원형 리스트(해적선 게임)

김 학 동

대덕소프트웨어마이스터고등학교

원형 리스트

■ 리스트의 구성

- 리스트에 저장할 자료형 정의

- 리스트 구조의 정의

//리스트 테일

//이전 참조 위치

//현재 참조하고 있는 위치

//리스트 내 자료의 개수

//자료 정렬을 위한 대소 비교 함수

- 리스트 자료형의 정의

원형 리스트

■ 리스트 운영 함수

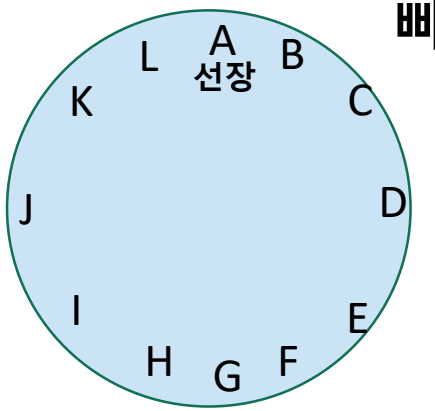
- 리스트의 초기화 함수: 해당 리스트의 초기화. 반환값 없음
- 자료 삽입: 리스트에 주어진 자료 삽입. 반환값 없음
- 첫 자료 검색: 0/1 반환, 검색 값은 리스트 자료의 포인터 변수
- 다음 자료 검색: 0/1 반환, 검색 값은 리스트 자료의 포인터 변수
- 현위치의 자료 삭제: 검색하여 찾은 현위치 자료를 삭제하고 삭제된 자료 반환
- 리스트 자료의 출력: 리스트 내 모든 자료의 출력. 반환값 없음
- 리스트 자료의 개수: 정수값 자료 개수 반환
- 리스트 자료의 정렬 기준 설정: 정수값 0/1 반환

원형 리스트

■ 원형 리스트를 이용한 문제

(1) 해적선에 선원들이 원탁 둘레에 앉아있다.

(2) 선장으로부터 시작하여 매 3번째 마다 물에 빠뜨리는 게임을 진행한다.



(3) 마지막으로 남은 선원은 물에 빠지지 않는다.

(4) 마지막 까지 남아 있으려면 맨 처음에 어떤 자리에 앉아야 할까?

(5) 인원수와 간격을 바꿀 때마다 물에 빠지지 않을 자리를 찾으려면?

(6) 이 경우에는 C F I L D H A G B K E 의 순서로 물에 빠지고 마지막으로 선원 J가 남는다.

■ 리스트의 구성

```
typedef      struct    _node {
      char       Item ;
      struct _node       *Next ;
} Node;

typedef      Node      *Ldata ;

typedef struct _list {
      Ldata      Tail ;
      Ldata      Before ;
      Ldata      Cur ;
      int        NumOfData ;
      int        (*func)(Ldata, Ldata) ;
} List ;
```

원형 리스트

■ 리스트 운영 함수

```
void        InitList(List *) ;          //리스트의 초기화
void        LInsert(List *, Ldata) ;//자료 삽입
int         LFirst(List *, Ldata) ;   //첫 자료가 있는가?
int         LNext(List *, Ldata) ;  //다음 자료가 있는가?
Ldata       LRemove(List *) ;       //자료 삭제
void        LPrint(List *) ;          //리스트 자료의 출력
int         LCount(List *) ;          //리스트 자료의 개수
void        SetSortRule(List *, int (*comp)(Ldata, Ldata)) ;      //정렬 규칙 설정
```

원형 리스트

■ CircularList의 헤더 파일: Circular.h, List.h

```
#ifndef     _CIRCULAR_H_
#define     _CIRCULAR_H_
typedef     struct   _node {
    char    Item ;
    struct _node    *Next ;
} Node;
typedef     Node    *Ldata ;

typedef struct _list {
    Ldata   Tail ;
    Ldata   Before ;
    Ldata   Cur ;
    int     NumOfData ;
    int     (*func)(Ldata, Ldata) ;
} List ;
#endif
```

```
#ifndef     _LIST_H_
#define     _LIST_H_

void        InitList(List *) ;
void        LInsert(List *, Ldata) ;
int         LFirst(List *, Ldata) ;
int         LNext(List *, Ldata) ;
Ldata       LRemove(List *) ;
void        LPrint(List *) ;
int         LCount(List *) ;
void  SetSortRule(List *, int (*comp)(Ldata, Ldata)) ;

#endif
```

원형 리스트

■ CircularList 의 함수 파일: CircularList.cpp

```
#include        <stdio.h>
#include        <stdlib.h>
#include        "Circular.h"
#include        "List.h“
```

```
void    InitList(List *plist) {                 //리스트의 초기화
        plist->Tail= NULL ;
        plist->Before= NULL ;
        plist->Cur= NULL ;
        plist->NumOfData= 0 ;
        plist->func= NULL;
}
```

```
void    PrintNode(Ldata pnode) {
        printf("%c ", pnode->Item);
}
```

원형 리스트

■ CircularList 의 함수 파일: CircularList.cpp

```
void    LPrint(List *plist) {
        int     last= LCount(plist), b ;
        if(last){  Ldata    pdata= plist->Tail->Next ;
                   for(b=0; b<last; b++, pdata= pdata->Next){
                            PrintNode(pdata) ;
                   } printf("\n") ;
        }
}
```

```
void    SInsert(List *plist, Ldata pdata) {
        Ldata   Before= plist->Tail ;
        if(plist->func(Before, pdata)){
                   while(plist->func(pdata, Before->Next))
                            Before= Before->Next ;
                   pdata->Next= Before->Next ;       Before->Next= pdata ;
        } else {
                   pdata->Next= Before->Next ;       Before->Next= pdata ;     plist->Tail= pdata ;
        }
}
```

원형 리스트

■ CircularList 의 함수 파일: CircularList.cpp

```
void    LInsert(List *plist, Ldata pdata) {    //자료 삽입
        if(plist->Tail){
                if(plist->func)
                        SInsert(plist, pdata) ;
                else {
                        pdata->Next= plist->Tail->Next ;
                        plist->Tail->Next= pdata ;
                        plist->Tail= pdata ;
                }
        } else {
                plist->Tail= pdata ;
                pdata->Next= pdata ;
        }
        plist->NumOfData++ ;
}
```

원형 리스트

■ CircularList 의 함수 파일: CircularList.cpp

```
int     LFirst(List *plist, Ldata pdata) {      //첫 자료가 있는가?
        if(LCount(plist)) {
                plist->Before= plist->Tail ;
                plist->Cur= plist->Tail->Next ;
                pdata->Item= plist->Cur->Item ;
                return 1 ;              //첫 자료 있음
        }
        return 0 ;                      //저장된 자료가 없음
}
```

```
int     LNext(List *plist, Ldata pdata) {      //다음 자료가 있는가?
        if(plist->Cur->Next) {
                plist->Before= plist->Cur ;
                plist->Cur= plist->Cur->Next ;
                pdata->Item= plist->Cur->Item ;
                return 1 ;              //다음 자료 있음
        }
        return 0 ;                      //다음 자료 없음
}
```

원형 리스트

■ CircularList 의 함수 파일: CircularList.cpp

```
Ldata LRemove(List *plist) {                        //자료 삭제
        if(LCount(plist)) {
                Ldata   rpos= plist->Cur ;          //삭제할 위치
                if(rpos==plist->Tail) plist->Tail= plist->Before ;

                plist->Before->Next= rpos->Next ;
                plist->Cur= plist->Before ;         //삭제한 후의 CurrentPosition
                plist->NumOfData-- ;                //자료 개수 감소
                return  rpos ;                      //삭제된 자료 반환
        }
}
```

```
void SetSortRule(List *plist, int (*comp)(Ldata, Ldata)){
        plist->func= comp ;
}
```

```
int     LCount(List *plist) {
        return plist->NumOfData ;
}
```

원형 리스트

■ CircularList 의 main 파일: CircularListmain.cpp

```
#include        <stdio.h>
#include        <conio.h>          //getch() 함수를 사용하기 위하여
#include        "Circular.h"
#include        "List.h"
#include        "CircularList.cpp"
#define         SSR(small,large)   ((small)<(large))
```

```
int WhoIsPred(Ldata d1, Ldata d2){
        if(SSR(d1->Item, d2->Item)) return 0;
        else return 1;
}
```

```
Ldata NewNode(char Item) {
        Ldata     NN= (Node *)malloc(sizeof(Node)) ;
        NN->Item= Item ;
        NN->Next= NULL ;
        return NN;
}
```

원형 리스트

■ CircularList 의 main 파일: CircularListmain.cpp

```
char    Delete(List *plist, int N){          //해적 원형 리스트 plist에 대하여 N의 간격으로 제거
        int     a=0 ;
        Ldata   Cur= plist->Tail, Before ;

        if(plist->NumOfData == 1)   return   plist->Tail->Item ;
        else {
                N의 간격으로 차례대로 선원을 제거할 것입니다.

                이곳에 여러분의 코드를 넣어 완성하시오.




                Delete(plist, N)
        }
}
```

원형 리스트

■ CircularList 의 main 파일: CircularListmain.cpp

```
int main(void) {
        List*     MyList= (List *)malloc(sizeof(List)) ;
        int       a=0;
        char      A[33]= "abcdefABCDEFGHIJKLMNOPQRSTUVWXYZ";

        InitList(MyList) ;   int MAX, NN;              //리스트의 초기화
        printf("인원수(<= 33), 간격 ");      scanf("%d %d", &MAX, &NN);

        printf("\n새로운 노드를 정렬하여 리스트에 추가 \n") ;
        SetSortRule(MyList, WhoIsPred) ;
        while(A[a] && a<MAX)                           //리스트에 자료 추가
                LInsert(MyList, NewNode(A[a++])) ;
        LPrint(MyList) ;

        printf("\n끝까지 남은 멤버는 누구일까? ");          getch() ;
        printf("\n%d번째 자료를 순환하며 제거 \n", NN) ;
        printf("\n남아 있는 멤버:  %c \n", Delete(MyList, NN)) ;
}
```

원형 리스트

■ CircularListmain.cpp의 실행 결과

```
인원수(<= 33), 간격 10 3

새로운 노드를 정렬하여 리스트에 추가
A B C D a b c d e f

끝까지 남은 멤버는 누구일까?
```

```
3번째 자료를 순환하며 제거
C] A B D a b c d e f
b] D a c d e f A B
e] c d f A B D a
B] f A D a c d
c] D a d f A
A] d f D a
d] D a f
a] f D
f] D

남아 있는 멤버: D
```