/\*

이 프로그램은 단순 및 이중 연결리스트의 생성, 삽입, 삭제 뿐만 아니라

연결리스트의 기본 연산을 통하여 스택, 큐, 트리를 연산하는 함수를 포함시키고 있음.

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <time.h>

#define entry -1 // 리스트의 첫번째 위치를 가리키기 위한 상수

typedef struct singly {

int data;

struct singly \*next;

}int\_singleNode; // 단순 연결리스트를 위한 자기참조 구조체 : 데이터는 정수임

typedef struct doubly {

void \*data;

struct doubly \*llink;

struct doubly \*rlink;

}int\_doubleNode; // 이중 연결리스트를 위한 자기참조 구조체 : 데이터는 정수임

void menu(void); // 사용자 선택을 보여주는 함수

int viewList(int\_singleNode \*); // 리스트 내용을 확인하는 함수

int create\_Integer\_Input(char \*, FILE \*\*); // 정수 입력 데이터 파일 생성 함수

int writeFile(char \*, int\_singleNode \*); // 리스트에 있는 데이터를 파일에 작성하는 함수

int create\_Singly\_Linked\_List\_Int(int\_singleNode \*\*, char \*); // 정수를 연결리스트로 만들어 주는 함수

int findLastNode(int\_singleNode \*, int\_singleNode \*\*, int\_singleNode \*\*); // 리스트의 마지막 노드를 가리키는 포이터를 리턴해 주는 함수

int push(int\_singleNode \*\*, char \*); // 리스트 마지막에 데이터 입력하여 데이터 파일을 새롭게 해주는 함수

int\_singleNode \* pop(int\_singleNode \*\*, char \*); // 리스트 마지막에 데이터를 삭제하고 리턴해 주는 함수

int enqueue(int\_singleNode \*\*, char \*); // 리스트의 마지막에 데이터를 입력하는 함수로 기능은 push와 같으나 큐에서 사용되어 enqueue로 명명함.

int\_singleNode \* dequeue(int\_singleNode \*\*, char \*);// 리스트의 처음 데이터를 삭제하고 그 포인터를 리턴해 주는 함수

int insert(int\_singleNode \*\*, char \*); // 리스트에 데이터를 삽입하는 함수로 리스트 중간에 삽입할 때 사용.

int findElement(int\_singleNode \*, int\_singleNode \*\*, int\_singleNode \*\*, int); // 리스트에서 삽입할 위치를 찾는 함수

int deleteElement(int\_singleNode \*\*, char \*); // 리스트에서 원하는 데이터를 삭제하는 함수

// int create\_double\_Linked\_List();

// int view\_Data();

// void main\_Menu();

int main(void) {

FILE \* intFilePtr; // 데이터 파일 포이터

int\_singleNode \*singlePtr = NULL; // 파일에서 데이터를 읽어 들여 만든 단순 연결리스트 헤더 포인터

int\_singleNode \*testPtr = NULL; // 디버그를 위해 만든 단순 연결리스트 포인터

int\_singleNode \*getData = NULL; // 리턴되는 포인터의 값

char intFilename[20]; // 데이터 파일 이름.

create\_Integer\_Input(intFilename, &intFilePtr); // 정수를 랜덤으로 만들어 데이터 파일 생성

printf("파일이 생성되었습니다\n\n");

create\_Singly\_Linked\_List\_Int(&singlePtr, intFilename); // 생성된 데이터 파일의 데이터를 이용하여 단순연결리스트 생성

push(&singlePtr, intFilename); // 데이터를 리스트 마지막에 입력

getData = pop(&singlePtr, intFilename); // 리스트 마지막 노드를 삭제하고 그 포인터를 리턴받는 함수

if (getData != NULL)

printf("\npop data is %d\n", getData->data);

else

printf("스택이 비었습니다!\n");

enqueue(&singlePtr, intFilename); // 큐에 데이터를 삽입하는 함수

getData = dequeue(&singlePtr, intFilename); // 큐에서 데이터를 삭제하고 그 값을 리턴하는 함수

if (getData != NULL)

printf("Dequeue data is %d\n", getData->data);

else

printf("큐가 비었습니다!\n");

insert(&singlePtr, intFilename); // 단순연결리스트에 데이터를 삽입하는 함수

deleteElement(&singlePtr, intFilename);

viewList(singlePtr);

}

int create\_Integer\_Input (char \*fileName, FILE \*\* intPtr) {

// 랜덤으로 데이터를 만들어 데이터 파일을 생성하는 함수

// fileName은 데이터 파일 이름으로 여러 함수에서 사용하여 메인에서 선언한뒤 parameter로 전달

// intPtr은 메인에서 선언한 파일 포인터

int i, n;

printf("정수 데이터를 저장할 파일 이름을 입력하시오 ->\n");

scanf("%s", fileName);

if ((\*intPtr = fopen(fileName, "w")) == NULL) {

printf("Cannot open file.\n");

exit(1);

}

printf("몇개의 숫자를 만들고 싶습니까?\n\n");

printf("데이터 입력을 원하지 않을 경우 데이터의 개수를 0으로 입력\n");

printf("->\n");

scanf("%d", &n);

srand(time(NULL));

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

/\* 이부분에 파일에 랜덤 데이터를 저장하는 명령어를 넣어야 함.

fclose(\*intPtr);

return 0;

}

int create\_Singly\_Linked\_List\_Int(int\_singleNode \*\*singlePtr, char \*FileName) {

// 생성된 랜덤 데이터 파일을 읽어 리스트로 연결시키는 함수

// singlePtr은 리스트의 헤더 포인터의 포인터값

// FileName은 데이터 파일 이름

FILE \*inputPtr;

int\_singleNode \*newNode; // 리스트에 연결될 새로운 노드의 포인터

int\_singleNode \*tempPtr; // 리스트 헤더 포인터 값이 수정되는 것을 방지하기 위해 값을 복사하여 대신 사용되는 포인터 변수

int data; // 파일에 있는 데이터를 읽을때 사용되는 변수

tempPtr = \*singlePtr;

if ((inputPtr = fopen(FileName, "r")) == NULL) {

printf("Cannot open file.\n");

exit(1);

}

// EOF에 도달 할때 까지 반복

while (파일의 끝에 도달하지 않았을 경우) {

새로운 노드 생성

새로운 노드에 데이터 삽입

새로운 노드의 링크 부분을 널로 만듬

if (리스트가 비었을 때, 즉 널일때) { // 리스트가 비었을때 헤더에 새로운 노드를 연결함

헤더에 새로운 노드를 연결시키고

마지막 노드에 tempPtr이 가리키도록 함

}

else { // 리스트에 노드가 존재할때 마지막에 데이터 연결

마지막 노드 다음에 새 노드를 연결

tempPtr이 마지막 노드 즉 새노드를 가리키도록 함

}

}

fclose(inputPtr);

return 0;

}

int insert(int\_singleNode \*\*singleListPtr, char \*inputFileName) {

FILE \*inputFilePtr = NULL;

int\_singleNode \*tempPtr = NULL;

int\_singleNode \*prePtr = NULL;

int\_singleNode \*targetPtr = NULL;

int\_singleNode \*newNode = NULL;

int target;

int data;

tempPtr = \*singleListPtr;

if (tempPtr == NULL) {

printf("리스트가 비었습니다!\n");

} // end of if

else {

printf("리스트 : ");

while (tempPtr != NULL) {

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

tempPtr = tempPtr->next;

}

} //end of else

printf("\n어디 다음에 무엇을 삽입할지 입력하시오\n");

printf("처음에 삽입을 원하시면 -1을 입력하시고 데이터를 입력하시오!\n");

scanf("%d%d", &target, &data);

새 노드 생성

새 노드에 데이터 입력

새 노드 링크를 널로 만듬

tempPtr = \*singleListPtr;

if (target == entry)

if (target == NULL)

\*singleListPtr = newNode;

else {

newNode->next = \*singleListPtr;

\*singleListPtr = newNode;

} // end of else

else {

어디에 삽입을 원하는지 리스트의 원소 위치 확인

새로운 노드가 위치가 확인된 노드(타겟노드)가 가르키는 곳을 가리키도록 함

타겟노드의 링크가 새 노드를 가리키게 함

} // end of else

/\* writeFile(inputFileName, \*singleListPtr); \*/ // 새로운 리스트를 파일에 작성하는 함수 호출

}