PROLOG ACADEMY

DATA STRUCTURE

By- MASEERA ALI

 Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

What actually Data structure is?

- Data is values or set of values
- Structure is organisation
- Data may be organised in many different ways
- The logical and mathematical model of organization of data is known as Data Structure.
- Algorithm + Data structure = Program
- Ex. of storing marks of 100 students
- Overview of some of the Data Structure -

Arrays	Linked Lists	Trees
Stack	Queue	Graph

Data Structure operations-

Traversing
Searching
Inserting
Deleting

Why do we need appropriate Data structure?

To reduce the time complexity
Memory management
**Will study time complexity later

Types-

Linear - Arrays,LinkedLists
Nonlinear - Trees , Graphs

Array

Defined as “ int a[10]; ” here 10 is called the subscript/index and a[10] is known as subscripted Value
You already know how to declare and define an array, 2D array etc.
What do you think are the problems with an ARRAY?

Problems with an ARRAY

Size can’t be changed - Once we declare the size of an array, we can’t resize it. If we want to extend the size we have to create another array with a larger size and then copy the previous one to it.
Insertion and deletion is difficult in the middle of the array - If we have to insert the data at the middle of the array then we have to shift either the right part or the left part of the array.
Large continuous memory is required to save the array.

So we arrived at the concept of the linked list

Can be one way or two way or m way linked list
linear collection of data element called as nodes
Maintains a pointer “start”
Declared in C as

```
struct node {
  int data;
  struct node *link;
};
```

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int data;
  struct node *next;
};

int main()
{
  struct node* head = NULL;
  struct node* second = NULL;
  struct node* third = NULL;
  struct node* temp;
  head  = (struct node*)malloc(sizeof(struct node));
  second = (struct node*)malloc(sizeof(struct node));
  third  = (struct node*)malloc(sizeof(struct node));
```

```
  head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    temp=head;
    while (temp != NULL)
    {
       printf(" %d ", temp->data);
       temp = temp->next;
    }

    return 0;
}
```

Operations of Linked List

Creation of a Linked List

[get one node from AVAIL and call it FIRST] // we will use malloc function
FIRST=(struct node *)malloc(sizeof(struct node));
Read the Data
[Put the Data]   FIRST->info=Data
[Put NULL in the link part]   FIRST->link=NULL
start=first

Insertion in a Linked list

At the beginning

[get one node from AVAIL and call it FIRST] // we will use malloc function
FIRST=(struct node *)malloc(sizeof(struct node));
Read the Data
[Put the Data]   FIRST->info=Data
[Put NULL in the link part]   FIRST->link=start
start=FIRST

Insertion in a Linked list

At the end

```
[get one node from AVAIL and call it NEW]  // we will use malloc function
  NEW=(struct node *)malloc(sizeof(struct node));
 Read the Data and set temp=start
 [Put the Data]    NEW->info=Data
 [Put NULL in the link part]   NEW->link=NULL
 while(temp->link!=NULL) {          // note the traversing here.
          temp=temp->link;  }
temp->link=NEW;
```

Insertion in a Linked list

after the nth node

```
[get one node from AVAIL and call it NEW]  // we will use malloc function
          NEW=(struct node *)malloc(sizeof(struct node));
         Read the Data and set temp=start, num=1;
         [Put the Data]    NEW->info=Data
         while(num!=n) {                          // note the traversing here.
                  temp=temp->link;
  num++;
         }
        NEW->link=temp->link;
        temp->link=NEW;
```

Combine all this-

```
#include<stdio.h>
#include<stdlib.h>
struct link_list
{
    int info;
    struct link_list *link;
}*start=NULL,*end=NULL,*NEW,*temp;
int enter_beg()
{
    NEW=(struct link_list *)malloc(sizeof(struct link_list));
    printf("Enter the info");
    scanf("%d",&(NEW->info));
    if(start==NULL)
    {
            start=NEW;
            end=NEW;
            NEW->link=NULL;
    }
 else {
            NEW->link=start;
            start=NEW; }
return 0;}
int enter_last()
{
    NEW=(struct link_list *)malloc(sizeof(struct link_list));
    printf("Enter the info");
    scanf("%d",&(NEW->info));
    if(start==NULL)
    {
            start=NEW;
            end=NEW;
            NEW->link=NULL;
    }
    else  {
            end->link=NEW;
            end=NEW;
            NEW->link=NULL;
    }
return 0;
}
```

```
int enter_mid()
{    int n,c=1;
    temp=start;
    NEW=(struct link_list *)malloc(sizeof(struct link_list));
    printf("Enter the node number ");
    scanf("%d",&n);
    while(c!=n)
    {  c++;
      temp=temp->link;
    }
    printf("Enter the info");
    scanf("%d",&(NEW->info));
    NEW->link=temp->link;
    temp->link=NEW;

return 0;
}
int display()
{         if(start==NULL)
    printf("No nodes to display");
    else
          {    temp=start;
           while(temp!=NULL)
          {
                printf("%d\t",temp->info);
                temp=temp->link;
          }
  }}
int main()
{    int n;
    while(1)
    {
          printf("\nWhat do you want to do\n1.enter at
beginning\n2.enter at mid\n3.enter at
last\n4.display\n5.exit\n");
          fflush(stdin);
          scanf("%d",&n);
          switch(n)  {
                case 1:enter_beg();
                      break;
                case 2:enter_mid();
                      break;
                case 3:enter_last();
                      break;
                case 4:display();
                      break;
                case 5: exit(0);     }
    }
return 0;   }
```

DELETE A NODE

```
int del()
{
        int n,flag;
        if(start==NULL)
        {
                printf("No nodes to delete\n");
                return 0;
        }
        printf("Enter the info you want to
delete");
        scanf("%d",&n);
        temp=start;
        if(temp->info==n)    //beginning
        {
                start=temp->link;
                return 0;
        }
        while(temp->link->link!=NULL)  // middle
                {
                        if(temp->link->info==n)
                        {
                                flag=1;
                                break;
                        }
                        temp=temp->link;
                }

          if(flag==1)
          {
                  temp->link=temp->link->link;
          }
          else if(temp->link->info==n)        //last
          {
                  temp->link=NULL;
                  end=temp;
          }
}
```

Applications of Linked List

Double Linked List-

1. Back button in the browser
2. Most recently used list
3. Hash table, priority queue and binary trees are implemented using Linked Lists
4. Undo button in word,photoshop.

Circular linked list is used in time sharing OS ie. For Multiple users and in some of the multiplayer board games.

Array VS Linked List

- Linked list provides following two advantages over arrays
- 1) Dynamic size
- 2) Ease of insertion/deletion
- Linked lists have following drawbacks:
- 1) Random access is not allowed. We have to access elements sequentially starting from the first node.
- 2) Extra memory space for a pointer is required with each element of the list.
- 3) Arrays have better cache locality(temporal and spatial) that can make a pretty big difference in performance.

Function to find the length of LL

```
int len(){
int c=0;
temp=start;
while(temp!=NULL)
{
        temp=temp->link;
        c++;
}
printf("No of nodes - %d\n",c );
Return 0;
 }
```

Function to find the location of info

```
int find(){
int c=1,n;
if(start==NULL)
{
        printf("Empty link_list\n");
        return 0;
}
temp=start;
printf("Enter the info");
scanf("%d",&n);
printf("Location of %d -\n",n);
```

```
while(temp!=NULL)
{
        if(temp->info==n)
        {
                printf("%d\t",c);
        }
        temp=temp->link;
        c++;
}
return 0;
}
```

PRACTICE

1.WAP to count the no. of nodes present in a linked list.

2.WAP to count the no. of occurrence of an info.

3.WAP to modify the info of a given node no.

4.WAP to swap two nodes of linked list

5.WAP to merge two sorted linked lists

END OF SINGLE LINKED LIST