201533661 이승수’s lab8

<lab 9-1>

#include <stdio.h>

struct STACK

{

int key;

struct STACK\* nextptr;

};

int is\_empty(struct STACK \*head)

{

if (head == NULL)

return 1;

else

return 0;

}

void push(struct STACK\*\* Preptr,int value)

{

if (is\_empty(\*Preptr))//head가 들어갔을 때

{

struct STACK \*Newptr = (struct STACK\*)malloc(sizeof(struct STACK \*Preptr = Newptr;

Newptr->key = value;

Newptr->nextptr = NULL;

}

else

{

struct STACK \*Newptr = (struct STACK\*)malloc(sizeof(struct STACK));

(\*Preptr)->nextptr = Newptr;//앞의 노드의 next=New 주소

Newptr->key = value;//새로 push해서 만든 ptr의 key에 value

Newptr->nextptr = NULL;//ptr nextptr NULL

}

}

int pop(struct STACK\*\* Curptr)//&head

{

while (((\*Curptr)->nextptr)->nextptr != NULL)

{

(\*Curptr)->nextptr = NULL;

free((\*Curptr)->nextptr);

}

}

void display(struct STACK \*head)

{

struct STACK \*Curptr=NULL;

Curptr = head;

if (is\_empty(head))

{

printf("\nStack is empty");

}

while (Curptr != NULL)

{

printf("%d ->", Curptr->key);

Curptr = (Curptr->nextptr);

}

printf("NULL");

}

int print()

{

int key;

printf("\nPut key\_value(1:push key,2:pop key,else:end stack)");

scanf\_s("%d",&key);

return key;

}

void main()

{

struct STACK \*head = NULL;

int key\_value,value;

while (1)

{

key\_value = print();

if (key\_value == 1)

{

printf("\nPut value:");

scanf\_s("%d",&value);

push(&head,value);

display(head);

}

else if (key\_value == 2)

{

pop(&head);

display(head);

}

else

{

display(head);

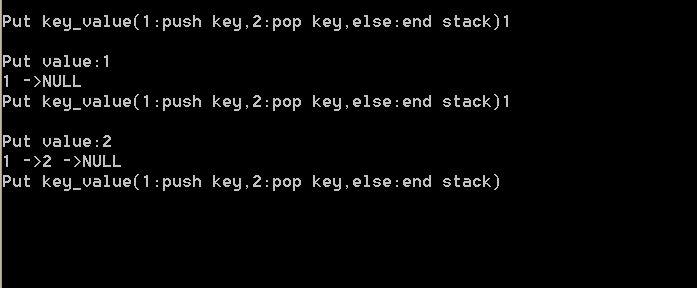
printf("\nBreak the stack");

break;

}

}

}



<lab 9-2>

#include <stdio.h>

struct Queue

{

int key;

struct Queue\* nextptr;

};

int is\_empty(struct Queue \*head)

{

if (head == NULL)

return 1;

else

return 0;

}

void push(struct Queue\*\* Preptr, int value)

{

if (is\_empty(\*Preptr))//head가 들어갔을 때

{

struct Queue \*Newptr = (struct Queue\*)malloc(sizeof(struct Queue \*Preptr = Newptr;

Newptr->key = value;

Newptr->nextptr = NULL;

}

else

{

struct Queue \*Newptr = (struct Queue\*)malloc(sizeof(struct Queue));

(\*Preptr)->nextptr = Newptr;//앞의 노드의 next=New 주소

Newptr->key = value;//새로 push해서 만든 ptr의 key에 value

Newptr->nextptr = NULL;//ptr nextptr NULL

}

}

int pop(struct Queue\*\* Curptr)//&head

{

struct Queue temp;

while ((\*Curptr)->nextptr!=NULL)

{

temp.nextptr = (\*Curptr)->nextptr;

free(\*Curptr);

\*Curptr = temp.nextptr;

}

}

void display(struct Queue \*head)

{

struct Queue \*Curptr = NULL;

Curptr = head;

if (is\_empty(head))

{

printf("\nStack is empty");

}

while (Curptr != NULL)

{

printf("%d ->", Curptr->key);

Curptr = (Curptr->nextptr);

}

printf("NULL");

}

int print()

{

int key;

printf("\nPut key\_value(1:push key,2:pop key,else:end stack)");

scanf\_s("%d",&key);

return key;

}

void main()

{

struct Queue \*head = NULL;

int key\_value,value;

while (1)

{

key\_value = print();

if (key\_value == 1)

{

printf("\nPut value:");

scanf\_s("%d",&value);

push(&head,value);

display(head);

}

else if (key\_value == 2)

{

pop(&head);

display(head);

}

else

{

display(head);

printf("\nBreak the stack");

break;

}

}

}

