Programming in the Digital Technologies curriculum

Algorithms Levels 7 and 8
Introduction

This video will cover

Curriculum Links
Defining a problem
Working out requirements
Designing an algorithm
Curriculum Links

Defining a problem

“Define and decompose real-world problems taking into account functional requirements and sustainability (economic, environmental, social), technical and usability constraints”

“Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors”
Curriculum Links

Defining a problem

“Define and decompose real-world problems taking into account functional requirements and sustainability (economic, environmental, social), technical and usability constraints”

Functional requirements

“Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors”
Curriculum Links

Defining a problem

“Define and decompose real-world problems taking into account functional requirements and sustainability (economic, environmental, social), technical and usability constraints”

Functional requirements

Design an algorithm

“Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors”
Defining a problem

Classic guessing game

Guess a number 1 – 100

Too high  x
Too low  x
Correct  ✓
Defining a problem

Classic guessing game

Problem: “Create a guessing game program”
Requirements

- Store a random number (1-100)
- Give player instructions
- Store the player’s guess
- Check player’s guess
- Guess was correct
- Guess was incorrect
- Guess was too low
- Guess was too high
- Allow the player 7 guesses
- Allow the player to play another game
- Keep score
## Design an Algorithm

### Flowchart Conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="start.png" alt="Start icon" /></td>
<td>Start / End</td>
<td><img src="get_temp.png" alt="get temp icon" /></td>
<td>Input / Output</td>
</tr>
<tr>
<td><img src="set_x_to_5.png" alt="set x to 5 icon" /></td>
<td>Process</td>
<td><img src="process.png" alt="process icon" /></td>
<td>Order of operation</td>
</tr>
<tr>
<td><img src="x_less_than_5.png" alt="x &lt; 5 icon" /></td>
<td>Decision</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Design an Algorithm

**Requirements**
- Store a random number (1-100)
- Give player instructions
- Store the player’s guess
- Check player’s guess
- Guess was correct
- Guess was incorrect
- Guess was too low
- Guess was too high
- Allow the player 7 guesses
- Allow the player to play another game
- Keep score

**Instructions**
```
Start

compNum = random(1-100)

Instructions

Guess = User Input

Check guess

True

Well Done

End
```
Design an Algorithm

Requirements
Store a random number (1-100)
Give player instructions
Store the player’s guess
Check player’s guess
Guess was correct
Guess was incorrect
Guess was too low
Guess was too high
Allow the player 7 guesses
Allow the player to play another game
Keep score

Instructions
Start
randNum = random(1-100)
Instructions
Guess = User Input
Check guess
False
Bad Luck
True
Well Done
End
Design an Algorithm

**Requirements**
- Store a random number (1-100)
- Give player instructions
- Store the player’s guess
- Check player’s guess
- Guess was correct
- Guess was incorrect
- Guess was too low
- Guess was too high
- Allow the player 7 guesses
- Allow the player to play another game
- Keep score
**Requirements**

- Store a random number (1-100)
- Give player instructions
- Store the player’s guess
- Check player’s guess
- Guess was correct
- Guess was incorrect
- Guess was too low
- Guess was too high
- Allow the player 7 guesses
- Allow the player to play another game
- Keep score

**Design an Algorithm**

Start, Random, Instructions, input etc.

- guesses = 0
- While guesses < 7
  - guesses + 1
  - guess = input
    - Too Low
    - Too High
    - Guess Again
  - Correct guess
    - Well Done
  - False
    - Too Low
    - Too High
    - Guess Again
  - True
    - End
- False
  - End
- True
  - End

True

False

Well Done

Bad Luck

Correct guess

End
In this video tutorial we covered

**Curriculum Links**

- **Defining a problem**
  "Define and decompose real-world problems taking into account functional requirements and sustainability (economic, environmental, social), technical and usability constraints."

- **Functional requirements**
  "Design algorithms represented diagrammatically and in English, and trace algorithms to predict output for a given input and to identify errors."

- **Design an algorithm**

**Defining a problem**

- **Classic guessing game**
  - Guess a number 1 – 100
  - Too high ×
  - Too low ×
  - Correct ✓

**Design an Algorithm**

- **Requirements**
  - Store a random number (1-100)
  - Give player instructions
  - Store the player’s guess
  - Check player’s guess
  - Guess was correct
  - Guess was incorrect
  - Guess was too low
  - Guess was too high
  - Allow the player 7 guesses
  - Allow the player to play another game
  - Keep score
Algorithms Part B

The next video tutorial will cover

Creating a “play again” function and keeping score
Representing the algorithm using English instructions
Testing and Tracing