

EOY Sth Else

Basic components of a program

1. Array

- **Reasons of usage**

- to store a large collection of values
- to reduce the need of declaring multiple variables with various identifier, and therefore reduce codes

- **Attributes**

- stores data in **continuous memory space**
- stores multiple values of the **same data type**
- if it is an static array, the size of the array is initialized at the start and **cannot be resized** throughout running the program

2. Constant

- **Reasons of usage**

- to make the readers know what the value is referring to
 - e.g. a single value of 30 is not easily identified, but if we assigned "num of stu" as 30, then the readers would know what this 30 refers to
- to easily call this value when needed, especially when the value is large
 - e.g. if you used a constant "Cost" to store a large value such as 1,000,000,000,000, you then don't need to type all those zeros when needing the value, and the value could be called through simply type in "Cost".
- it cannot be altered when the throughout running the program

3. Meaningful identifier

- **Reason of usage**

- help the developer and other programmers to comprehend the program at present or later

4. Indentation

- **Reason of usage**

- easier to identify the blocks of codes
 - e.g. For loop / Selection(if)
- thus easier to debug and understand the program

- **Requirements**

- Indent 4 places writing both pseudocode and Python in the exam!

5. Comments

- A description of a line of code, or a section of code

- **Reason of usage**

- telling the developer and other programmers what this program or this specific line means
- thus easier to debug and understand the program

6. Variables

- **Reasons of usage**

- to store the values inputted
- if variables are said to be altered when the program is running, variables are used to give the value a meaningful identifier for the sake of ease to understand and read

Pseudocode

- Remember to indent 4 spaces in the exam

1. Linear Search

```
// If isn't required to terminate when the serch value is found
FOR Index <- 1 TO LENGTH(Fruit)-1
  IF Array[Index] = SearchValue THEN
    OUTPUT "Search value is found at" & Index
  ELSE
    ArrayIndex <- ArrayIndex + 1
  ENDIF
NEXT Index
// If is required to terminate once the search value if found
Found <- FALSE
Index <- 0
While Found = FALSE AND Index < LENGTH(Array) DO
```

```
IF Array[Index] = SearchValue THEN
    OUTPUT "Search value is found at" & Index
    Found <- TRUE
ELSE
    ArrayIndex <- ArrayIndex + 1
ENDIF
ENDWHILE
```

2. Exchanging Values

```
If a > b THEN
    Temp <- b
    b <- a
    a <- Temp
ENDIF
```

3. Bubble Sort

```
DECLARE Letters : Array[0:3] OF CHAR
Letters <- ["A", "B", "D", "C"]
FOR Outer <- 0 TO LENGTH(Letters)-2
    FOR Inner <- 0 TO LENGTH(Letters)-2
        IF Letters[Inner] > Letters[Inner+1] THEN
            Temp <- Letters[Inner]
            Letters[Inner] <- Letters[Inner+1]
            Letters[Inner+1] <- Temp
        ENDIF
    NEXT Inner
NEXT Outer
```

4. Finding Maximum

```
DECLARE Nums : Array[0:3] OF INTEGER
Nums <- [3, 4, 1, 2]
Maxn <- -1 // should not max here as the identifier
FOR X <- 0 TO LENGTH(Nums)-1
    IF Nums[X] > Maxn THEN
        Maxn <- Nums[X]
    ENDIF
NEXT X
```