Session 8

Arrays

Objectives

- Define Arrays
- List the different types of Arrays
- Describe the different search techniques
- Describe the different sort techniques

Introduction

- The organized collection of objects into rows and columns is known as an array

- It can be defined as a group of variables of the same data type grouped together under a single name

Overview of Array 1-2

- A variable is used to store a piece of information in memory

- Earlier the OS would allocate memory space to the variables at random location, then, the OS would need to gather information from all the scattered variables

- They allow the programmer to store more than one value in a variable, at the same time retaining a single reference

Overview of Array 2-2

- The values in an array are stored in contiguous location in memory and can be accessed by a single name
- One constraint with an array is that they store only data pertaining to one particular data type
- While declaring an array, the data type of the data that is stored needs to be specified and all the elements in it have to be of the same type

Array Example 1-2

- Create a program, which accepts five numbers and display their total. There can be two methods to proceed:
  - In example 1, declare five variables to store five numbers, and one variable to store the total.

Example 1

```
BEGIN
DECLARE num1, num2, num3, num4, num5 and sum as integers
ACCEPT num1
ACCEPT num2
ACCEPT num3
ACCEPT num4
ACCEPT num5
Sum=num1+num2+num3+num4+num5
Display sum
End
```

Array Example 2-2

- In example 2 an array is used

Example 2

```
BEGIN
ARRAY arrNums [3] is an integer
arrNums [0] = 2
arrNums [1] = 4
arrNums [2] = 7
total=arrNums [0] +arrNums [1] +arrNums[2]
DISPLAY total
END
```

Advantages of Arrays

- Advantages of using an array are as follows:
  - Reduction in the number of variable names
  - Selection of the variable based on the value of the variable
  - Storage of the entire data sets for multiple time use in a program
  - Declaration of fixed length data set
  - Access the data in any order or at random

Declaring an Array

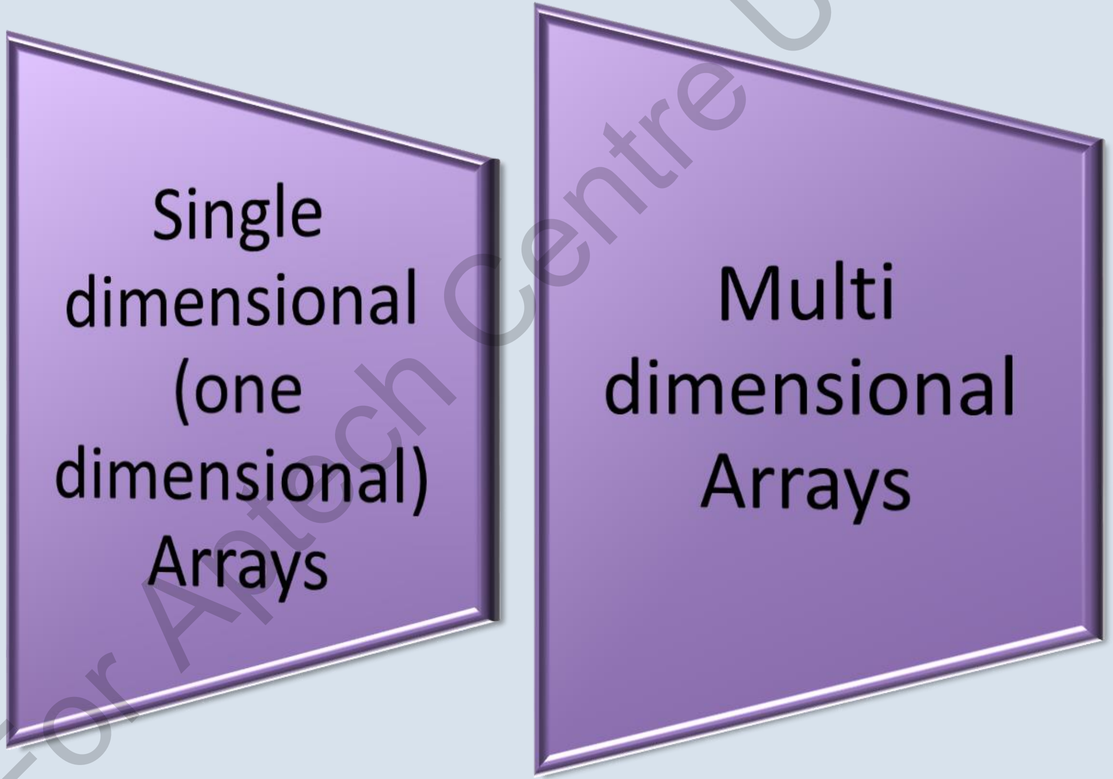
- In example 2, an array is declared with the statement, **`ARRAY arrNums[3]`**
- **ARRAY** is the keyword used in an algorithm to declare an array
- **`arrNums`** is the name of the array, while **`[3]`** indicates the size of the array.

Syntax:

```
arrNum [element number] = value
```

Different Type of Arrays

- Arrays can be divided into two categories:



Single Dimensional Arrays 1-2

- ❑ Single dimensional arrays are the simplest form of arrays and it is a type of linear array
- ❑ All items in one dimensional array are stored in a row starting from zero to the size of array
- ❑ To access an element in one dimensional array, a single subscript is used which can either represent a row or column index

Single Dimensional Arrays 2-2

- The different components of one dimensional array are as follows:
  - A name
  - A data type
  - A size

Multi Dimensional Arrays 1-4

- Sometimes data has to be stored and manipulated in a tabular format in the form of a matrix
- For example, consider the data provided in table, which represents marks scored by 3 students in 2 different subjects

Name	Physics	Chemistry
John	45	60
Mathew	20	67
Ronald	90	35

Multi Dimensional Arrays 2-4

- If the result of the highest scored student in all subjects needs to be realized then, one way is by finding out who got highest in physics and chemistry
- If several operations are required to be carried out on the data, then these data needs to be stored in the memory
- The solution to this problem is multi dimensional array, where array would resemble the structure as shown in table

Multi Dimensional Arrays 3-4

- Multi dimensional arrays resembles a matrix, and has rows and columns.
- Multi dimensional arrays are implemented as arrays of arrays.

	0	1
0	45	60
1	20	67
2	90	35

Syntax:

```
ARRAY arrMarks [element number  1][element number 2]
```

Multi Dimensional Arrays 4-4

Example 4

```
FOR i IN RANGE 0 TO 1
DO
FOR j IN RANGE 0 TO 2
DO
ACCEPT arrMarks [i][j]
END DO
END DO
```

Different Search Techniques

- Searching refers to the operation of finding the location of a specific item in a group of items.

- The different search algorithms are as follows:

Sequential Search

Binary Search

Sequential Search 1-2

- Sequential or linear search is the simplest form of search algorithm

- In this technique, search starts at the first element and continues until either the item is found or end of the list is reached

Sequential Search 2-2

- The algorithm for a sequential or linear search is as follows:

```
INPUT: Array of Size N. Target Value T
OUTPUT: Position of T in the list-1
BEGIN
Set FOUND = false
Set I = 0
While (I <= N) and (FOUND is false)
IF List[i] == T THEN
FOUND = true
ELSE
I = I+1
END
IF FOUND==false THEN
T is not present in the List
END
```

Binary Search 1-3

- Binary search is the best search algorithm for a sorted array
- It is a powerful technique for searching an ordered list
- The concept is similar to the way people look for an entry in a dictionary or telephone book

Binary Search 2-3

- In binary search the data set is split into half and the middle item value is compared with the search value
- If the search value is smaller than the middle item then the first half of the data set or list is searched
- This continues until the search value is located or the remaining list consists of only one item

Binary Search 3-3

- The algorithm for a binary search is as follows:

```
BOTTOM = first element
TOP = last element
WHILE ((TOP>=BOTTOM) and (not found)) loop
MID= (TOP + BOTTOM)/2
IF (LIST(MID) = item to find) THEN
FOUND = true
ELSE IF (item to find > LIST(MID) then
BOTTOM = MID+1
ELSE
TOP = MID - 1
END IF
END loop
IF FOUND = true
Wanted item is in database
ELSE
Wanted item is NOT in database
END IF
```

Different Sort Techniques

- The function of sorting or ordering a list of objects according to some linear order is very fundamental
- Sorting algorithm arranges the elements of a list in a sorted order
- The two types of sorting methods are as follows:

Internal sort

External sort

Internal Sort 1-2

- Internal sorting takes place in the main memory of the computer

- This is possible when the data collection to be sorted is small and can be accommodated in the main memory

- The programmer benefits in this type of sorting because of the random access nature of the main memory

Internal Sort 2-2

- The different types of internal sorts are as follows:

Selection sort

Quick sort

Bubble sort

Insertion sort

Selection Sort

- Selection sort is a very simple sorting algorithm
- It works by following these steps:
  - Find the minimum value in the list by iterating over the whole list
  - Swap this value with the first value in the list
  - Repeat these two steps, but each time decrease the list by starting with the second position
- By following these steps, the list is divided into two parts: a sorted part, the part to the left and an unsorted part, the part to the right

Quick Sort

- Quick sort divides the lists into two sub lists, and then sorts the sub lists

Pick an element from the list. This element is called the pivot value.

Reorder the list so that all elements smaller than the pivot value are positioned before the pivot value, and all the larger elements after the pivot value. This operation is called partitioning.

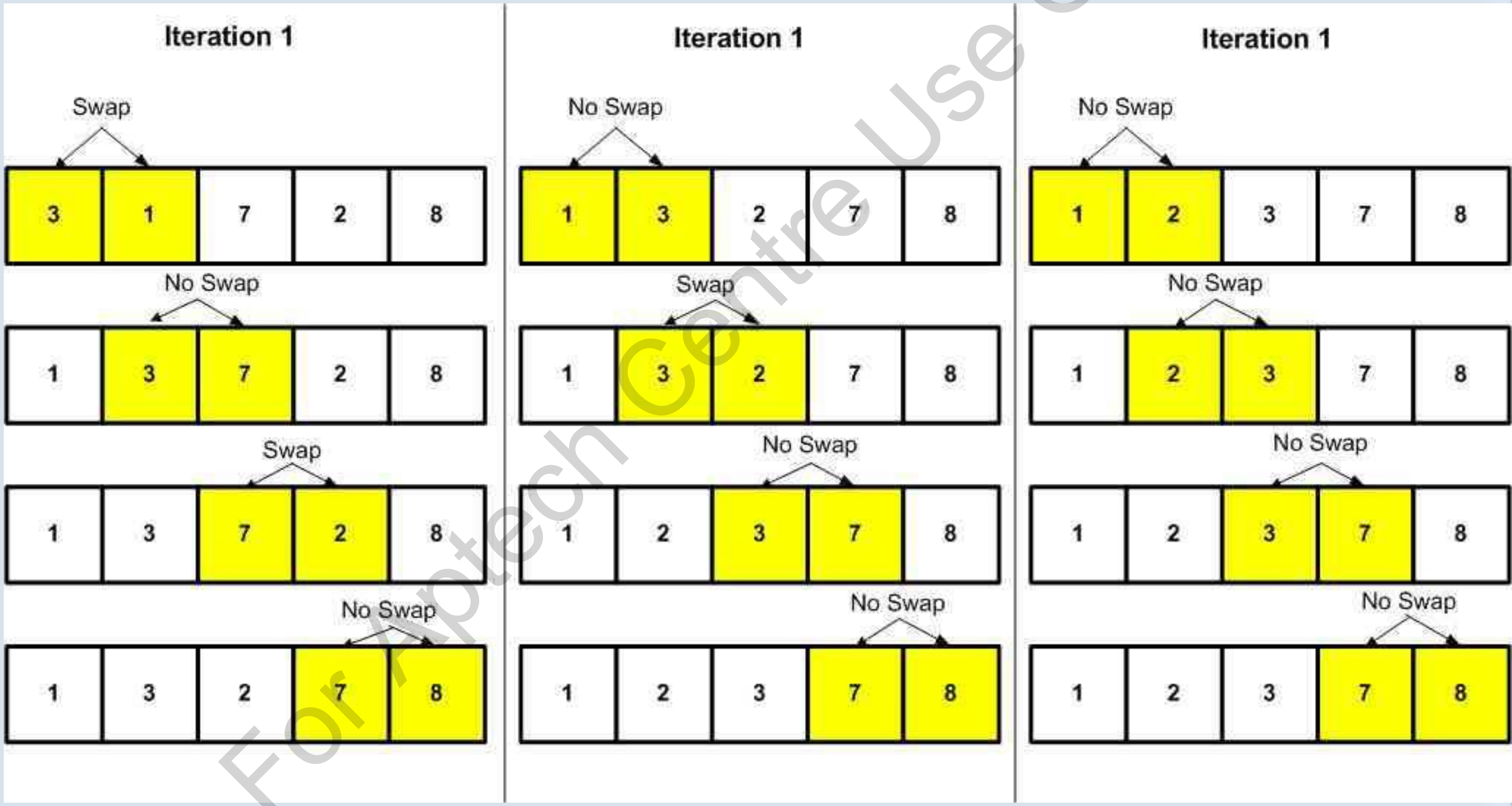
Consider the two newly created partitions, place the inner lists before and after the pivot value, as separate lists. Recursively sort these with quick sort, which means beginning at step one again.

Bubble Sort 1-3

- It works by repeatedly iterating through a list, comparing two adjacent elements at a time and swapping them wherever necessary

- If an iteration through the list makes no further swaps, then the sorting is considered to be complete

Bubble Sort 2-3



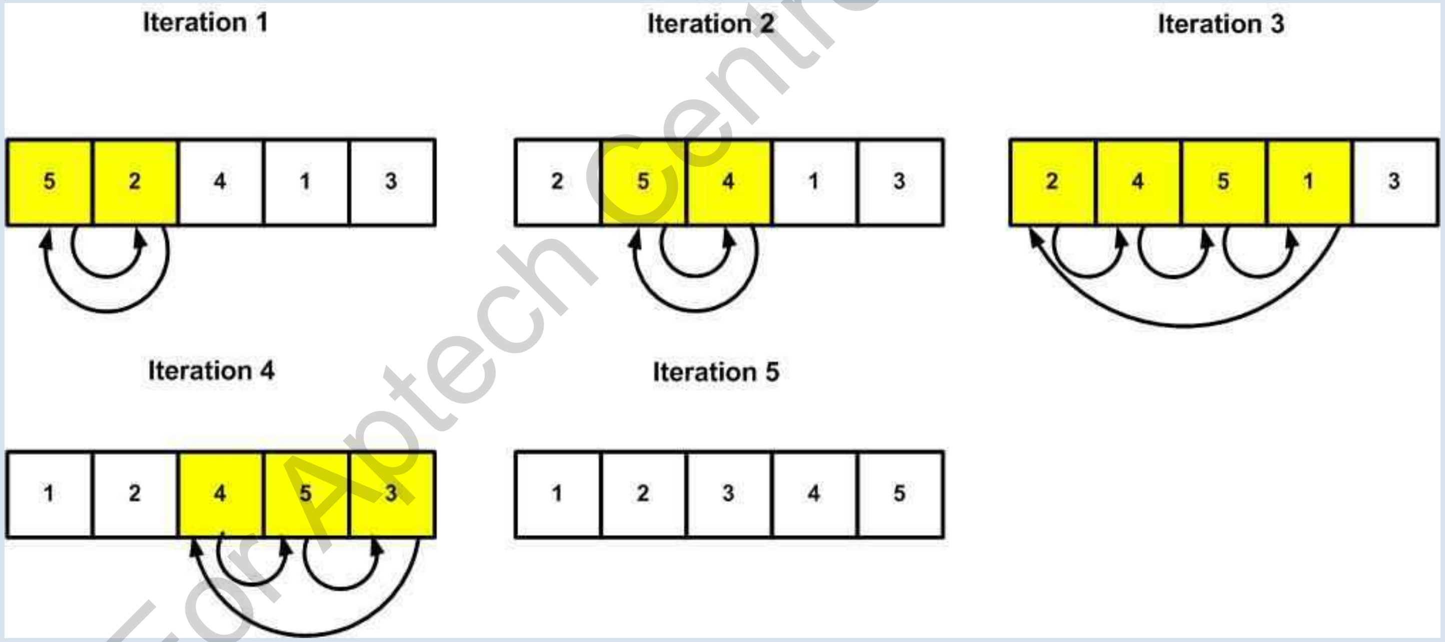
Bubble Sort 3-3

- The algorithm for a bubble sort is as follows:

```
Procedure bubbleSort( A : list of sortable items )
REPEAT
SWAPPED = false
FOR i = 1 to length(B) - 1 inclusive do:
IF B[i-1] > B[i] THEN
SWAP(B[i-1], A[i])
SWAPPED = true
END IF
END FOR
UNTIL not SWAPPED
END Procedure
```

Insertion Sort 1-2

- Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time



Insertion Sort 2-2

- Characteristics of insertion sort are as follows:

  - Simple to implement
  - Efficient for small data sets
  - Efficient for data sets that are already sorted
  - More efficient in practice than most other algorithms such as selection sort or bubble sort
  - It is stable and does not change the relative order of elements with equal keys
  - It can sort a list as it receives the list

External Sort

- The examples of external sorting are as follows:
  - Sorting with Disk
  - Sorting with Tapes

Summary 1-2

- Arrays are an essential part of programming, as they allow the programmer to store more than one value in a variable, at the same time retaining a single reference
- Array can be defined as a collection of elements of same type that are referenced by a common name
- All items in a one dimensional array are stored either in a row or column and indexing starts from zero and ends with the size of the array minus one
- Most languages support multi dimensional arrays, where instead of storing the data in a single dimension, it can be stored in more than one dimension
- External sorting is necessary when the number and size of objects are large and cannot be accommodated in the main memory

Summary 2-2

- Searching refers to the operation of finding the location of a specific item in a group of items. The different search algorithms are as follows:
  - Sequential Search
  - Binary Search
  - Binary Tree Search
- Internal sorting takes place in the main memory when the data to be sorted is small. The different types of internal sorts are as follows:
  - Selection sort
  - Quick sort
  - Bubble sort
  - Insertion sort