#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*link;

};

struct node \*head;

void beginsert ();

void lastinsert ();

void randominsert();

void begin\_delete();

void last\_delete();

void random\_delete();

void display();

void main ()

{

int choice =0;

while(choice != 8)

{

printf("\nMENU\n");

printf("\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning\n5.Delete from last\n6.Delete node after specified location\n7.Show\n8.Exit\n");

printf("\nEnter your choice?\n");

scanf("\n%d",&choice);

switch(choice)

{

case 1: beginsert();

break;

case 2: lastinsert();

break;

case 3: randominsert();

break;

case 4: begin\_delete();

break;

case 5: last\_delete();

break;

case 6: random\_delete();

break;

case 7: display();

break;

case 8: exit(0);

break;

default:

printf("Please enter valid choice..");

}

}

}

void beginsert()

{

struct node \*ptr;

int item;

ptr = (struct node \*) malloc(sizeof(struct node \*));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value\n");

scanf("%d",&item);

ptr->data = item;

ptr->link = head;

head = ptr;

}

}

void lastinsert()

{

struct node \*ptr,\*temp;

int item;

ptr = (struct node\*)malloc(sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter value?\n");

scanf("%d",&item);

ptr->data = item;

if(head == NULL)

{

ptr -> link = NULL;

head = ptr;

}

else

{

temp = head;

while (temp -> link != NULL)

{

temp = temp -> link;

}

temp->link = ptr;

ptr->link = NULL;

}

}

}

void randominsert()

{

int i,loc,item;

struct node \*ptr, \*temp;

ptr = (struct node \*) malloc (sizeof(struct node));

if(ptr == NULL)

{

printf("\nOVERFLOW");

}

else

{

printf("\nEnter element value");

scanf("%d",&item);

ptr->data = item;

printf("\nEnter the location after which you want to insert ");

scanf("\n%d",&loc);

temp=head;

for(i=0;i<loc;i++)

{

temp = temp->link;

if(temp == NULL)

{

printf("\nInvalid entry\n");

return;

}

}

ptr ->link = temp ->link;

temp ->link = ptr;

}

}

void begin\_delete()

{

struct node \*ptr;

if(head == NULL)

{

printf("\nList is empty\n");

}

else

{

ptr = head;

head = ptr->link;

free(ptr);

}

}

void last\_delete()

{

struct node \*ptr,\*ptr1;

if(head == NULL)

{

printf("\nlist is empty");

}

else if(head -> link == NULL)

{

head = NULL;

free(head);

}

else

{

ptr = head;

while(ptr->link != NULL)

{

ptr1 = ptr;

ptr = ptr ->link;

}

ptr1->link = NULL;

free(ptr);

}

}

void random\_delete()

{

struct node \*ptr,\*ptr1;

int loc,i;

printf("\n Enter the location to delete \n");

scanf("%d",&loc);

ptr=head;

for(i=0;i<loc;i++)

{

ptr1 = ptr;

ptr = ptr->link;

if(ptr == NULL)

{

printf("\nCan't delete");

return;

}

}

ptr1 ->link = ptr ->link;

free(ptr);

printf("\nDeleted node %d ",loc+1);

}

void display()

{

struct node \*ptr;

ptr = head;

if(ptr == NULL)

{

printf("Empty");

}

else

{

printf("\nThe array is\n");

while (ptr!=NULL)

{

printf("\n%d",ptr->data);

ptr = ptr -> link;

}

}

}







