Fundamentals of
Data Structures using C

List ADT

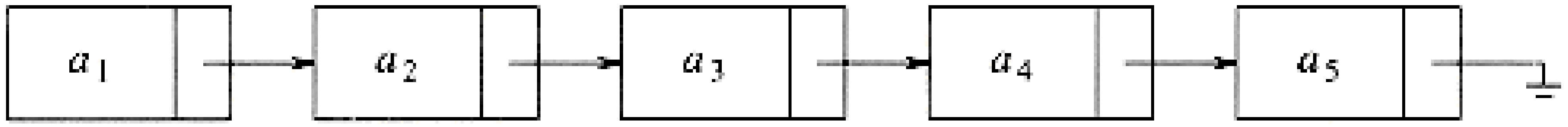
B.Bhuvaneswaran, AP (SG) / CSE

9791519152
bhuvaneswaran@rajalakshmi.edu.in

RAJALAKSHMI
ENGINEERING COLLEGE

Introduction

- List is an ordered set of elements.
- For any list except the empty list, we say that A_{i+1} follows (or succeeds) A_i ($i<N$) and that A_{i-1} precedes A_i ($i>1$).
- The first element of the list is A_1, and the last element is A_N.
- Some popular operations are printList, makeEmpty, find, insert, remove, findKth, next, previous.



Operations of the List ADT

- printList – contents of the list is displayed
- makeEmpty – makes the list empty
- find – returns the position of the first occurrence of an item
- insert – insert some element from some position in the list
- remove – remove some element from some position in the list
- findKth – returns the element in some position (specified as an argument)
- next – return the position of the successor
- previous – return the position of the predecessor

Different Ways to Implement List

- Array
- Linked list

Arrays

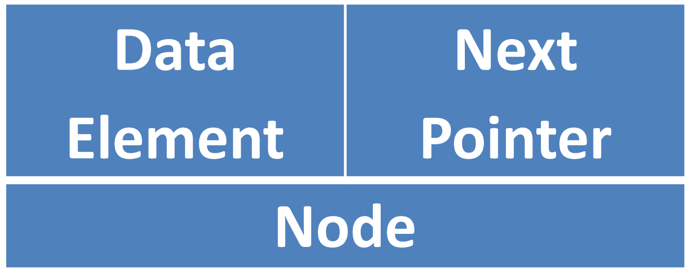
- An array implementation allows print_list and find to be carried out in linear time, which is as good as can be expected, and the find_kth operation takes constant time.
- However, insertion and deletion are expensive. For example, inserting at position 0 (which amounts to making a new first element) requires first pushing the entire array down one spot to make room, whereas deleting the first element requires shifting all the elements in the list up one, so the worst case of these operations is O(n).

Arrays

- On average, half the list needs to be moved for either operation, so linear time is still required. Merely building a list by n successive inserts would require quadratic time.
- Because the running time for insertions and deletions is so slow and the list size must be known in advance, simple arrays are generally not used to implement lists.

Linked List

- The linked list consists of a series of structures, which are not necessarily adjacent in memory.
- Each structure contains the element and a pointer to a structure containing its successor.
- We call this the next pointer.
- The last cell's next pointer points to NULL.



Advantages of Linked Lists

- Linked lists facilitate dynamic memory management by allowing elements to be added or deleted at any time during program execution.
- The use of linked lists ensures efficient utilization of memory space as only that much amount of memory space is reserved as is required for storing the list elements.
- It is easy to insert or delete elements in a linked list, unlike arrays, which require shuffling of other elements with each insert and delete operation.

Disadvantages of Linked Lists

- A linked list element requires more memory space in comparison to an array element because it has to also store the address of the next element in the list.
- Accessing an element is a little more difficult in linked lists than arrays because unlike arrays, there is no index identifier associated with each list element.
- Thus, to access a linked list element, it is mandatory to traverse all the preceding elements.

Types of Linked List

- Depending on the manner in which its nodes are interconnected with each other, linked lists are categorized into the following types:
  - Singly linked list
  - Doubly linked list
  - Circular linked list

Queries?

Thank You!