

DATA STRUCTURES USING “C”

For
BCA Part-II (Session 2018-21) Students

BY

ANANT KUMAR
MCA, M. Phil, M. Tech.
Faculty Member
Department of Computer Science
J. D. Women’s College, Patna

Chapter 3: LINKED LISTS

A linked list is a linear collection of data elements. These data elements are called nodes. Linked list is a data structure which is used to implement other data structures. It acts as a building block to implement data structures like stacks, queues, trees, graphs etc.

Linked lists allocate memory for each element separately and only when necessary whereas arrays allocate the memory for all its elements in one block.

A linked list is a non-sequential collection of data items. It is a dynamic data structure. For every data item in a linked list, there is an associated pointer that would give the memory location of the next data item in the linked list.

The data items in the linked list are not in consecutive memory locations. They may be anywhere, but the accessing of these data items is easier as each data item contains the address of the next data item.

Advantages of linked lists:

Linked lists have many advantages. Some of the very important advantages are:

1. Linked lists are dynamic data structures. i.e., they can grow or shrink during the execution of a program.
2. Linked lists have efficient memory utilization. Here, memory is not pre-allocated. Memory is allocated whenever it is required and it is de-allocated (removed) when it is no longer needed.
3. Insertion and Deletions are easier and efficient. Linked lists provide flexibility in inserting a data item at a specified position and deletion of the data item from the given position.
4. Many complex applications can be easily carried out with linked lists.

Disadvantages of linked lists:

1. It consumes more space because every node requires an additional pointer to store address of the next node.
2. Searching a particular element in list is difficult and also time consuming.

Types of Linked Lists:

1. Single Linked List.
2. Double Linked List.
3. Circular Linked List.
4. Circular Double Linked List.

A single linked list is one in which all nodes are linked together in some sequential manner. Hence, it is also called as linear linked list.

A double linked list is one in which all nodes are linked together by multiple links which helps in accessing both the successor node (next node) and predecessor node (previous node) from any arbitrary node within the list. Therefore each node in a double linked list has two link fields (pointers) to point to the left node (previous) and the right node (next). This helps to traverse in forward direction and backward direction.

A circular linked list is one, which has no beginning and no end. A single linked list can be made a circular linked list by simply storing address of the very first node in the link field of the last node.

A circular double linked list is one, which has both the successor pointer and predecessor pointer in the circular manner.

Applications of linked list:

1. Linked lists are used to represent and manipulate polynomial. Polynomials are expression containing terms with non-zero coefficient and exponents. For example:

$$P(x) = a_0X^n + a_1X^{n-1} + \dots + a_{n-1}X + a_n$$

2. Represent very large numbers and operations of the large number such as addition, multiplication and division.
3. Linked lists are to implement stack, queue, trees and graphs.
4. Implement the symbol table in compiler construction

The linked list code will depend on the following functions:

malloc() is a function which allocates a block of memory in the "heap" and returns a pointer to the new block. The prototype of malloc() and other heap functions are in stdlib.h. malloc() returns NULL if it cannot fulfill the request. It is defined by:

```
void *malloc (number_of_bytes)
```

Since a void * is returned the C standard states that this pointer can be converted to any type. For example,

```
char *cp;  
cp = (char *) malloc (100);
```

Attempts to get 100 bytes and assigns the starting address to cp.

We can also use the sizeof() function to specify the number of bytes. For example,

```
int *ip;  
ip = (int *) malloc (100*sizeof(int));
```

free() is the opposite of malloc(), which de-allocates memory. The argument to free() is a pointer to a block of memory in the heap — a pointer which was obtained by a malloc() function. The syntax is:

```
free (ptr);
```

The advantage of free() is simply memory management when we no longer need a block.

Comparison between array and linked list:

ARRAY	LINKED LIST
Size of an array is fixed	Size of a list is not fixed
Memory is allocated from stack	Memory is allocated from heap
It is necessary to specify the number of elements during declaration (i.e., during compile time).	It is not necessary to specify the number of elements during declaration (i.e., memory is allocated duringrun time).

It occupies less memory than a linked list for the same number of elements.	It occupies more memory.
Inserting new elements at the front is potentially expensive because existing elements need to be shifted over to make room.	Inserting a new element at any position can be carried out easily.
Deleting an element from an array is not possible.	Deleting an element is possible.

5.