1. ARRAY IMPLEMENTATION OF LIST ADT

Preamble

In computer science, an abstract data type (ADT) is a mathematical model for data types, defined by its behavior (semantics) from the point of view of a user of the data, specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations.

In List ADT, the data are generally stored in key sequence in a list. We can implement list using arrays or linked lists which has a head structure consisting of count, pointers and address of compare function needed to compare the data in the list.

Steps

Step 1: Define all global variables and functions.

Step 2: Get the option from user and execute switch statement.

Step 3: If it is insertion get the number from user and insert into the list.

- If the list is empty add that number as the first element.
- If the list id full then display it is full or else insert according to the position.

Step 4: If it is deletion, check list is empty or not. If it is empty print empty else search that element and remove the number from the list else display

Step 5: If operation is display then list the elements in the list.

Step 6: Stop the program

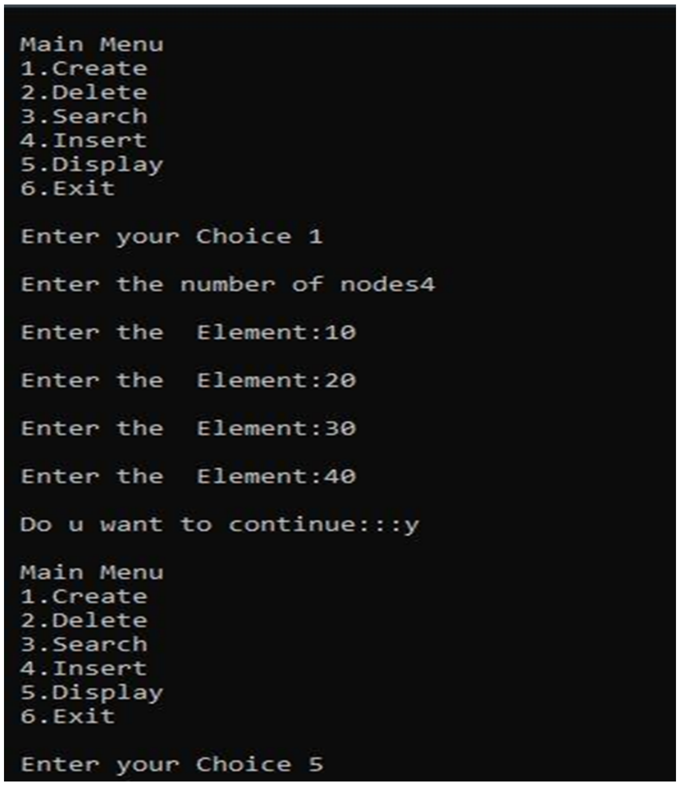
Implementation in C

```
#include<stdio.h>

#include<conio.h>

#define MAX 10

/* Define the required functions to create a list, insert into the list,
delete an element from the list, search and display the list */

void create();

void insert();

void deletion();

void search();

void display();

int a,b[20], n, p, e, f, i, pos;
```

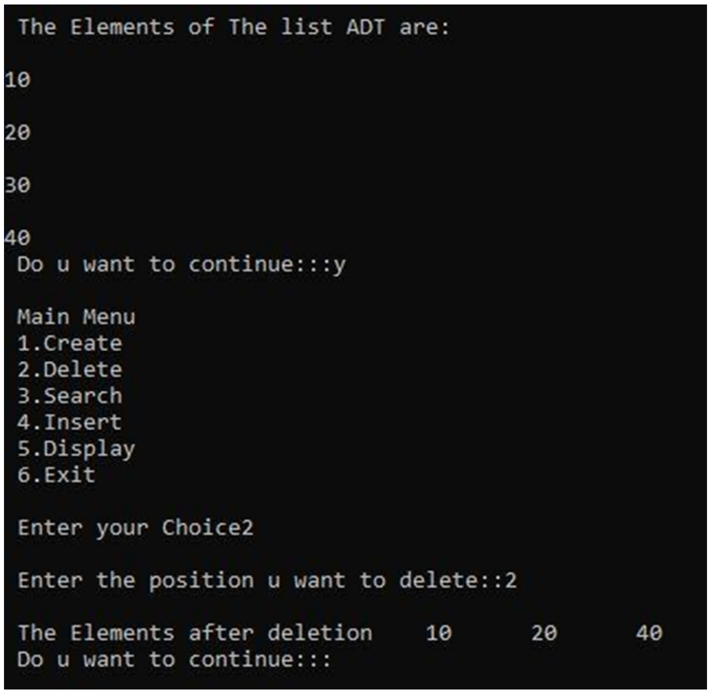
```
void main()

{

      //clrscr();

      int ch;

      char g='y';

      do

      {

            printf("\n main Menu");

            printf("\n 1.Create \n 2.Delete \n 3.Search \n 4.Insert \n
            5.Display\n 6.Exit \n");

            printf("\n Enter your Choice");

            scanf("%d", &ch);

            /* The following switch will call the appropriate choice provided
            by the user */

            switch(ch)

            {

                  case 1:

                        create();

                        break;

                  case 2:

                        deletion();

                        break;

                  case 3:

                        search();

                        break;

                  case 4:

                        insert();

                        break;

                  case 5:

                        display();

                        break;

                  case 6:

                        exit();
                        break;

                    default:

                        printf(" \n Enter the correct choice:");

                }

                printf("\n Do u want to continue:::");

                scanf("\n%c", &g);

        }

        /* The program is intended to run till we provide inputs other that ‘y’
        or ‘Y’ */

        while(g=='y'||g=='Y');

                getch();

}

void create()

{

        printf("\n Enter the number of nodes");

        scanf("%d", &n);

        /* The loop should run till we get the ‘n’ inputs */

        for(i=0;i<n;i++) {

        {

                printf("\n Enter the Element:",i+1);

                scanf("%d", &b[i]);

        }

}

void deletion()

{

        printf("\n Enter the position u want to delete::");

        scanf("%d", &pos);

        if(pos>=n)

        {

                printf("\n Invalid Location::");

        }

        else
        {
                for(i=pos+1;i<n;i++)
                {
                        b[i-1]=b[i];
                }n--;
        }
        printf("\n The Elements after deletion");
        for(i=0;i<n;i++)
        {
                printf("\t%d", b[i]);
        }
}
void search()
{
        printf("\n Enter the Element to be searched:");
        scanf("%d", &e);
        for(i=0;i<n;i++)
        {
                if(b[i]==e)
                {
                        printf("Value is in the %d Position", i);
                }
                else
                {
                        printf("Value %d is not in the list::", e);
                        continue;
                }
        }
}
void insert()
{
        printf("\n Enter the position u need to insert::");
```

```
	scanf("%d", &pos);
	if(pos>=n)
	{
		printf("\n invalid Location::");
	}
	else
	{
		for(i=MAX-1;i>=pos-1;i--)
		{
			b[i+1]=b[i];
		}
		printf("\n Enter the element to insert::\n");
		scanf("%d",&p);
		b[pos]=p;
		n++;
	}
	printf("\n The list after insertion::\n");
	display();
}
void display()
{
	printf("\n The Elements of The list ADT are:");
	for(i=0;i<n;i++)
	{
		printf("\n\n%d", b[i]);
	}
}
```

Sample Input and Output:



```
Main Menu
1.Create
2.Delete
3.Search
4.Insert
5.Display
6.Exit

Enter your Choice 1

Enter the number of nodes4

Enter the  Element:10

Enter the  Element:20

Enter the  Element:30

Enter the  Element:40

Do u want to continue:::y

Main Menu
1.Create
2.Delete
3.Search
4.Insert
5.Display
6.Exit

Enter your Choice 5
```



```
 The Elements of The list ADT are:

10

20

30

40
 Do u want to continue:::y

 Main Menu
 1.Create
 2.Delete
 3.Search
 4.Insert
 5.Display
 6.Exit

 Enter your Choice2

 Enter the position u want to delete::2

 The Elements after deletion    10      20      40
 Do u want to continue:::
```