

Tuesday, 3 January 2023 6:51 PM

DEC X : INT

Name 1:
 Name 2:
 Name 3:
 ⋮
 Name 12:

12 DBC
12 INIT
12 Assignments

Diagram illustrating an array structure:

index	Element
1	0
2	0
3	0
4	0
5	80
6	0
7	0
8	0
9	0
10	0
11	0
12	0

Labels and Annotations:

- index**: Points to the index column.
- lower bound**: Points to index 1.
- upper bound**: Points to index 12.
- Element**: Points to the value 80 at index 5.

```
DECLARE x : ARRAY[1:12] OF INT
```

Handwritten diagram illustrating array indexing and memory addresses. The diagram shows a memory layout with addresses 1200, 1204, 1208, and 1212. At address 1200, the value 30 is stored. At address 1204, the value 5 is stored. At address 1208, the value 30 is stored. At address 1212, the value 0 is stored. The diagram includes arrows indicating the flow of data and the relationship between the array elements and their memory addresses. A yellow box highlights the value 5 at address 1204, and another yellow box highlights the value 30 at address 1208. A yellow arrow points from the word "Subscript" to the value 5. The text "For i ← 1 TO 12" and "NEXT" are written below the memory layout.

1. Declarations
2. Initialisation
3. Population (input to every element in a loop)
4. Searching (Linear Search)
5. Sorting (Bubble Sort)

grades	
1	C
2	A
3	B
4	C
5	D
6	U
7	E
8	A

2. DECLARATION:

```
DECLARE grades : ARRAY [1:8] OF CHARACTER
```

2. INITIALISE:

```
FOR i ← 1 TO 8
    grades[i] ← ""
NEXT
```

3. Population:

```

FOR i ← 1 TO 8
  INPUT theGrade
  grades[i] ← theGrade
NEXT i

```

} For i ← 1 TO 8
 INPUT grades[i]
 NEXT i

4. Linear Search:

```

INPUT "Enter grade to search for: ", gde
isFound ← FALSE // Flag
FOR i ← 1 TO 8
    IF grades[i] = gde
        THEN
            OUTPUT "Grade found at location:", i
            isFound ← TRUE
        ENDIF
    NEXT i
IF isFound = FALSE
    THEN
        OUTPUT "Grade not found!"
    ENDIF

```

5. Bubble Sort

$\rightarrow c = a$
 $a = b$
 $b = c$

Bubble

For $y \leftarrow 1$ TO 9 Step 1
Sorted = TRUE
For $i \leftarrow 1$ TO 8 y

```

IF Numbers[ i ] > Numbers[ i+1 ] THEN
    Temp ← Numbers[ i ]
    Numbers[ i ] ← Numbers[ i+1 ]
    Numbers[ i+1 ] ← Temp
    Sorted = FALSE
ENDIF

```

NEXT i
IF Sorted = TRUE THEN EXIT FOR
NEXT Y

<u>7</u> 1	<u>7</u> 2	<u>7</u> 3	<u>7</u> 4	<u>7</u> 5	...	<u>7</u> 7
<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>	<u>1</u>		<u>1</u>
1 ✓ 9	1 9	1 1	1 1	1 1		1 1
2 ✓ 30	2 30 1	2 1	2 9 1	2 9		2 9
3 ✓ 140 1	3 1 30 12	3 1	3 12	3 12		3 12
4 ✓ 1 40 12	4 12 30 20	4 1	4 20	4 20		4 20 2
5 ✓ 1 40 20	5 20 30	5 1	5 30	5 20 2		5 1 20
6 ✓ 20 40	6 40	6 1	6 40 2	6 30		6 30
7 ✓ 80 50	7 50 2	7 1	7 40	7 40		7 40
8 ✓ 50 80 2	8 50	8 1	8 50	8 50		8 50
9 80	9 80	9 1	9 80	9 80		9 80

8 7 6 5 4