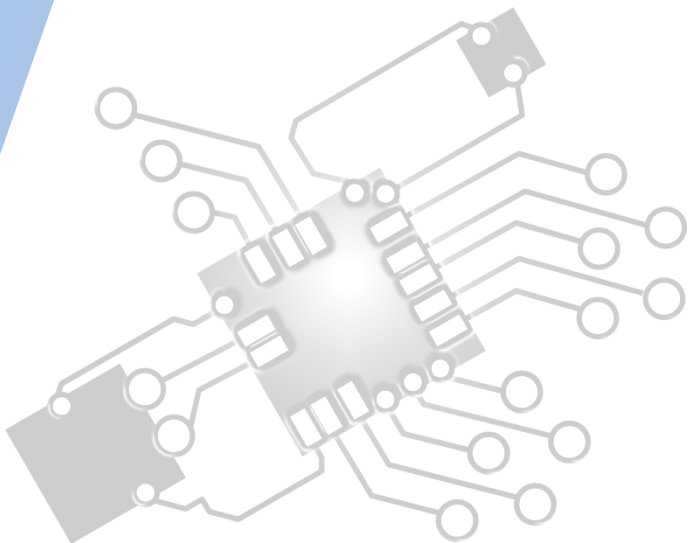




Planning & system installation

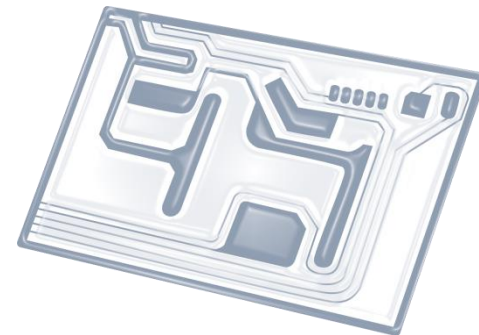
IB Computer Science



*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



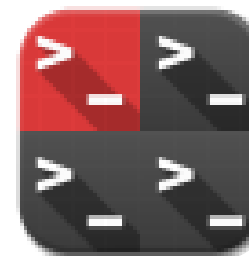
5: Abstract data structures



6: Resource management



7: Control



D: OOP

HL *only* 5 Overview

Thinking recursively

- 5.1.1 Identify a situation that requires the use of recursive thinking
- 5.1.2 Identify recursive thinking in a specified problem solution
- 5.1.3 Trace a recursive algorithm to express a solution to a problem

Abstract data structures

- 5.1.4 Describe the characteristics of a two-dimensional array
- 5.1.5 Construct algorithms using two-dimensional arrays
- 5.1.6 Describe the characteristics and applications of a stack
- 5.1.7 Construct algorithms using the access methods of a stack
- 5.1.8 Describe the characteristics and applications of a queue
- 5.1.9 Construct algorithms using the access methods of a queue
- 5.1.10 Explain the use of arrays as static stacks and queues

Linked lists

- 5.1.11 Describe the features and characteristics of a dynamic data structure
- 5.1.12 Describe how linked lists operate logically
- 5.1.13 Sketch linked lists (single, double and circular)

Trees

- 5.1.14 Describe how trees operate logically (both binary and non-binary)
- 5.1.15 Define the terms: parent, left-child, right-child, subtree, root and leaf
- 5.1.16 State the result of inorder, postorder and preorder tree traversal
- 5.1.17 Sketch binary trees

Applications

- 5.1.18 Define the term dynamic data structure
- 5.1.19 Compare the use of static and dynamic data structures
- 5.1.20 Suggest a suitable structure for a given situation



1: System design

2: Computer Organisation



3: Networks

4: Computational thinking



5: Abstract data structures

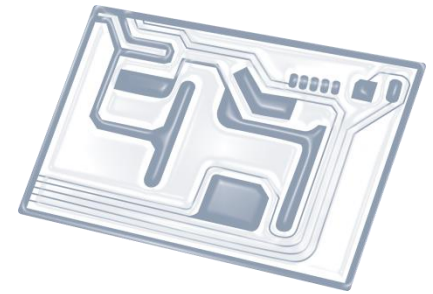
6: Resource management



7: Control

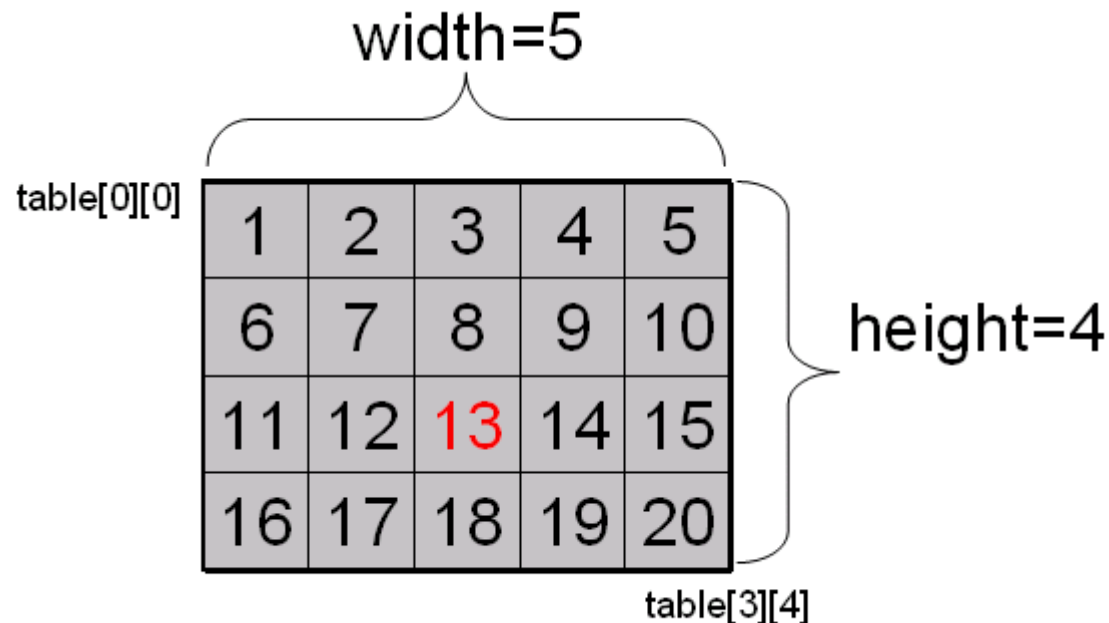
D: OOP





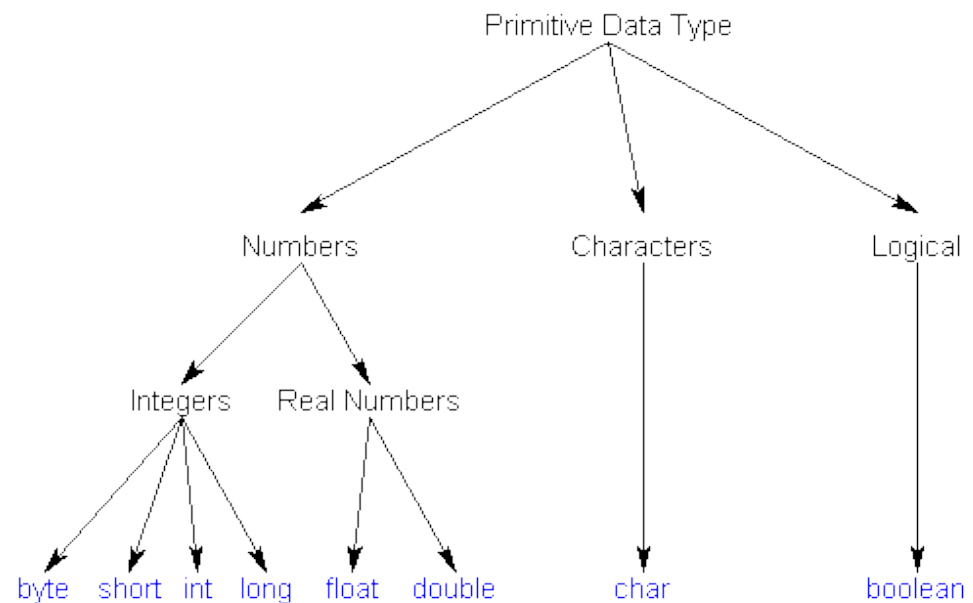
Topic 5.1.4

Describe the **characteristics** of a **two-dimensional array**

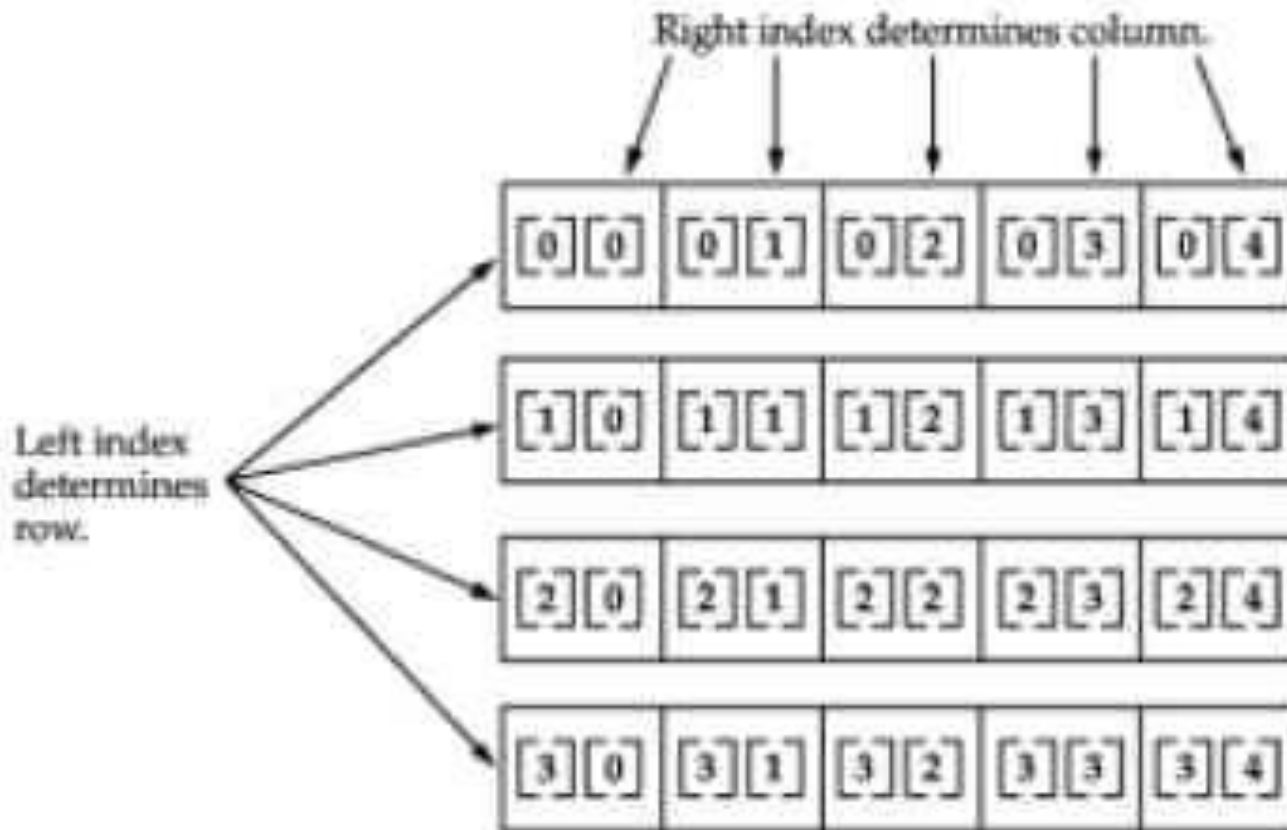


Abstract Data Structures (ADTs)

- **2D array**
- Stack
- Queue
- Linked List
- (Binary) Tree
- Recursion



2D array (grids/tables)



Given: `int twoD[] [] = new int [4] [5];`

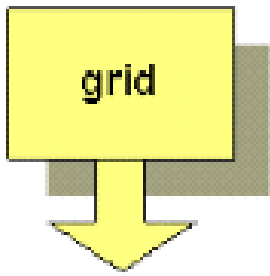
| | | x = | | | | |
|-----|---|-----|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 |
| y = | 0 | 1 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 0 | 2 | 0 | 0 |
| | 2 | 1 | 1 | 0 | 0 | 1 |
| | 3 | 2 | 1 | 1 | 1 | 0 |
| | 4 | 1 | 2 | 1 | 2 | 2 |

Array.At(x,y) = value

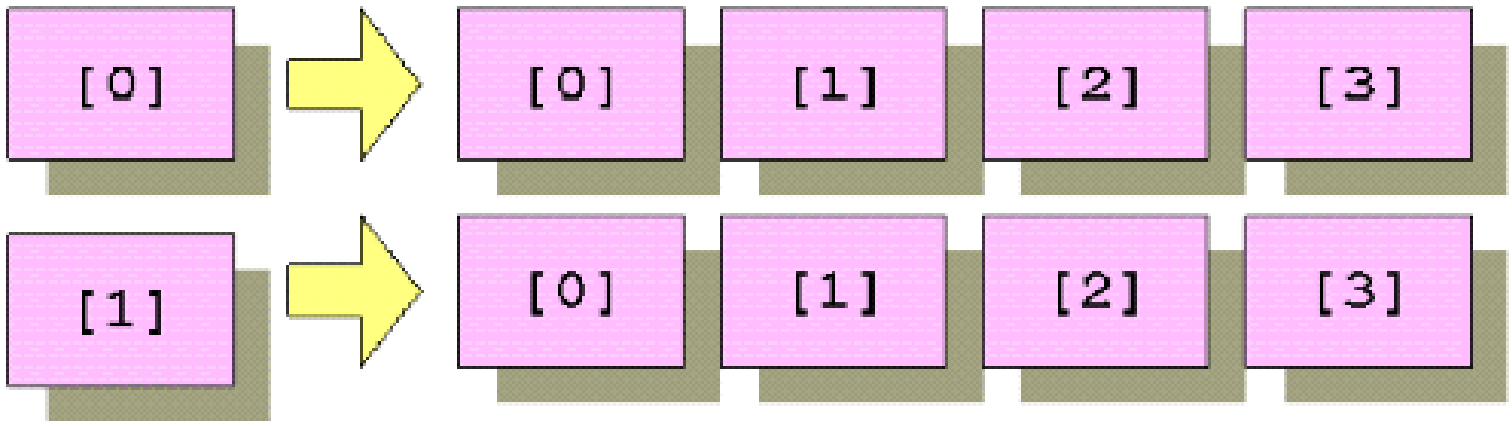
Array.At(0,0) = 1

Array.At(2,3) = 1

Array.At(4, 4) = 2



```
int[][] grid;  
grid = new int[2][4]
```



Example of creating, filling and printing a 2D array in Java using dedicated methods (level 8)

```
int main()
{
    int array[5][5];

    fillArray(array, 5);
    printArray(array, 5);

    return 0;
}
void fillArray(int ar[][5], int numRows)
{
    srand(time(0));
    for(int row = 0; row < numRows; row++)
    {
        for(int col = 0; col < 5; col++)
        {
            ar[row][col] = rand() % 101;
        }
    }
}
void printArray(const int ar[][5], int numRows)
{
    for(int row = 0; row < numRows; row++)
    {
        for(int col = 0; col < 5; col++)
        {
            cout << ar[row][col] << "\t";
        }
        cout << endl;
    }
}
```