

---

# 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 자료 보기 : 출력
  - 임의의 월이 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)
    {
        // 메뉴 출력
        // 원하는 메뉴 선택 입력
        // 종료 메뉴 선택 시 프로그램 종료
        // 해당하는 메뉴 기능을 하는 함수 호출
    }
}
```

## 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)
- 입력값들 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 저장되도록 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 : 원하는 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

- : 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, 생일자 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 설계 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 정렬한다**
  - : 처리 후 다시 시작화면으로 돌아감

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)

## 2) 프로젝트 설계 과정 : 합성

---

- (1) 등록 화면 예외 처리 : 시작화면에서 '1' 선택 시  
: 최대 수용 가능 한 연락처 수(100명)를 초과하면  
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.)

2 

HongGilDong 01011111111 20000301

SungChunHyang 0111112222 19960101

1.Registration

2.ShowAll

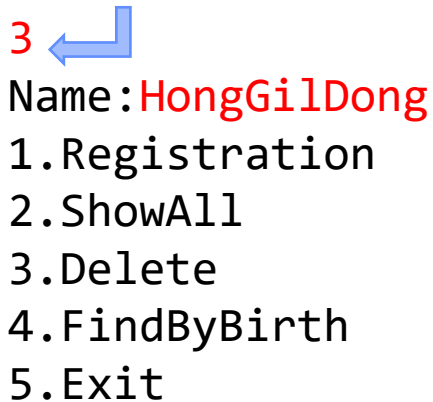
3.Delete

4.FindByBirth

5.Exit

## 2) 프로젝트 설계 과정 : 합성

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




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

- ✓ 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

## 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

# 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

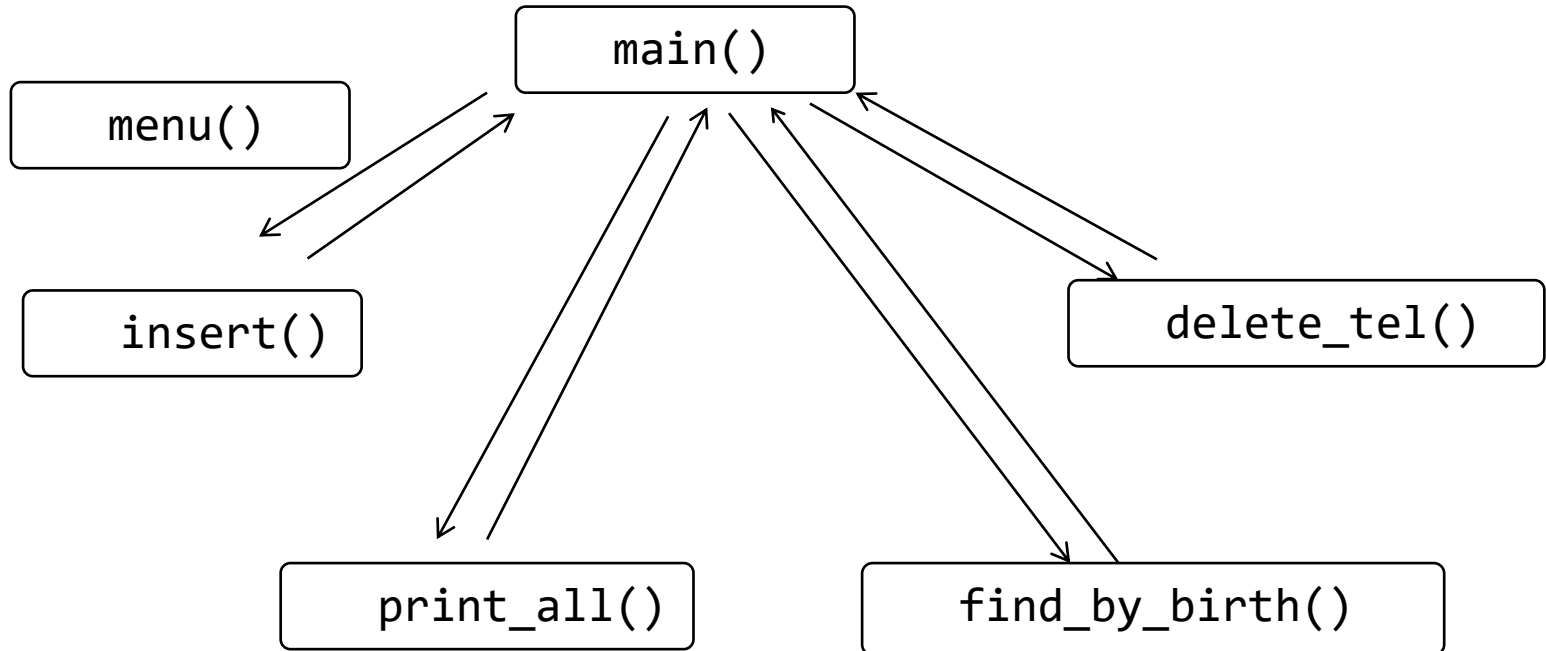
## 2) 프로젝트 설계 과정 : 제작 및 시험/평가

---

- 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

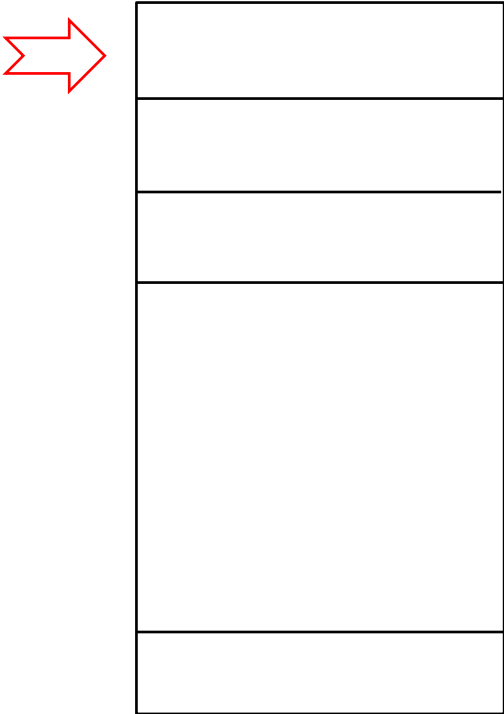
---

```
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 ; // 종료
    }
}
```

# 등록 : insert() - 1

---

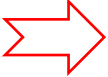
1) 초기 : count = 0



## 등록 : insert() - 2

---

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



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

tel\_list[0] 에 저장되고

count++; // count = 1

<<1>> // 현재 인원수를 화면에 출력한다

## 등록 : insert() - 3

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



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

tel\_list[0] 을 tel\_list[1]로 옮기고

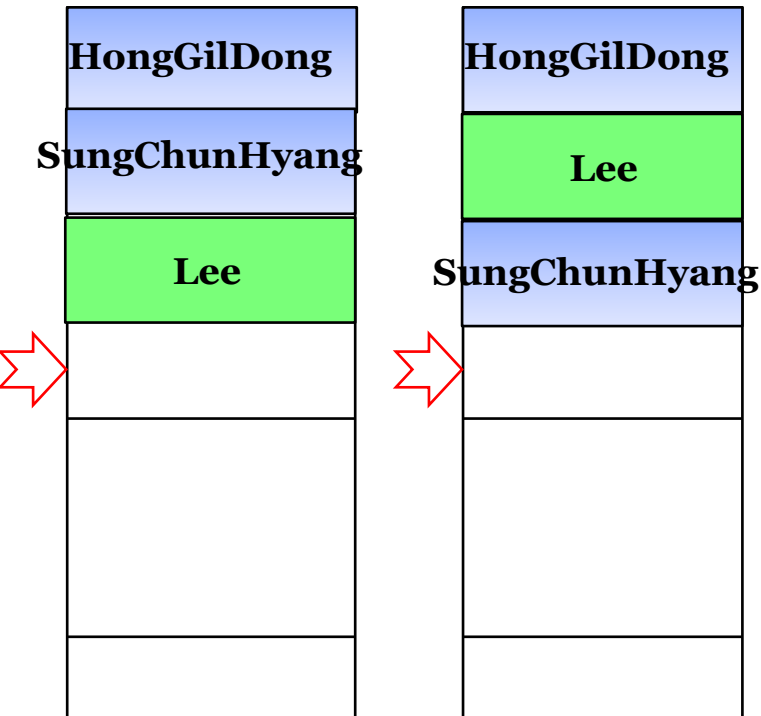
'HongGilDong'의 정보가 tel\_list[0]에 저장

```
count++;    // count = 2
```

<<2>>

## 등록 : insert() - 4

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



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

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

<<3>>

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

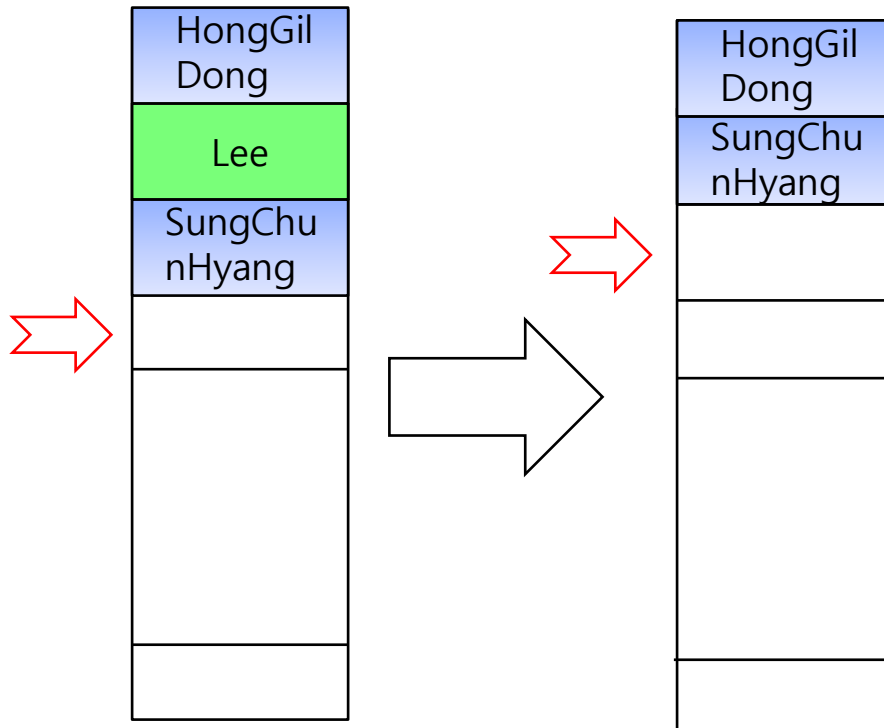
## 등록 : insert() - 5

---

- 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. (사전 순)

### 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

안내된 방법으로 정렬해야 됨

(2) 1단계 : ~~11월 6일 (일)~~ 마감

<고정 크기 멤버변수들을 가진 고정 크기 구조체 배열 >

- 연락처 관리 프로그램을 '구조체 배열'을 사용하여 구현
- 이름 : 20 bytes(마지막 널 문자 제외크기)
- 전화번호 : 15 bytes ( '-' 없이 입력, 마지막 널 문자 제외크기)
- 생일 : 8 bytes (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 = 0;                // 현재 저장되어 있는 연락처 개수
```

참고: 전역변수 사용 금지

### 3) 단계별 사양 - 3

2단계

(3) 2단계 : ~~11월 20일 (일)~~ 마감

< 가변 크기 구조체 배열 및 가변 크기 멤버 변수 사용 >

- 연락처 관리 프로그램을 '구조체 포인터 배열'을 사용하여 구현
- 최대 입력 가능한 숫자를 프로그램 실행 제일 처음 입력
- 이 입력 숫자 만큼 구조체 포인터 배열을 할당
- 멤버변수들을 포인터로 선언
- 멤버변수들은 최대 100 bytes까지 저장 가능하도록 하되, 포인터로 선언(즉, 필요한 크기만큼 메모리 할당), 12장 예제3 참조.
- 기능 : 등록, 삭제, 출력, 생일자 검색 (1단계와 동일)
- 중복된 이름은 없다고 가정
- 연락처를 추가할 때 마다 이름순으로 저장이 되도록 등록함수 작성
- OI에 제출 후 코드 검증
- 2단계 설계명세서 참조

## 2 단계 구조

```
struct tel
{
    char *name;
    char *tel_no;
    char *birth;
};
typedef struct tel TEL;
TEL **tel_list;

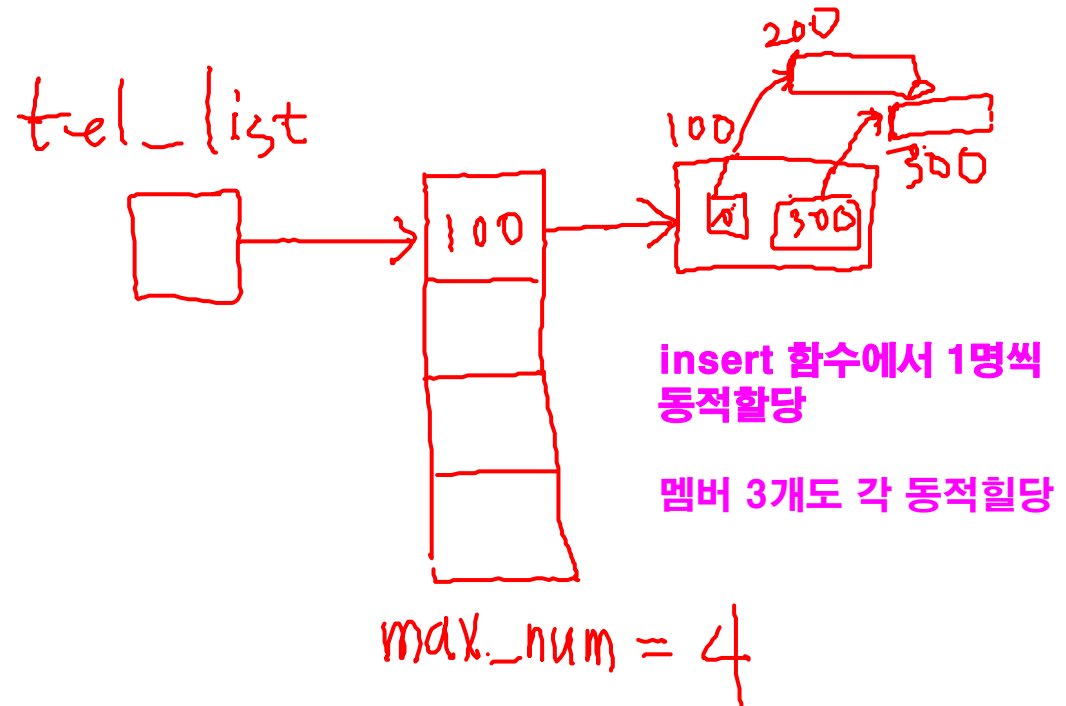
int max_num;           // 최대 회원수, 사용자로부터 입력
int count = 0 ;        // 현재 연락처 개수
```

## 이중 포인터 사용 이유: 메모리를 효율적으로 사용하기 위한 2 단계 구조

```
struct tel
{
    char *name;
    char *tel_no;
    char *birth;
};

typedef struct tel TEL;

TEL **tel_list;
```



### 1) max\_num 입력

- 2) `tel_list = (TEL **)malloc(sizeof(TEL *) * max_num);`
- 3) 새로운 연락처가 등록 될 때 마다 `tel_list[ ]`에 하나씩 **TEL** 메모리 할당
  - 구조체 내부 변수들을 위한 최소한의 메모리 할당 (최대 100)
  - 즉 이름, 번호, 생일은 널문자 포함 최대 100 글자이다.



## 2 단계 주의 사항

2단계 끝

- 등록 시 tel\_list[] 에 하나씩 TEL 구조체 메모리를 할당 후 내부 멤버들에 대해서 최소한의 메모리를 할당하여 처리한다.
- 삭제 시에는 반대로 내부멤버들에 대한 메모리를 먼저 free()하고, 그 이후 할당되어있는 구조체 메모리를 free()한다.
- 최대 저장 가능한 연락처의 개수를 맨 처음 입력하여 (max\_num), 그 수가 넘어가는 연락처를 등록하려면 OVERFLOW처리를 해야 한다.

### 3) 단계별 사양 - 4

3단계

(4) 3단계 : ~~11월 27일 (일)~~ 마감

파일에서 1명 읽어올 때  
멤버 각각 동적할당

< 파일 입출력을 2단계 산출물에 추가 >

- 기능추가:

- 1) **파일**에 저장되어있는 연락처 정보를 하나씩 읽어 들여  
구조체 포인터 배열에 추가(이름순 정렬 동일)하는 기능

**RegFromFile()** , **read mode**

- 2) 프로그램 종료 시 구조체 포인터 배열에 저장되어있는 정보들을  
연락처 **파일에 저장**하는 기능 , **write mode**

- 연락처 파일 명 : PHONE\_BOOK.txt
- 처음 프로그램을 실행하는 경우에는 연락처 파일 생성
- 개별 채점
- 3단계 명세서를 참조할 것

### 3 단계 주의 사항

---

- 파일에는 연락처의 정보들이 줄 단위로 차례차례 저장되어 있고 총 몇 개의 연락처인지는 알 수 없다.
  - 파일로 부터 등록하려면, **파일의 끝까지 처리**하여야 한다.
  - 파일 마지막에는 개행 문자가 있다고 가정한다.
- **파일에서** 읽어 들인 연락처는, **등록메뉴**를 통하여 입력된 연락처와 **같이 처리**가 되어 프로그램 정상 종료 시에 **모두 파일에** 입력된다.

즉, 프로그램 실행 후 등록 메뉴를 통하여 몇 명 등록시키고, 그 이후 파일을 읽어 입력하는 경우에도 무리 없이 동작해야 한다.

sort() 함수 필요. insert 와 RegFromFile에서 필요

### 3) 단계별 사양 - 5

4단계

(5) 4단계 : ~~12월 4일(일)~~ 마감

< 분리 컴파일을 적용 >

- `insert()`, `delete_tel()`, `print_all()`, `find_by_birth()`, `RegFromFile()`  
함수를 각각 다른 .c파일에 저장 (전체 6개)  
예) `main.c`, `insert.c`, `delete.c`, `print.c` 등
- tel 구조체 자료형 정의는 `my_struct.h`에 저장
- tel 구조체 자료형 재정의(`typedef`)는 `my_define.h`에 저장
- 여러 파일에서 사용되는 공통된 함수원형들은 `my_func.h`에 저장
- .c 파일에서는 필요한 헤더파일을 적절히 `include`
- 개별 채점
- 4단계 설계명세서를 참조 (예시이므로 수정, 삭제, 추가가능)

## my\_struct.h (예시)

---

```
#ifndef MY_STRUCT_H
#define MY_STRUCT_H
struct tel {
    ... };

#endif
```

my\_struct.h

#include "my\_struct.h"

insert.c

#include "my\_struct.h"

delete.c

- struct 선언을 공유해야 하는 소스파일들은 **하나의 헤더파일**에 struct 선언문을 담아 두고 이 헤더파일을 공유한다.

# my\_func.h (예시)

```
#ifndef MY_FUNC_H
#define MY_FUNC_H
int insert( ... );
int delete_tel( ... ); ...
#endif
```

my\_func.h

insert.c 에 insert()가 정의  
delete.c에 delete\_tel()가 정의

main에서 insert, delete\_tel를 호출  
하므로 이들 함수의 원형을 선언해  
주어야 한다.

main.c에서 이를 그냥 나열해도 되  
지만, 만일 insert와 delete\_tel함수를  
다른 함수들에서도 호출하게 된다면  
헤더파일에 원형을 선언해 두고, 이  
를 공유해서 처리한다.

```
#include "my_func.h"
```

main.c

## my\_define.h (예시)

```
#ifndef MY_DEFINE_H
#define MY_DEFINE_H
typedef struct tel TEL ;
#endif
```

my\_define.h

이 부분은 각 파일별로 필요한 헤더를 추가

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "my_define.h"
```

insert.c

*myfunc.h*

```
#include "my_define.h"
```

delete.c

**TEL**이라는 형선언문을 공유하기 위함

# 간단한 방법: **my\_func.h** 에 필요한 것을 모두 넣어두고 각 c 파일에서 **my\_func.h** 만 include 한다

```
#ifndef MY_FUNC_H
#define MY_FUNC_H
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "my_struct.h"
#include "my_define.h"
int insert( ... );
int delete_tel( ... ); ...
#endif
```

**my\_func.h**

```
#include "my_func.h"
int main(){

}
```

**main.c**

```
#include "my_func.h"
int insert( 인자 ){

}
```

**insert.c**

```
#include "my_func.h"
int insert( 인자 ){

}
```

**delete.c**



## 추가 기능 (심화반)

---

- 여러 가지 추가 기능을 가능한 학생들은 넣어보도록 예)
  - 1) 생일 정보를 검색하여 1997년 이후 출생자들의 정보를 화면에 출력
  - 2) 전화번호가 010이 아닌 사람들의 정보를 화면에 출력
  - 3) 이름순이 아니라 생일 순으로 정렬된 순서로 정보를 화면에 출력
  - 4) 구조체내에 level변수를 추가하여, VIP, GOLD, SILVER 레벨별로 회원을 출력
  - 5) 이름으로 검색하여, 전화번호와 생일을 변경

## 4) 일정

|     | 구현내용                      | 마감 일시               | 비고               |
|-----|---------------------------|---------------------|------------------|
| 1단계 | 기본기능<br>구조체 배열로 구현        | <del>11/6(일)</del>  | OJ에서 채점<br>+코드검사 |
| 2단계 | 기본기능<br>구조체 포인터 배열로<br>구현 | <del>11/20(일)</del> | OJ에서 채점<br>+코드검사 |
| 3단계 | 2단계 + 파일입출력               | <del>11/27(일)</del> | 개별채점             |
| 4단계 | 3단계+분할컴파일 +<br>추가기능       | <del>12/4(일)</del>  | 개별채점             |