#include "../head/duLinkedList.h"

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

/\*\*

\* @name : Status InitList\_DuL(DuLinkedList \*L)

\* @description : initialize an empty linked list with only the head node

\* @param : L(the head node)

\* @return : Status

\* @notice : None

\*/

Status InitList\_DuL(DuLinkedList \*L) {

DuLinkedList p,p1;

p=\*L;

if(p==NULL){

p=(DuLinkedList)malloc(sizeof(DuLNode));

p->next=NULL;

p->prior=NULL;

(\*L)=p;

return SUCCESS;

}

if((\*L)->next!=NULL){

p=(\*L)->next;

}

while(p!=NULL){

p1=p;

p=p->next;

free(p1);

}

return SUCCESS;

}

void DoInitList\_DuL(DuLinkedList \*head){

if(InitList\_DuL(head)==SUCCESS){

printf("Successfully initialize an empty linked list\n");

}

}

/\*\*

\* @name : void DestroyList\_DuL(DuLinkedList \*L)

\* @description : destroy a linked list

\* @param : L(the head node)

\* @return : status

\* @notice : None

\*/

void DestroyList\_DuL(DuLinkedList \*L) {

DuLinkedList p;

p=\*L;

while(p!=NULL){

\*L=p->next;

free(p);

p=\*L;

}

}

void DoDestroyList\_DuL(DuLinkedList \*L){

DestroyList\_DuL(L);

printf("Finish destroy the linked list\n");

}

/\*\*

\* @name : Status InsertBeforeList\_DuL(DuLNode \*p, LNode \*q)

\* @description : insert node q before node p

\* @param : p, q

\* @return : status

\* @notice : None

\*/

Status InsertBeforeList\_DuL(DuLNode \*p, DuLNode \*q) {

if(p->prior==NULL){

p->next=q;

q->prior=p;

q->next=NULL;

return SUCCESS;

}

q->next=p;

q->prior=p->prior;

p->prior->next=q;

p->prior=q;

return SUCCESS;

}

void DoInsertBeforeList\_DuL(DuLNode \*p){

DuLNode \*q;

q=(DuLNode \*)malloc(sizeof(DuLNode));

printf("Please enter the data of insert:");

scanf("%d",&q->data);

if(p->next==NULL){

if(InsertBeforeList\_DuL(p,q)==SUCCESS){

printf("Successfully insert.\n");

return ;

}

}

else {

printf("Please enter the location's data':");

scanf("%d",&q->data);

p=p->next;

while(p!=NULL){

if(p->data==q->data){

break;

}

}

if(p==NULL){

printf("Can't find the location.\n");

return ;

}

if(InsertBeforeList\_DuL(p,q)==SUCCESS){

printf("Successfully insert.\n");

return ;

}

}

}

/\*\*

\* @name : Status InsertAfterList\_DuL(DuLNode \*p, DuLNode \*q)

\* @description : insert node q after node p

\* @param : p, q

\* @return : status

\* @notice : None

\*/

Status InsertAfterList\_DuL(DuLNode \*p, DuLNode \*q) {

q->next->prior=p;

p->next=q->next;

p->prior=q;

q->next=p;

}

void DoInsertAfterList\_DuL(DuLNode \*p){

DuLNode \*q;

q=(DuLNode \*)malloc(sizeof(DuLNode));

printf("Please enter the data of insert:");

scanf("%d",&q->data);

if(p->next==NULL){

if(InsertAfterList\_DuL(p,q)==SUCCESS){

printf("There is on data in the list\n");

return ;

}

}

else{

printf("Please enter the location's data':");

scanf("%d",&q->data);

p=p->next;

while(p!=NULL){

if(p->data==q->data){

break;

}

}

if(p==NULL){

printf("Can't find the location.\n");

return ;

}

if(InsertAfterList\_DuL(p,q)==SUCCESS){

printf("Successfully insert.\n");

return ;

}

}

}

/\*\*

\* @name : Status DeleteList\_DuL(DuLNode \*p, ElemType \*e)

\* @description : delete the first node after the node p and assign its value to e

\* @param : p, e

\* @return : status

\* @notice : None

\*/

Status DeleteList\_DuL(DuLNode \*p, ElemType \*e) {

if(p->next==NULL){

return ERROR;

}

p->next->data=\*e;

return SUCCESS;

}

/\*\*

\* @name : void TraverseList\_DuL(DuLinkedList L, void (\*visit)(ElemType e))

\* @description : traverse the linked list and call the funtion visit

\* @param : L(the head node), visit

\* @return : Status

\* @notice : None

\*/

void TraverseList\_DuL(DuLinkedList L, void (\*visit)(ElemType e)) {

}

void ShowMenu(void){

int i=70;

while(i--){

putchar('\*');

}

putchar(10);

printf("Please enter number to choose function:\n");

printf("0:initialize an empty linked list with only the head node.\n");

printf("1:Destroy the linked list.\n");

printf("2:Insert a node before other node.\n");

printf("3:Insert a node after other node.\n");

printf("5:Search List.\n");

printf("6:reverse the linked list.\n");

printf("-1:exit the pragram.\n");

i=70;

while(i--){

putchar('\*');

}

putchar(10);

}

int GetNumber(int \*n){

char a[4];

while(1){

scanf("%3s",a);

if(strcmp(a,"-1")==0){

printf("pragram has exited.\n");

exit (0);

}

if(strlen(a)==1&&'0'<=a[0]&&a[0]<='9'){

\*n=a[0]-'0';

putchar(10);

return \*n;

}

else{

while(getchar()!='\n');

printf("enter error,please enter again:\n");

return GetNumber(n);

}

}

}