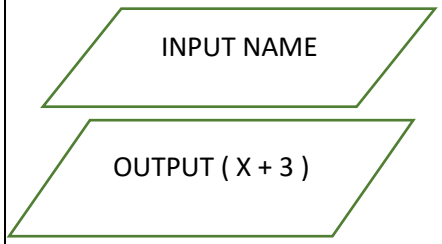
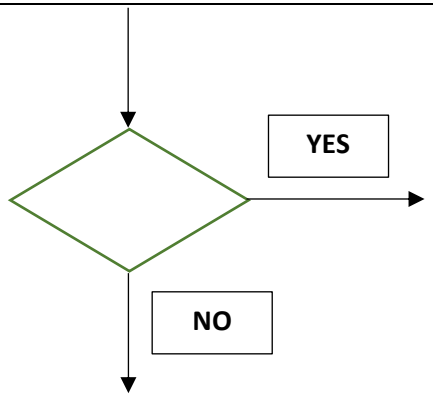
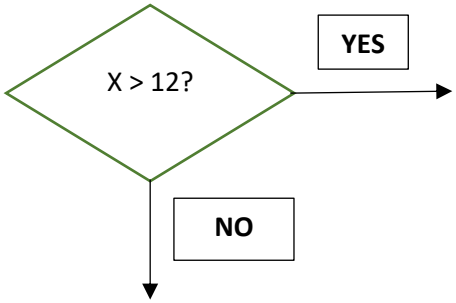
GCSE Unit 2 – 2.1 - Algorithms

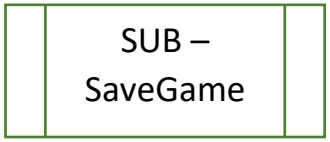
Symbol	Means…	Example	Pseudo Code
	Flow of control – follow the line!		None

	Input or Output		`NAME = INPUT("Please enter your name")`
			
			`OUTPUT( x + 3 )`

	Process – do something!		`Score = Score + NewPoints`

	Decision – IF this THEN that
	
	Note: must ALWAYS have TWO arrows coming out of it – one for YES and one for NO.
			`IF x > 12 THEN`
			`  Code for  YES`
			`ELSE`
			`  Code for NO`
			`END IF`

	Terminal – Start/Stop		`END`
			`END PROCEDURE`
			`END FUNCTION`

	Sub program – a separate flow chart to show a procedure or function		`PROCEDURE SaveGame()`
			`  Code`
			`END PROCEDURE`

GCSE Unit 2 – 2.1 - Algorithms

Key Vocabulary and Definitions

Term	Meaning
Abstraction	The removal of unnecessary detail from a problem. Helps us to focus on what is important, reduces complexity
Decomposition	The systematic breaking down of a problem into component parts which can then be solved with code
Input	In programming, Input is data being entered in to a program. Usually stored in a variable.
Process	A calculation or action carried out in a program
Output	In programming, output is the generation of text, graphics or sound shown to the user
Flowchart	A method of logically designing/showing the steps of an algorithm
Algorithm	A set of short, concise, logical steps which solve a problem.
Pseudocode	A method of writing code in a "generic" format. Not a real programming language, but can be used to help design and plan algorithms
Low Level Language	A language close to the binary instructions the CPU understands. E.g. Machine code, Assembly language
High Level Language	A language which is closer to structured English. E.g. Java, Visual Basic, Python
Syntax Error	An error of spelling or "grammar" in program code. Prevents the program from running.
Logic Error	An error in the logic of a program, for example using > instead of <. The program will run but won't behave as expected.
Runtime Error	An error which occurs when the program is running, usually causes the program to crash or stop.
Trace Table	A method of debugging a program by manually following the value of variables in a table whilst stepping through the code
Binary Search	A searching algorithm. Data must be sorted before it can be used, very efficient and quick, especially with large data sets
Linear Search	A simple searching algorithm which systematically checks data in order until the correct element is found
Bubble Sort	A simple method of sorting data. Works by comparing pairs of numbers and swapping if one is bigger than the other.
Merge Sort	A method of sorting data by breaking data down into atomic elements and then merging pairs of lists in order
Insertion Sort	A method of sorting data by systematically adding data into a sorted list in the correct place

GCSE Unit 2 – 2.2 – Programming Fundamentals

INPUT

```
VARIABLE = INPUT("sensible message to display")
```

Example:

```
Age = INPUT ("How old are you?")
```

OUTPUT

```
OUTPUT("Message")
```

Example:

```
OUTPUT("Don't look back in anger, I heard you say.")
```

OUTPUT OF VARIABLES

```
OUTPUT (VARIABLE_NAME)
```

Or

```
OUTPUT ("MESSAGE " + VARIABLE_NAME)
```

Example:

```
OUTPUT("You were sick " + AMOUNT_OF_CHUNDER + " times")
```

DECISIONS (SELECTION) – IF

```
IF true or false question/decision THEN
      What to do if true
ELSE
      What to do if false
END IF
```

ITERATION – LOOPS

```
FOR I = start_number TO end_number
      Code to repeat goes here
NEXT I
```

```
WHILE condition
      Code to repeat goes here
END WHILE
```

SAVING TO A FILE

```
FILEWRITER.OPENWRITE("filename.txt")
FILEWRITER.WRITELINE(data_to_write)
FILEWRITER.CLOSE
```

Example:

```
FILEWRITER.OPENWRITE("high_scores.txt")
FILEWRITER.WRITELINE(best_score)
FILEWRITER.CLOSE
```

READING FROM A FILE

```
FILEREADER.OPENREAD("filename.txt")
VARIABLE = FILEREADER.READLINE()
FILEREADER.CLOSE
```

Example:

```
FILEREADER.OPENREAD("high_scores.txt")
Best_score = FILEREADER.READLINE()
FILEREADER.CLOSE
```

PROCEDURES

```
PROCEDURE name_of_procedure (parameter, parameter…)
      CODE GOES HERE AS NORMAL
END PROCEDURE
```

Example:

```
PROCEDURE roll_dice (player_name)
      Dice_roll = getrandom(1,6)
      Output(player_name + " rolled a " + dice_roll)
END PROCEDURE
```

FUNCTIONS

```
FUNCTION name_of_function (parameter, parameter…) AS DATA_TYPE
      CODE GOES HERE AS NORMAL
      RETURN something!
END PROCEDURE
```

Example:

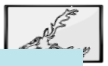
```
FUNCTION check_if_even (number_to_check) as BOOLEAN
      IF number_to_check MOD 2 = 0 THEN
            RETURN TRUE
      ELSE
            RETURN FALSE
      END IF
END FUNCTION
```

Key Vocabulary and Definitions

Term	Meaning
Sequence	Program instructions which are executed one after the other.
Selection	The use of "IF… THEN… ELSE…." To make decisions in a program or change the program flow
Iteration	The use of WHILE or FOR loops to repeat code a number of times. Makes code more efficient!
Variable	A piece of memory allocated to the storage of data, with a name/label. Used to store a SINGLE piece of data in a program. Usually has a data type (can store only a certain type of data)
Constant	A piece of memory, allocated to the storage of a single piece of data, with a name/label. The contents CANNOT BE CHANGED in the program.
Operator	A collective term for mathematical operations, +, -, *, /
Input	Used to get data IN to a program, usually in the form VARIABLE = INPUT("MESSAGE")
Output	Used to get data OUT to the user, usually in the form OUTPUT("MESSAGE")
Assignment	The placing of data into a variable, uses the = symbol. E.g. day = "Thursday"
AND, OR, NOT	Boolean operators. Used to make comparisons or decisions, usually in IF… THEN statements. E,g, "IF a=2 AND b=2 THEN…"
Integer	Data type – whole numbers only
Real	Data type – Numbers with fractional parts
Boolean	Data type – True/False only
Character (Char)	Data type – A single letter, number or symbol
String	Data type – A collection of 0 or more letters, numbers, symbols
Casting	The conversion of one data type into another. E.g. INT to STRING
Open	The process of connecting to an external file so data can be READ or WRITTEN
Read	The process of taking data IN to a program from a file
Write	The process of taking data OUT of a program and placing it in a file
Close	The command used to disconnect from an external file
Record	A collection of related data, contains multiple fields which can have different data types
SQL	Structured Query Language – a language used to create, manage, search and maintain databases
Array	A method of storing multiple items of data, of the same data type, in an indexed list with a single name.
Sub Program – Function	A distinct block of code in a program. Has a name and can be "called" in the code when required. RETURNS A VALUE.
Sub Program - Procedure	A distinct block of code in a program. Has a name and can be "called" in the code when required.

GCSE Unit 2 – 2.2 – Programming Fundamentals

GCSE Unit 2 – 2.3 - Producing Robust Programs



2.3.1 Defensive design

- ☐ Defensive design considerations:
  - Anticipating misuse
  - Authentication
- ☐ Input validation
- ☐ Maintainability:
  - Use of sub programs
  - Naming conventions
  - Indentation
  - Commenting

Required

- ✓ Understanding of the issues a programmer should consider to ensure that a program caters for all likely input values
- ✓ Understanding of how to deal with invalid data in a program
- ✓ Authentication to confirm the identity of a user
- ✓ Practical experience of designing input validation and simple authentication (e.g. username and password)
- ✓ Understand why commenting is useful and apply this appropriately

2.3.2 Testing

- ☐ The purpose of testing
- ☐ Types of testing:
  - Iterative
  - Final/terminal
- ☐ Identify syntax and logic errors
- ☐ Selecting and using suitable test data:
  - Normal
  - Boundary
  - Invalid
  - Erroneous
- ☐ Refining algorithms

Required

- ✓ The difference between testing modules of a program during development and testing the program at the end of production
- ✓ Syntax errors as errors which break the grammatical rules of the programming language and stop it from being run/translated
- ✓ Logic errors as errors which produce unexpected output
- ✓ Normal test data as data which should be accepted by a program without causing errors
- ✓ Boundary test data as data of the correct type which is on the very edge of being valid
- ✓ Invalid test data as data of the correct type but outside accepted validation limit
- ✓ Erroneous test data as data of the incorrect type which should be rejected by a computer system
- ✓ Ability to identify suitable test data for a given scenario
- ✓ Ability to create/complete a test plan

GCSE Unit 2 – 2.3 - Producing Robust Programs

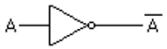
Key Vocabulary and Definitions

Term	Meaning
Defensive Design	Creating programs and applications that are designed to withstand misuse, unusual input or other unexpected circumstances
Anticipating Misuse	An aspect of good design and programming where a developer will predict how a program may be misused and design to combat this
Authentication	A method of securing a program or system, usually involving user names and passwords. Can be used as a method of preventing misuse
Input Validation	Checking any data which is entered into a program to ensure it meets a set of rules such as data type, length, format, case etc.
Verification	Checking data for accuracy by comparing two versions of the same thing to see if they match. For example, password verification when signing up to a web service.
Maintainability	Designing and coding in a way which enables the program to be edited, kept up to date, updated, amended or fixed if necessary in the future.
Sub Programs	A method of breaking complex programs up into smaller, re-usable chunks. This makes managing large projects much simpler as each sub program can be allocated to a different developer.
Naming Conventions	A form of maintainability and readability in code. Helps programmers to understand what variables and sub programs do.
Indentation	A coding convention where the TAB key is used to indent text. Helps to show which decision, loop, procedure or function each line of code belongs to. Makes reading code much easier.
Commenting	Notes left in the code by developers to help maintainability and readability of code. Helps future developers understand what the programmer was thinking and how the program works.
Testing	The systematic trialling of code with a set of data to determine whether a program behaves as expected and to highlight errors that will need to be addressed.
Iterative Testing	Testing which is carried out during development. A programmer will run their program frequently to check that changes work as intended, if not they will be fixed before moving on,
Final or Terminal Testing	Testing carried out at the end of development to check that the program meets the success criteria, works as expected and also works for large numbers of users.
Test data – Normal	A type of test data used to check that the system responds as expected under normal conditions
Test data – Boundary	A type of test data which tests the edge cases of a system. For example, a form which asks users to enter their age may accept values between 0 and 100. Boundary data would be 0, 100.
Test data – Invalid	A type of test data which is of the correct data type, but deliberately beyond the expected boundaries – in the previous example this would be -1 or 101
Test data – Erroneous	Test data which is deliberately incorrect and of the wrong type. Used to test how the system behaves under unexpected conditions.

GCSE Unit 2 – 2.4 – Boolean Logic

NOT

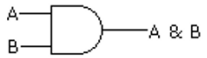
- The simplest logic gate
- Simply inverts the input
- Remember it by:
  - "The output is NOT the input"



A	NOT A
0	1
1	0

AND

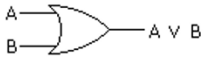
- The AND gate is true ONLY if both inputs are true
- Useful for testing whether things are on or off (masking)
- Remember it by:
  - "one AND the other input must be a 1"



A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

OR

- The OR gate can be seen as (almost) representing binary addition (learn this next lesson)
- OR is true if any input is true
- Remember it by:
  - "one OR the other must be true"



A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

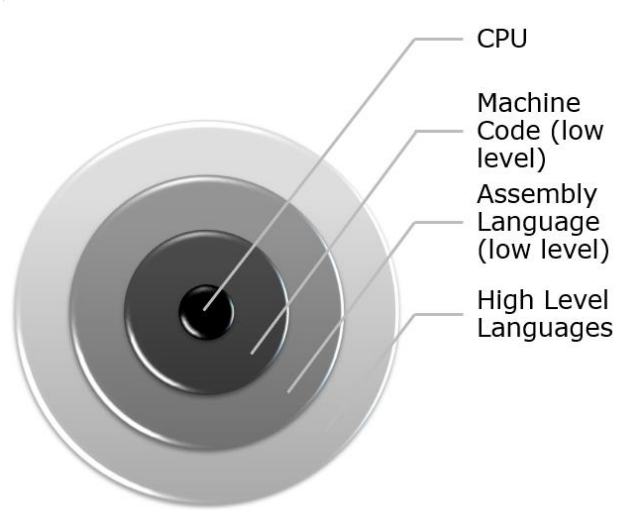
Symbols	Gate
. ∧	AND
+ v	OR
¬ ! _	NOT

GCSE Unit 2 – 2.4 – Boolean Logic

Key Vocabulary and Definitions

Term	Meaning
Boolean	A form of logic expressed using only two states – `True or False (1 or 0)`
AND	A Boolean operator. Will output a 1 ONLY if both inputs are also a 1.
OR	A Boolean operator. Will output a 1 if ANY of the inputs are also a 1.
NOT	A Boolean operator. Will invert (flip) the input. 1 = 0, 0 = 1
Truth Table	A table which shows <u>all possible</u> inputs and outputs for a Boolean operator.

GCSE Unit 2 – 2.5 – Languages and Integrated Development Environments



IDE's

- Integrated Development Environment (IDE)
  - Editor (for writing the code)
  - Error Diagnostics (such as debug facilities)
  - Run-Time Environment
  - Translators
  - Compiler
  - Syntax Highlighting
  - Code Completion

Interpreters

- ***Translate and execute*** source code line by line.
- Source code is checked for correct syntax – if ok, code is executed. If not interpreting is stopped.
- **NO** executable is produced (it is translated **EVERY TIME** the program is run)
- Less optimisation than compiled code (doesn't look at the code as a whole)
- Slower than compiled code
- Can be used for development (aide debugging)
- Ideal for "platform independent" code

Compilers

- Takes the entire source code (high level program) at once and turns it into Machine Code (binary) creating an "**Executable**" file.
- Compilers will attempt to optimise code
- Slow to compile but only needs to be done once
- Once compiled, generally runs faster than interpreted languages
- If problems occur it will generate error reports

GCSE Unit 2 – 2.5 – Languages and Integrated Development Environments

Key Vocabulary and Definitions

Term	Meaning
High Level Languages	Languages which are written in structured English. Easy for users to understand, require translation to work.
Low Level Languages	Languages which are either binary (machine code) or close to binary (assembly language)
Translators	The process of converting program code into binary machine code instructions
Interpretors	A program which converts high level languages into machine code during run time (one line at a time effectively)
Compiler	A program which converts high level languages into a machine code executable file all in one go.
IDE	Integrated Development Environment. A program which contains a set of tools to allow developers to write, test, debug and execute program code.
Editor	Part of an IDE. The place where code is written. May provide syntax highlighting or code completion.
Error Diagnostics	Tools in an IDE which aid a developer in finding errors, such as tool tips, break points, syntax highlighting, variable watches etc
Run-Time Environment	A tool in an IDE which allows the developer to run the program code they are creating
Code/syntax highlighting	A tool in the code editor which changes the colour of key words depending on their role/function. Helps to read code more easily and spot potential errors/mistakes.
Code Completion	A tool in the code editor which will attempt to predict what the developer is about to type. Speeds up code entry,