UNIT- FOUR
LIST

Topics to be Covered

- Introduction
- Static and Dynamic list structure
- Array implementation of List
- Queue as a list

Introduction: List

- The list can be defined as an abstract data type in which the elements are stored in an ordered manner for easier and efficient retrieval of the elements.
- List Data Structure allows repetition that means a single piece of data can occur more than once in a list. In the case of multiple entries of the same data, each entry of that repeating data is considered as a distinct item or entry.
- It is very much similar to the array but **the major difference between the array and the list data structure is that array stores only homogenous data in them whereas the list (in some programming languages) can store heterogeneous data items in its object.** List Data Structure is also known as a sequence.
- Common Example: list of books, list of goods at grocery, list of physical infrastructure. etc

List Implementation

1. Static implementation(Array)
2. Dynamic Implementation(Linked list)

Static(Array) implementation to list

- In Static implementation of List, elements are stored in contiguous array position. Once array is created, its size can't be changed.

Advantage of array implementation:

- Fast, random access of elements
- Very little memory is required

Disadvantage of array implementation :

- Fixed Size
- If more data is to be stored, it can't be performed
- If less data is to be stored, space may be left unused
- Slow deletion and insertion of elements

Basic Operations

1. Insertion:
   - At the end of array
   - At the beginning of Array
   - At the given Position
   - In the sorted Array
2. Deletion
   - From the end
   - From the beginning
   - Deletion of given element
   - Deletion from given position
3. Traversing a List: Displaying individual element of array
4. Concatenation: Combine two array into the third array
5. Merging: Merge two sorted array into the third sorted array

Traversing Operation

- Process of visiting each element of the list- starting from the first element up to the last element

Algorithm:

Step 1: Start

Step 2: LOOP I=LB to UB

PRINT LIST[I]

Step 3: Stop

Insertion at the end

Step 1: Start

Step 2: IF UB==MAX-1

THEN WRITE “OVERFLOW” STOP

Step 3: READ DATA

Step 4: UB← UB+1

LIST[UB]← DATA

Step 5: Stop

Insertion at the beginning

Step 1: Start

Step 2: IF UB==MAX-1

THEN WRITE “OVERFLOW” STOP

Step 3: READ DATA

Step 4: K←UB

Step 5: Repeat step 6 while K>=LB

Step 6: LIST[UB+1]← LIST[UB]

K←K-1

Step 7: LIST[LB]←DATA

Step 8: STOP

Insertion at the given position

Input: 'POS' is the given position of array where new data item is inserted

Step 1: Start

Step 2: IF UB==MAX-1

THEN WRITE "OVERFLOW" STOP

Step 3: READ DATA

Step 4: READ Position 'POS' where data will be inserted

Step 5: K← UB

Step 6: REPEAT STEP 7 while K>=POS

Step 7: LIST[K+1]← LIST[K]

K← k-1

STEP 8: LIST[POS] ← DATA

Step 9: STOP

Insertion in the sorted array

Input: LIST is a sorted array, UB is the current upper bound, 'DATA' will be inserted at right position in sorted array.

Step 1: Start

Step 2: IF UB==MAX-1

THEN WRITE "OVERFLOW" STOP

Step 3: READ DATA to be inserted

Step 4: K ← LB

Step 5: Repeat step 6 while (LIST[K]<DATA)

Step 6 K←K+1

Step 7: POS← K

Step 8: **K← UB**

STEP 9: REPEAT STEP 10 while K>=POS

Step 10: LIST[K+1]← LIST[K]

K← k-1

STEP 11: LIST[POS] ← DATA

Step 12: STOP

Deletion from the end

Step 1: Start

Step 2: IF UB<0

THEN WRITE “UNDERFLOW” and STOP

Step 3: LIST[UB]← NULL

Step 4: UB← UB-1

Step 5: Stop

Deletion From the beginning

Step 1: Start

Step 2: IF UB<0

THEN WRITE “UNDERFLOW” STOP

Step 3: K←LB

Step 4: Repeat step 5 while K<UB

Step 5: LIST[K]← LIST[K+1]

K←K+1

Step 6: LIST[UB]←NULL

UB← UB-1

Step 7: STOP

Deletion of the given element

Input: element 'DATA' to be deleted

Step 1: Start

Step 2: IF UB<0

THEN WRITE "Underflow" STOP

Step 3: READ DATA as element to be deleted

Step 4: K←LB

Step 5: REPEAT STEP 6 while LIST[K]≠DATA

Step 6 : K←K+1

STEP 7: REPEAT STEP 8 while K<UB

Step 8: LIST[K]← LIST[K+1]

K←K+1

STEP 9: LIST[UB] ← NULL

UB←UB-1

Step 10: STOP

Concatenation

Input: Two arrays A and B are combined together in third array C

Step 1: Start

Step 2: K← LBA

Step 3: Repeat step 4 while K<=UBA

Step 4: C[K]←A[K]

K← K+1

Step 5: L←LBB

Step 6 : Repeat step 7 while L<= UBB

STEP 7: C[K]←B[L]

K← K+1

L← L+1

Step 8: STOP

Merging

Input: Two sorted arrays A and B are combined together in third array C which is also in the sorted form

Step 1: Start

Step 2: I← LBA, J← LBB, K← 0

Step 3: Repeat step 4 to 5 while I<=UBA AND J<=UBB

Step 4: If A[I] < B[J]

C[K]←A[I], I← I+1

ELSE

C[K]←B[J], J←J+1

Step 5: K← K+1

STEP 6: REPEAT 7 WHILE I<=UBA

Step 7 : C[K]←A[I], I← I+1, K← K+1

STEP 8: REPEAT STEP 9 while J<= UBB

STEP 9: C[K]←B[j], J←J+1, K← K+1

Step 10: STOP

Any Query?