#include <stdio.h>

#include <stdlib.h>

void insrtbegin(void);

void insrt(void);

void insrtend(void);

void deletbegin(void);

void delet(void);

void deletend(void);

struct node

{

int data;

struct node\* next;

};

void findprev(int,struct node\*);

typedef struct node position;

typedef struct node\* temp;

typedef struct node\* list;

struct node \*head = NULL;

int main(void)

{

int i;

int option,a,b;

checkpoint:

printf("\nChoose the option\n1.Insert\n2.Delete\n3.Display\n4.Exit\n");

scanf("%d",&option);

//head = malloc(sizeof(struct node));

switch(option)

{

case 1:

{ insertopr:

printf("\nChoose the option\n1.Insert\_first\n2.Insert\_of\_choice\n3.Insert\_last\n4.go\_back\n");

scanf("%d",&i);

if(i == 1)

insrtbegin();

else if(i == 2)

insrt();

else if(i == 3)

insrtend();

else if(i == 4)

goto checkpoint;

printf("\npress 0 to go back");

scanf("%d",&i);

if(i == 0)

goto insertopr;

else

goto checkpoint;

}break;

case 2:

{deletopr:

printf("\nChoose the option\n1.Delete\_first\n2.Delete\_of\_choice\n3.Delete\_last\n4.go\_back\n");

scanf("%d",&i);

if(i == 1)

deletbegin();

else if(i == 2)

delet();

else if(i == 3)

deletend();

else if(i == 4)

goto checkpoint;

printf("\npress 0 to go back");

scanf("%d",&i);

if(i == 0)

goto deletopr;

else

goto checkpoint;

}break;

case 3:

{

position \*ptr;

if(head==NULL)

{

printf("\nNo elements in the list\n");

goto checkpoint;

}

else

{

ptr=head;

printf("\nThe contents of the List are :\n");

while(ptr!=NULL)

{

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

printf("->");

ptr=ptr->next ;

}

printf("NULL\n");

goto checkpoint;

}break;

}

default:

{

printf("Thank you for using my singly linked list application");

exit(0);

}

}

return 0;

}

void insrt()

{

int x,y;

position \*temp,\*ptr;

temp=malloc(sizeof(struct node));

printf("\nEnter the element to insert\n");

scanf("%d",&x);

temp -> data = x;

temp -> next = NULL;

if(head == NULL)

{

head = temp;

printf("element %d inserted successfully\n",x);

return;

}

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

scanf("%d",&y);

ptr = head;

if(ptr->data == y)

{

temp->next = ptr ->next;

ptr->next = temp;

}

else

{

while(ptr != NULL && ptr->data != y)

{

ptr = ptr->next;

}

temp->next = ptr ->next;

ptr->next = temp;

}

printf("element %d inserted successfully\n",x);

}

void insrtend()

{

int x;

position \*temp,\*ptr;

temp=malloc(sizeof(struct node));

printf("\nEnter the element to insert\n");

scanf("%d",&x);

temp -> data = x;

temp -> next = NULL;

if(head == NULL)

{

head = temp;

printf("element %d inserted successfully\n",x);

return;

}

ptr = head;

while(ptr != NULL && ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->next = temp;

printf("element %d inserted successfully\n",x);

}

void insrtbegin()

{

int x;

position \*temp,\*ptr;

temp=malloc(sizeof(struct node));

printf("\nEnter the element to insert\n");

scanf("%d",&x);

temp -> data = x;

temp -> next = NULL;

if(head == NULL)

{

head = temp;

printf("element %d inserted successfully\n",x);

return;

}

ptr = head;

head = temp;

head->next = ptr;

printf("element %d inserted successfully\n",x);

}

void delet()

{

int x;

printf("\nEnter the element you want to delete\n");

scanf("%d",&x);

position \*temp,\*ptr;

ptr = head;

if(ptr->data == x)

{

head = ptr->next;

free(ptr);

}

else

{

while( ptr != NULL && ptr->next->data != x)

{

temp = ptr;

ptr=ptr->next ;

}

temp -> next = ptr -> next;

free(ptr);

}

printf("\nelement %d deleted successfully",x);

}

void deletbegin()

{

position \*temp,\*ptr;

ptr = head;

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

head = ptr -> next;

free(ptr);

}

void deletend()

{

position \*temp,\*ptr;

ptr = head;

temp = ptr -> next;

if(temp == NULL)

{

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

free(temp);

free(ptr);

head = NULL;

return;

}

while(ptr != NULL && ptr->next->next != NULL)

ptr = ptr -> next;

temp = ptr -> next;

printf("\nelement %d deleted successfully\n",temp->data);

ptr -> next = NULL;

free(temp);

}