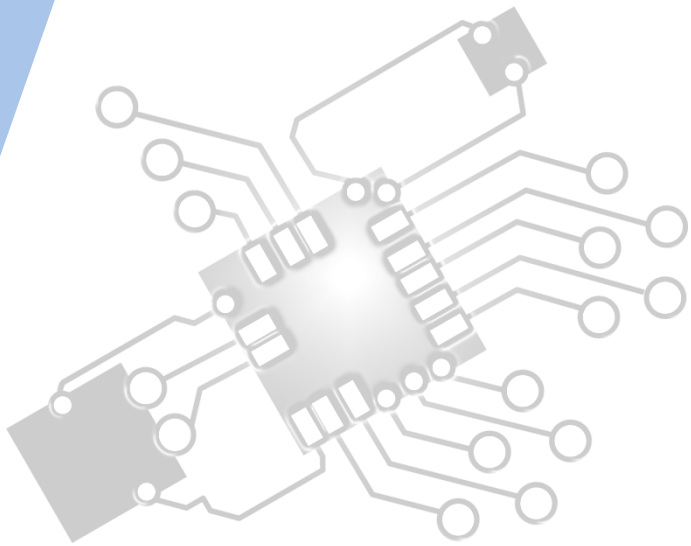
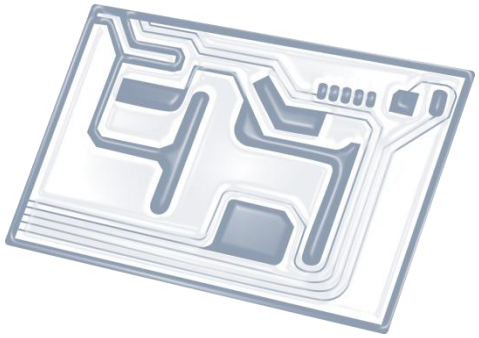
Computational thinking, problem-solving and programming: Introduction to programming

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL & SL 4.3 Overview

Nature of programming languages

4.3.1 State the fundamental operations of a computer

4.3.2 Distinguish between fundamental and compound operations of a computer

4.3.3 Explain the essential features of a computer language

4.3.4 Explain the need for higher level languages

4.3.5 Outline the need for a translation process from a higher level language to machine executable code

Use of programming languages

4.3.6 Define the terms: variable, constant, operator, object

4.3.7 Define the operators =, ., <, <=, >, >=, mod, div

4.3.8 Analyse the use of variables, constants and operators in algorithms

4.3.9 Construct algorithms using loops, branching

4.3.10 Describe the characteristics and applications of a collection

4.3.11 Construct algorithms using the access methods of a collection

4.3.12 Discuss the need for sub-programmes and collections within programmed solutions

4.3.13 Construct algorithms using predefined sub-programmes, one-dimensional arrays and/or collections


1: System design

2: Computer Organisation



3: Networks

4: Computational thinking

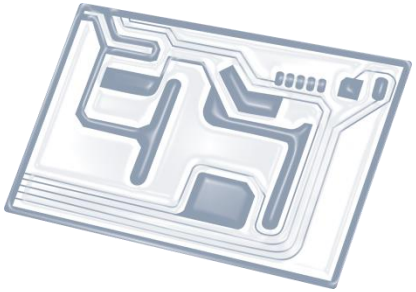

5: Abstract data structures

6: Resource management



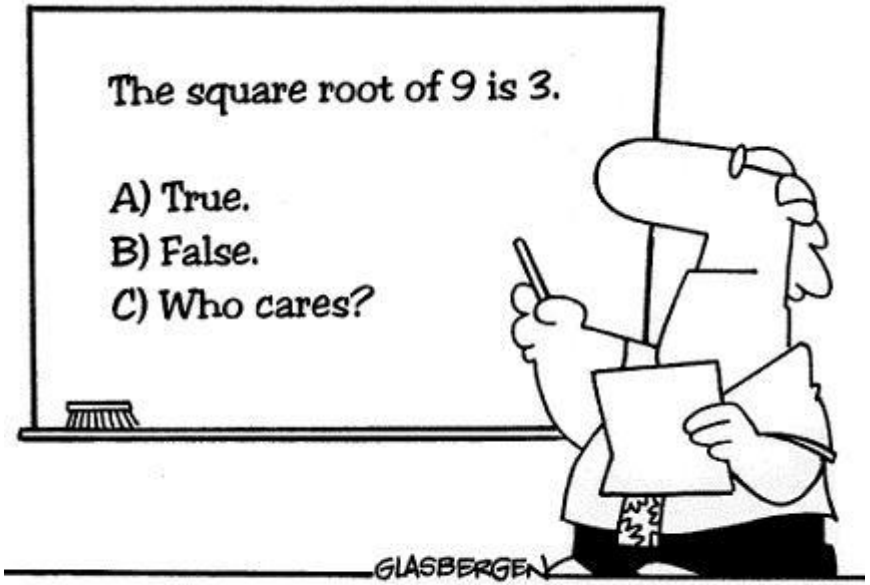
7: Control

D: OOP

Topic 4.3.7

Define the operators: **=**, **≠**, **<**, **<=**, **>**, **>=**, **mod**, **div**



Many students actually look forward to Mr. Atwadder's math tests.

Simple Assignment operator

- = means "*gets the value*"
- *Example*: `int i = 25`
- *Reads*: An integer i gets the value of 25

Arithmetic Operators

- **+** : used for adding two numbers (also used for String concatenation)
- **-** : used for subtractions
- ***** : used for multiplication
- **/** : used for division
- **%** or **mod** : returns the remainder of a division calculation
- **div** : returns the numbers of times X divides into Y without a remainder

Unary operators

- `++` : Increment operator; increments a value by 1
- `i++` is the same as `i = i + 1`
- `--` : Decrement operator; decrements a value by 1
- `i--` is the same as `i = i - 1`

Equality and Relational Operators

- `==` : Equal to (only for non-Strings!)
- `!=` : Not equal to
- `>` : Greater than
- `>=` : Greater than or equal to
- `<` : Less than
- `<=` : Less than or equal to