201533661 이승수’s 9/24 소프트웨어 패턴 구현 과제

#include<stdio.h>

#include<stdlib.h>

struct listNode//구조체 listNode의 생성(내부 데이터로 data와 \*nextPtr을 가진다.)

{

char data;

struct listNode\* nextPtr;//구조체 안에 \*구조체를 넣어줌으로써 연쇄적으로 다른 구조체를 지정하도록 함.

};

typedef struct listNode ListNode;//struct listNode를 ListNode로 줄여준다.

typedef ListNode \*ListNodePtr;//ListNode를 \*ListNodePtr로 대신 부른다. 결과적으로 struct listNode ListNode를 \*ListNodePtr로 줄여준다.

void insert(ListNodePtr \*sPtr, char value);

char delete(ListNodePtr \*sPtr, char value);

int isEmpty(ListNodePtr sPtr);

void printList(ListNodePtr currentPtr);

void instructions(void);

int main(void)

{

ListNodePtr startPtr = NULL;//startPtr의 data에 NULL을 넣어준다.

int choice;

char item;

instructions();//출력함수

printf("? ");

scanf("%d", &choice);//첫번째로 숫자를 입력받는다.

while (choice != 3)//3이면 입력받던 것을 멈추고 프로그램을 멈춘다.그 외의 경우는 모두 반복.

{

switch (choice)//입력받은 숫자에 따라 link를 choice가 1이면 잇거나 choice가 2이면 지운다.

{

case 1://새로운 캐릭터()

printf("Enter a character: ");

scanf("\n%c", &item);//item을 입력받음.

insert(&startPtr, item);//한 블럭씩 list의 node들을 입력받는다.

printList(startPtr);//list를 출력

break;

case 2://입력받은 캐릭터를 지운다.

if (!isEmpty(startPtr))//sPtr!=NULL일 때

{

printf("Enter character to be deleted: ");

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

if (delete(&startPtr, item))//입력받은 item이 NULL이 아닐때 실행

{

printf("%c deleted \n", item);

printList(startPtr);

}

else//리턴받은 item이 NULL일 때 실행

{

printf("%c not found.\n\n", item);

}

}

else//sPtr이 NULL일 때

{

printf("List is empty.\n\n");

}

break;

default://나머지경우:1,2가 아닌 그 이외의 숫자

printf("Invalid choice.\n\n");

instructions();

break;

}

printf("? ");

scanf("%d", &choice);

}//그 뒤로 반복함.

printf("End of run.\n");

return 0;

}//end main

void instructions(void)//사용자의 choice를 1,2,3 중에서 하나 입력받도록 출력해주는 함수

{

printf("Enter your choice:\n"

" 1 to insert an element into the list.\n"

" 2 to delete an element from the list.\n"

" 3 to end.\n");

}//end instructions

void insert(ListNodePtr\* sPtr, char value)//list들의 순서를 오름차순으로 배열해주고, 사이에 newPtr을 껴준다.

{

ListNodePtr newPtr;

ListNodePtr previousPtr;

ListNodePtr currentPtr;//listNode의 포인터로서 newPtr,previousPtr,currentPtr의 선언.

newPtr = malloc(sizeof(ListNode));//newPtr에 한 리스트 만큼의 메모리를 할당해줌.

if (newPtr != NULL)//메모리 할당이 된 경우

{

newPtr->data = value;//newPtr의 data에는 value를 입력해주고

newPtr->nextPtr = NULL;//nextPtr은 NULL로 초기화해준다.

previousPtr = NULL;//previousPtr을 NULL로 초기화 해주고

currentPtr = \*sPtr;//currentPtr에 sPtr의 값을 넣어준다.다시말해 currentPtr에 \*sPtr을 복사해준다.

while (currentPtr != NULL && value > currentPtr->data)//linked list가 오름차순에 따르지 않을 경우 재배열 해준다.

{

previousPtr = currentPtr;//previous구조체에 current구조체를 입력

currentPtr = currentPtr->nextPtr;//current구조체에 next의 주소의 구조체를 입력

}

if (previousPtr == NULL)//NewPtr 다음에 sPtr이 온다.

{

newPtr->nextPtr = \*sPtr;//NewPtr의 nextPtr을 NULL에서 \*sPtr의 주소로 바꿔준다.(큰 의미는 알겠지만 여기서 \*sPtr 대신에 &\*sPtr이 들어가야 주소를 전해준다는 의미상 맞는 것 아닌가요?)

\*sPtr = newPtr;//sPtr에는 newPtr의 값들이 들어간다.

}

else//previousPtr과 CurrentPtr 사이에 NewPtr을 끼워넣는다.

{

previousPtr->nextPtr = newPtr;//

newPtr->nextPtr = currentPtr;

}

}

else//NewPtr==NULL인 경우(메모리 할당이 안된 경우)

{

printf("%c not inserted. No memory available.\n", value);

}

}//end insert

char delete(ListNodePtr\* sPtr, char value)

{

ListNodePtr previousPtr;

ListNodePtr currentPtr;

ListNodePtr tempPtr;

if (value == (\*sPtr)->data)//sPtr의 data가 item과 같을 때는 해당 node를 delete해줌

{

tempPtr = \*sPtr;//sPtr의 내용을 tempPtr에 복사(=call by reference방식과 비슷)

\*sPtr = (\*sPtr)->nextPtr;//sPtr에 그 다음에 링크된 구조체의 값들을 넣어줌(한칸씩 땡겨줌)

free(tempPtr);//tempPtr을 free해줌으로써 sPtr의 내용을 간접적으로 delete시켜줌

return value;//item값을 리턴

}

else//

{

previousPtr = \*sPtr;

currentPtr = (\*sPtr)->nextPtr;//이전 것과 현재 것을 뒤로 같이 밀면서 검사하게 함.

while (currentPtr != NULL&&currentPtr->data != value)//sPtr의 데이터와 item의 값이 같지 않으면 currentPtr의 자리를 빼준다.

{

previousPtr = currentPtr;

currentPtr = currentPtr->nextPtr;

}

if (currentPtr != NULL)//

{

tempPtr = currentPtr;//tempPtr에 currentPtr을 복사

previousPtr->nextPtr = currentPtr->nextPtr;//현재next주소를 앞의 next 주소로 옮김

free(tempPtr);//tempPtr(현재 node)를 delete함

return value;//item값을 리턴

}

}

return '\0';//NULL을 리턴

}//end delete

int isEmpty(ListNodePtr sPtr)//sPtr을 NULL로 바꿔서 리턴해줌

{

return sPtr == NULL;//sPtr==NULL을 리턴해줌

}

void printList(ListNodePtr currentPtr)//list의 내용을 출력해줌

{

if (currentPtr == NULL)//startPtr이 NULL일 떄는 초기화 외에는 할당받지 못한 것이므로 list가 비었다.

{

printf("List is empty.\n\n");

}

else//리스트 출력

{

printf("The list is:\n");

while (currentPtr != NULL)

{

printf("%c --> ", currentPtr->data);

currentPtr = currentPtr->nextPtr;//포인터의 주소를 계속 업데이트하면서 while문 내에서 출력

}

printf("NULL\n\n");//구조체는 어레이의 형태이므로 끝에 마지막이라는 표시로 NULL을 넣어줌

}

}//printList end

