
C 프로그래밍 및 실습

실습 프로젝트

세종대학교

목차

- 1) 개요
- 2) 설계 과정
- 3) 단계별 사양
- 4) 일정

1) 프로젝트 개요

- 1) 개발 프로그램 : <<**연락처 관리 프로그램**>>
- 2) 사용자 인터페이스 : 콘솔모드에서 메뉴방식 으로 구성
 - 선택된 메뉴에 따라 적절한 작업실행 후 다시 메뉴 선택과정 반복
- 3) 저장할 정보
 - **이름**
 - **전화번호**
 - **생일**
- 4) 기본기능
 - **등록** : 입력 및 저장
 - 연락처 입력마다 이름 순으로 정렬되어 저장되도록 한다.
 - **삭제** : 연락처 삭제
 - 이름 입력 후 검색하여 해당 연락처 삭제
 - 전체 **자료보기** : 출력
 - **임의의 월**이 생일인 사람 검색

2) 프로젝트 설계과정 : 기본 구조

- main함수에서는 각 기능을 처리하는 함수들을 호출하는 역할을 하도록 하고, 기본적인 기능들은 함수에서 처리되어 유기적으로 연계되어 실행되도록 한다.

```
// 기본 메인 함수 구조
main()
{
    while (1)
    {
        // Menu Display
        // Select Menu Num#
        // Select Exit#
        // Call the Menu Function
    }
}
```

2) 프로젝트 설계 과정 : 구성 요소 분석 1

(1) 자료 구조

저장 정보

- **이름, 전화번호, 생일** → 필요한 자료구조는 '**구조체**'
- 이름 : 20 bytes, 전화번호 : 15 bytes , 생일 : 8 bytes
(이름은 영문대소문자,
전화번호는 '-'없이 번호만, 생일은 YYYYMMDD형식)
- 입력 값들 내부에 공백은 없는 것으로 간주

2) 프로젝트 설계 과정 : 구성 요소 분석 2

(2) 기능 세분화

- **등록** : 자료 입력 및 저장
 - : 연락처 입력 **마다 이름순으로 정렬된 순서로 저장**되도록 자료의 위치를 이동
 - : 연락처 입력 후 전체 자료 출력 시 항상 이름순으로 출력
 - : **동명이인은 없다고 가정**
- **삭제** : 이름으로 검색 후 해당 연락처 삭제
- **보기** : 표준 출력함수를 이용하여 화면에 차례로 출력
 - : 등록 시 정렬된 순서로 저장하므로 항상 이름순으로 정렬된 자료로 출력
- **생일인 사람 검색** : 월을 입력 받아, 그 달에 생일인 사람들의 정보를 출력

참고1: 이름으로 정렬 – 아스키 코드상의 순서를 말합니다. (strcmp사용)

2) 프로젝트 설계 과정 : 합성

(1) 필요 자료 구조 (변수사용)

- : 다수의 자료를 저장하는 구조체 배열 변수
- : 현재 저장된 자료의 개수를 나타내는 정수 변수
- : 최대 저장 가능한 자료의 숫자를 나타내는 정수 (상수 또는 변수)
MAX_NUM

(2) 필요 함수

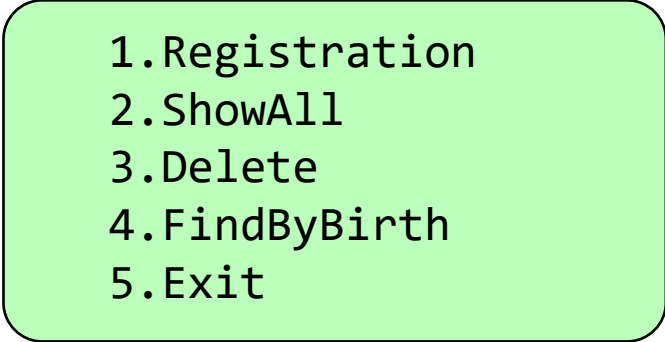
- : 메뉴 관련 함수 - 메뉴 보여주기,
- : 자료 처리 관련 함수 - 등록, 삭제, 보기, 월(생일) 검색

(3) 실행 화면

2) 프로젝트 설계 과정 : 합성

OJ 시스템에서 채점하므로 모든 입출력은 간단하고 정확하게 진행되어야 한다.

(1) 시작화면



- 1.Registration
- 2.ShowAll
- 3.Delete
- 4.FindByBirth
- 5.Exit

참고1 : OJ의 채점을 위하여 프로그램 명세서에 있는 출력코드를 그대로 사용

2) 프로젝트 설계 과정 : 합성

OJ 시스템에서 채점하므로 모든 입출력은 간단하고 정확하게 진행되어야 한다.

(1) 등록 화면 :

- 시작화면에서 '1' 선택 시
- 이름, 전화번호, 생일 순으로 입력
- **이름순으로 정렬된 순서로 저장**
- 처리 후 다시 시작화면으로 돌아감

```
1.Registration
2.ShowAll
3.Delete
4.FindByBirth
5.Exit
```

참고1 : 동명이인은 없다고 가정

참고2 : 이름 순으로 저장함

참고3 : 입력정보의 검증은 하지 않음

참고4 : 입력정보에는 빈칸을 허용하지 않음

참고5 : 이름 정렬 - 아스키 코드상의 순서 (`strcmp` 사용)

예시 - Registration

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number: 1

Name: KimEunJoo

Phone_number: 0001112222

Birth: 19960101

<<1>>

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number: 1

Name: LeeEunJoo

Phone_number: 0103332222

Birth: 19960202

<<2>>

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number: 1

Name: HanEunJoo

Phone_number: 0114445555

Birth: 20000101

2) 프로젝트 설계 과정 : 합성

- (1) 등록 화면 예외 처리 : 시작화면에서 '1' 선택 시
: 최대 수용 가능 한 연락처 수에 도달했으면
오류 메시지("OVERFLOW")를 출력 후 시작화면으로
복귀

2) 프로젝트 설계 과정 : 합성

- (2) **보기 화면** : 시작화면에서 '2' 입력 시 아래와 같이 **출력 후 시작화면**으로 복귀 (OJ시스템에서 채점하므로 별도 장식출력 없고 정보와 정보 사이에는 한 칸 빈칸을 둔다.)

```
HongGilDong 01011111111 20000301
SungChunHyang 0111112222 19960101
1.Registration
2.ShowAll
3.Delete
4.FindByBirth
5.Exit
```

예시 - Show All

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:2

HanEunJoo 0114445555 20000101

KimEunJoo 0001112222 19960101

LeeEunJoo 0103332222 19960202



알파벳 Ascending 순서로 출력

2) 프로젝트 설계 과정 : 합성

- (3) 삭제 화면 : 시작화면에서 '3' 입력 시
이름을 입력하면 삭제 후 다시 시작 화면으로 복귀

```
Name:HongGilDong
1.Registration
2.ShowAll
3.Delete
4.FindByBirth
5.Exit
```

참고1: 저장된 정보가 없는데 삭제 메뉴를 선택 시 오류 메시지 ("NO MEMBER") 출력 후 메뉴복귀

- 없는 이름을 삭제하는 경우는, 그냥 무시하고 메뉴로 복귀

참고: 빨간 색 부분이 입력

예시 - Delete

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:3

Name:KimEunJoo

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:2

HanEunJoo 0114445555 20000101

LeeEunJoo 0103332222 19960202

2) 프로젝트 설계 과정 : 합성

- (4) 생일자 검색 화면 : 시작화면에서 '4' 입력 시
달을 입력하면 해당하는 사람 정보 출력 후 시작 화면으로 복귀

```
Birth:3
HongGilDong 01011111111 20000301
1.Registration
2.ShowAll
3.Delete
4.FindByBirth
5.Exit
```

참고1: 해당 정보가 없으면 곧장 메뉴 복귀

참고2: 같은 달이 생일인 사람이 여러 명인 경우에는 자료가 저장되어 있는 순서에 맞춰 출력된다. (즉, 아스키 코드상의 순서)

- (5) 종료화면 : 시작화면에서 '5'번 입력 시 종료

예시 - FindByBirth

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:2

Han 01011112222 19960302

Kim 0112223333 19950101

Lee 0101234567 19970903

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:4

Birth:1

Kim 0112223333 19950101

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:5

1단계 확정 자료 구조

연락처 : 이름(20bytes) + 연락처(15bytes) + 생일(8bytes)

```
#define MAX_NUM 100    // 전처리기 에서 배울 내용 (상수값 선언)
struct tel
{
    char name[21];
    char tel_no[16];
    char birth[9];
};
main( ) {
    struct tel tel_list[MAX_NUM]; // 최대 100개 가능
    int count ;
}
```

참고 1: 전역변수 사용 금지, `tel_list` 변수와 `count`는 main 함수에서 선언

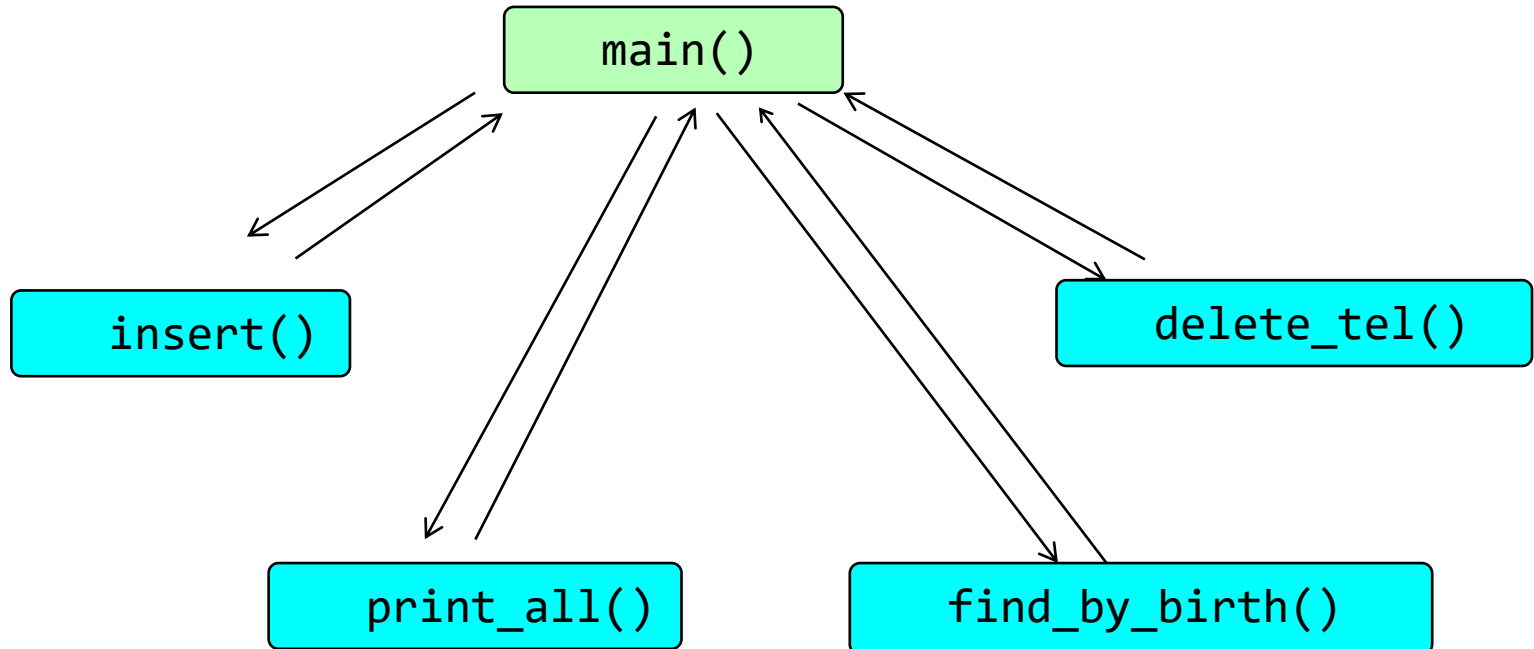
2) 프로젝트 설계 과정 : 제작 및 시험/평가

- 제작
 - 계획한 설계에 따라 프로그램을 구현
 - 코딩

- 시험/평가
 - 구현한 결과 테스트
 - 문제점 분석
 - 해결방안 모색 및 수정

함수 내역

- 필요 함수 : 함수명은 변경 가능 (필요에 따라 인자와 반환값 추가)
 - 메인 함수 : main()
 - 등록 : insert()
 - 삭제 : delete_tel()
 - 보기 : print_all()
 - 생일자 검색 : find_by_birth()

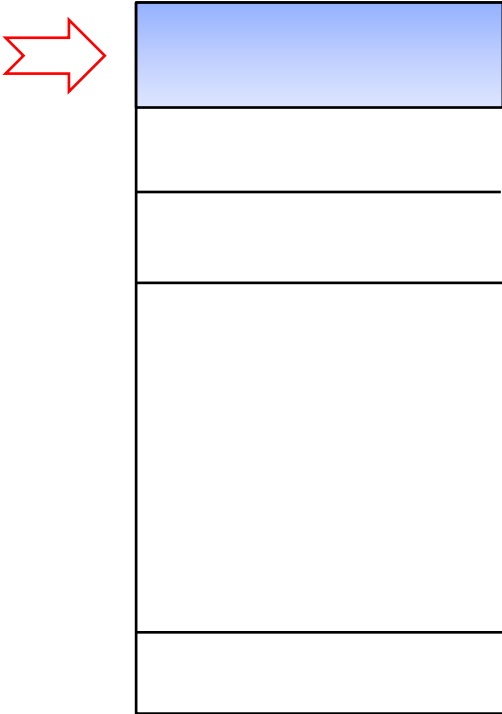


main() 함수 구조 예

```
while (1)
{
    // 시작화면 출력
    // 번호 n 입력
    switch (n)
    {
        case 1: insert(); break;
        case 2: print_all(); break;
        case 3: delete_tel(); break;
        case 4: find_by_birth(); break;
        case 5: return 0 ;
    }
}
```

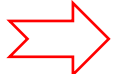
등록 : insert() - 1

1) 초기 : count = 0



등록 : insert() - 2

2) 'SungChunHyang' '0111112222' '19960101' 입력 :

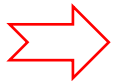


| |
|----------------------|
| SungChunHyang |
| |
| |
| |
| |

tel_list[0] 에 저장되고
count++; // count = **1**

등록 : insert() - 3

3) 'HongGilDong' '01022223333' '20000301' 입력 :



| |
|---------------|
| HongGilDong |
| SungChunHyang |
| |
| |
| |

저장되어 있는 'SungChunHyang' 보다
'HongGilDong'이 이름순으로 앞이므로


tel_list[0] 을 tel_list[1]로 옮기고

'HongGilDong'의 정보가 tel_list[0]에 저장

count++; // count = 2

등록 : insert() - 4

4) 'Lee' '01011113333' '19970101' 입력 :



| |
|---------------|
| HongGilDong |
| Lee |
| SungChunHyang |
| |
| |
| |

저장되어 있는 'SungChunHyang' 보다는 앞,
'HongGilDong'보다는 뒤에 저장되어야 함

```
count++;          // count = 3
```

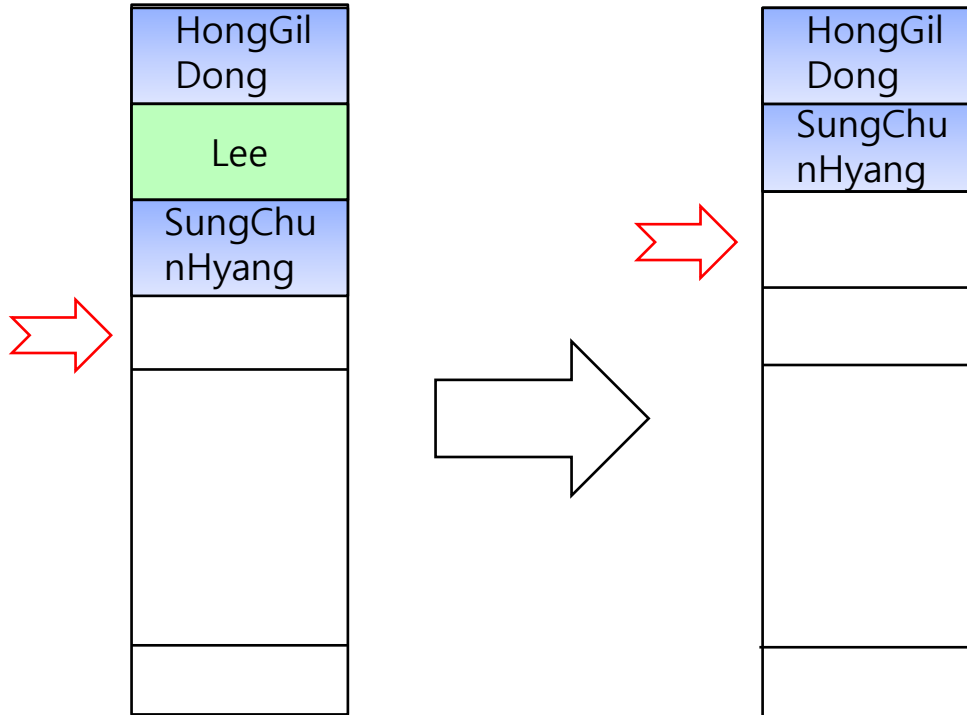
참고 1: 매번 자료가 등록될 때 마다 전체자료를 정렬하지 말고,
삽입될 위치를 찾은 후 나머지 자료를 이동하는 방식으로 한다.

등록 : insert() - 5

- 예외 처리
 - 만일 100개 정보가 다 저장되어있는 상태라면?
→ "OVERFLOW" 출력

삭제 : delete_tel() - 1

- 'Lee' 입력 시



'Lee'이 있던 배열의
위치로 'SunChunHyang'
이하 모든 정보를 옮겨야
한다.

```
count-- ;  
// count = 2
```

삭제 : delete_tel() - 2

- 만일 저장된 연락처가 하나도 없는데 삭제하려고 들어오는 경우
→ "NO MEMBER" 출력
- 없는 이름을 삭제하려고 하는 경우는 그냥 무시하고 메뉴로 복귀

출력 : print_all()

HongGilDong□01011111111□20000301
SungChunHyang□0111112222□19960101

위에서 □는 빈 칸 (공백) 화면 출력을 의미한다.

생일자 검색 : find_by_birth()

3을 입력하면 3월 생 연락처가 모두 출력된다.

HongGilDong□01011111111□20000301

□ 빈 칸

참고: 같은 달이 생일 인 사람이 여러 명인 경우에는 자료가 저장되어 있는 순서에 맞춰 출력. (아스키 코드상의 순서)

3) 단계별 사양 - 1

(1) 공통사항

- 4단계로 나누어 진행
- 1,2 단계는 OJ시스템에서 채점하고 간단한 코드점검이 이루어진다
- 전역 변수 사용은 원칙적으로 금지
- 기능 별로 함수 독립
- 메인 함수는 메뉴 출력 및 함수 호출의 반복문으로 이루어짐

3) 단계별 사양 - 2

정렬해야 됨

(2) 1단계 :

<고정 크기 멤버변수들을 가진 고정 크기 구조체 배열 >

- 연락처 관리 프로그램을 '구조체 배열'을 사용하여 구현
- 이름 :20bytes(마지막 널 문자 제외크기)
- 전화번호 : 15bytes ('-' 없이 입력, 마지막 널 문자 제외크기)
- 생일 : 8bytes (YYYYMMDD형식, 마지막 널 문자 제외 크기)
- 최대 저장 가능 연락처 100개로 설정
- 기능 : 등록, 삭제, 출력, 생일자 검색
- 중복된 이름은 없다고 가정
- 연락처를 추가할 때 마다 이름순(아스키코드상의 순서) 으로 저장이 되도록 등록함수 작성
- OI에 제출 후 코드 검증
- 1단계 설계명세서 참조

1 단계 구조

1단계 끝

```
#define MAX_NUM 100          // 상수 선언

struct tel
{
    char name[21];
    char tel_no[16];
    char birth[9];
};

struct tel tel_list[MAX_NUM]; // 최대 100개 가능
int count ;                   // 현재 연락처 개수
```

참고: 전역변수 사용 금지

코딩1

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAXCONTACT 100

typedef struct ContactInfo {
    char name[21];
    char phone[16];
    char birth[9];
} Contact;

void printMainMenu();
void swapContact(Contact* contactData, int index1, int index2);
void sortContact(Contact* contactData, int numContact);
void insertContact(Contact* contactData, int *numContact);
void deleteContact(Contact* contactData, int *numContact);
void printAll(Contact* contactData, int numContact);
void findContactByBirth(Contact* contactData, int numContact);
```

```

int main()
{
    Contact contactData[MAXCONTACT]; //100의 구조체공간 정의
    int numContact = 0; //요소인덱스
    int menu;

    while (1) {
        printMainMenu(); //메뉴 출력
        scanf("%d", &menu); //메뉴선택

        switch (menu) {
            case 1: insertContact(contactData, &numContact); //등록
                    break;
            case 2: printAll(contactData, numContact); // 보기
                    break;
            case 3: deleteContact(contactData, &numContact); //삭제
                    break;
            case 4: findContactByBirth(contactData, numContact); //생일자 검색
                    break;
            case 5:
                    return 0;
        }
    }
    return 0;
}

```

```
void printMainMenu()  
{  
    printf("*****Menu*****\n");  
    printf("<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>\n");  
    printf("Enter_the_menu_number:");  
}
```

```

void insertContact(Contact* contactData, int* numContact)
{
    if ( ) {
        printf("OVERFLOW\n");
        return;
    }
    printf("Name:");
    scanf("%s", );//No blank
    printf("Phone_number:");
    scanf("%s", );
    printf("Birth:");
    scanf("%s", );
    ;//인덱스 증가

    sortContact(contactData, *numContact);
    // 방금 입력된 데이터를 적절한 위치로 이동
    printf("<<%d>>\n", );

    return;
}

```

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:1

Name:KimEunJoo

Phone_number:0001112222

Birth:19960101

<<1>>

```
void sortContact(Contact* contactData, int numContact)
{
    // contactData의 데이터를 알파벳 순 정렬
    // strcmp 사용
    // swapContact 함수 사용
}
```

호출



```
swapContact(Contact* contactData, int index1, int index2);
```

//Swap Two-Element ---- 구조체 데이터는 배열문을 사용할 수 있음!

정렬은

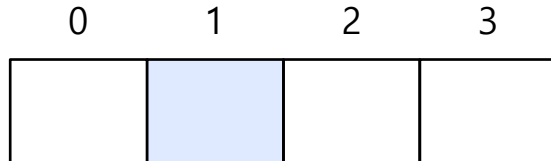
```
데이터가 입력될 때 최초의 데이터는 정렬이 필요 없음
두번째 데이터부터 정렬 {
    두번째 입력한 데이터가 첫번째 데이터와 교환이 필요한 지 뒤에서 부터 비교함!
    교환이 필요하다면 swapContact(contactData, i, i-1) 호출
    // 세번째 데이터가 입력될 때 2개의 데이터는 교환된 상태이고,
    // 세번째 데이터와 두번째 데이터를 비교하고 교환이 필요하다면 함수 호출
}
```

```

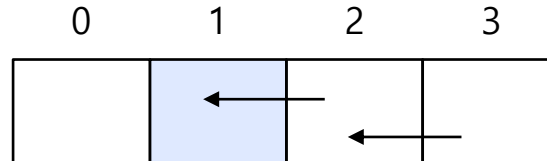
void deleteContact(Contact* contactData, int* numContact)
{
    if (삭제할 데이터가 없으면) {
        printf( " NO MEMBER\n" ); return;
    }
    printf( " Name: " );//이름입력 & 이름과 일치하는 데이터 삭제
    // 반복문을 이용해서 해당 데이터가 있는 인덱스를 찾는다.

    // 데이터가 정렬되어 있으므로, 삭제할 데이터의 인덱스+1의 데이터를
    // 인덱스로 이동시킨다.
    // 데이터의 개수를 감소시킴 (--)
    return;
}

```



삭제할 데이터



인덱스 2, 3의 데이터를 왼쪽으로 1-칸씩 이동

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:3

Name:KimEunJoo

```
void findContactByBirth(Contact* contactData, int numContact) {  
    int birth;  
    printf("Birth:");  
    scanf("%d", &birth); //월  
    // 반복문으로 birth와 contactData를 비교해서 일치하는 data를 출력함!  
}
```

*****Menu*****

<1.Registration><2.ShowAll><3.Delete><4.FindByBirth><5.Exit>

Enter_the_menu_number:4

Birth:1

Kim 0112223333 19950101