings – .substring(x,i) returns the value between x which is the starting value and i which is the number of characters.

.left(i) returns the values starting from the left with i referring to how many and **.right(i)** does the same but from the right

Uppercase - .upper converts a value into capitals

Lowercase - .lower converts a value into lower case

ASC(..) – returns the numerical equivalent in ascii of a give value

CHR(..) – returns the character for a given ascii number

```
subject = "computerScience"
subject.length # this gives the value 15
subject.substring(3,5) #gives the value "puter"
subject.left(4) #gives the value comp
subject.right(3) #gives the value nce
```

+ when working with strings + is used to concatenate. Join two or more values together

File Handling

```
myFile = open("sample.txt")
#this opens up a file from storage
#making the connection to the file
myFile.readLine()
#readLine returns the next line in the file
myFile.writeLine("add this")
#writeLine adds whatever is between the brackets
#to the end of the file

while NOT myFile.endOfFile()
    print(myFile.readLine())
endWhile
#the code above will read all the data
#from within the text file
#printing it out on a new line each time

myFile.close()
#the file should always be closed at the end

newFile("nameofFile.txt")
#this creates a new file
```

Be prepared to be asked to write the read the data from a given text file, create a new text file or add data to a text file. You are guaranteed a mark for opening an closing the file. The only bit that should change is the name of the file in green. The same when writing to a file the string in green is the only bit that you would need to change

Random number Generation

In python we use **import random** this allows us to use the functions of this module. In pseudocode, you just need to know how to generate random numbers

lotterynum1 = random(1,6) #this generates a number between 1 and 6 and stores it in a variable

Colours.txt
Red
Blue
Green
Orange
yellow

```
myFile = open("colours.txt")
while NOT myFile.endOfFile()
    print(myFile.readLine())
endWhile
myFile.close()
```

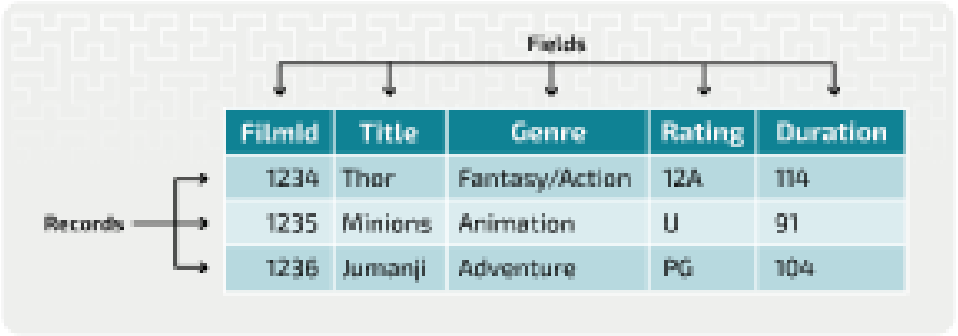
This would print out all the values in the text file. Each on a new line

GCSE | Advanced programming 2

Knowledge Organiser

Records

Database – A database is a collection of organized data. Databases are made up of tables and within each table there are fields and records.



FilmId	Title	Genre	Rating	Duration
1234	Thor	Fantasy/Action	12A	114
1235	Minions	Animation	U	91
1236	Jumanji	Adventure	PG	104

Record – is a collection of data for one object, person or thing. For example, in the Film table above, a record is a single row in this table about an individual file.

Fields – are used in a database to provide category headings for each item of data in the record

Array – an array is a data structure that is used to store multiple values. It is also known as a list.

One-dimensional array – a data structure that contain a set of elements of the same data type. To access an element of an array, an **index** number is used.

```
emotions = ["amazed", "delighted", "ecstatic", "enthusiastic", "lively"]
print(emotions[2])
#this would display ecstatic because indexing starts at 0
emotions[0] = "shocked"
#this would change amazed to shocked
for i in emotions:
    print(i)
#this would print out all the items in the array each on a new line
```

Index – This specifies the position of the element within an array.

Two-dimensial arrays – same as a one-dimensional array but uses two index positions. One represents the row and one represents the column

```
visitors = [["Bob", "M", 79, "y"],["Sue","F",65,"y"],["Rita","F",78,"N"]]
print(visitors[1,0])
print(visitors[1,1])
#This will print Sue and F
#the first value refers to the row
#the second value refers to the column
#print(visitors[row, column])
```

SQL - Structured query language, is used in order to extract, update, delete and add data into a database. It is used when searching a database for specific values. There are many SQL commands but you just need to know 3.

SELECT – specify the fields

FROM – specify the table

WHERE - specify the condition

Visitors Table

Name	Gender	Age	Member
Bob	M	79	Y
Sue	F	65	Y
Rita	F	78	N
Bill	M	70	Y

To search the table to the left for the name and age of all members the following query would be used

SELECT Name, Age
FROM visitors
WHERE Member = "Y";

GCSE | Advanced Programming 3

Knowledge Organiser

Sub-programs – These are small programs withing a larger, main program. They are designed to perform a specific task, as the task may need to be done more than once at various points in the main program. There are two types of sub-programs that you need to know about

- **Functions –** As above, bit functions specifically return a value back to the main program
- **Procedures –** As per sub-program but these do not return a value.

Both functions and procedures can be passed in parameters. These are values that are being transferred from the main program into the procedure or function. These are between the brackets. If you look at the codes below the function as one parameter number and the procedure has two num1 and num2

```
function squared(number)
    squared = number^2
    return squared
endfunction
#this sets up the function
newValue = squared(4)
#this calls the function passed in the value 4
#newValue will store 16
```

```
procedure multiply(num1, num2)
    print(num1* num2)
endProcedure
#this sets up the procedure

multiply(10,5)
#this calls the procedure passed 10 and 5 into it
#the outcome is 10 x 5
#Therefore 50 would be printed
```

What are the benefits of using sub-programs?

- They make the code easier to read because it is more organized
- If there is an error it only needs to be fixed once within the sub-program
- They are easier to maintain, if anything needs updating or changing
- They save the programmer time as they only need to write the code once
- It saves on storage as there is less lines of code

Call – This is what we do when we want to use a sub-program
Global Variable – this is a variable that has been created to be used in any part of the program
Local variable – This is a variable that can only be used in one part of the program. For example if it is created within a procedure it wouldn't be able to be used elsewhere