

소프트웨어공학

1차 과제 보고서

학과 : AI소프트웨어학과

학번 : 202322371

이름 : 하승연



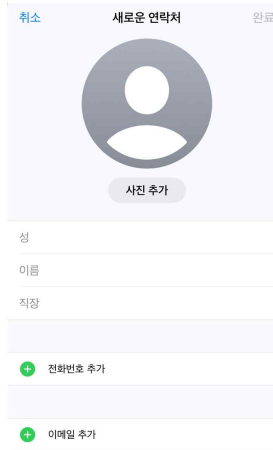
목 차

1. 전화번호부 틀 설정에 관한 고찰
2. 전화번호부 코드와 순서도를 통한 과제 해결 과정 탐색
 - 1) 전화번호부 구축을 위한 데이터 관리 방법
 - 2) 전체 진행 과정
 - 3) 주요 함수 진행 과정
 - (1) AddNumBook 함수
 - (2) SearchNumBook 함수
 - (3) PrintNumBook 함수
 - (4) ReadFileLine 함수
 - (5) WriteOnFile 함수
 - 4) 서브 함수 및 예외 처리 함수 진행 과정
 - (1) rightkey 함수
 - (2) rightsize 함수
 - (3) SortName 함수
 - (4) IsSameName 함수
 - (5) IsSameNum 함수
 - (6) IsSameMail 함수
 - (7) IsNum 함수
 - (8) IsMail 함수
 - (9) IsNoEnter 함수
 - (10) IsNoSpaceBar 함수
3. 전화번호부 코드의 진행 예시
4. 과제 후 느낀 점과 보완 사항

1. 전화번호부 틀 설정에 관한 고찰

전화번호부의 틀을 만들기 위해 나의 핸드폰 연락처를 열어보았다. 별것 아니라고 생각했던 연락처는 생각보다 꽤 많은 기능이 있었다. 그중 내가 자주 쓰는 기능들을 정리하여 이번 과제를 완수하리라고 마음먹었다.

1) 전화번호부 추가 기능



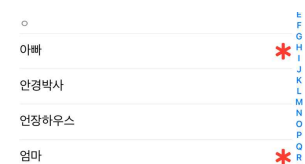
나의 핸드폰 연락처 추가 기능에 들어가면 입력을 받을 수 있는 칸은 다음과 같다. 성, 이름, 직장, 전화번호, 이메일 등이 있다.

이 중 성과 이름을 나누는 칸은 사용하면서 성가셨기에 성과 이름을 합친 이름만 받기로 생각하였다. 전화번호부 추가 칸은, 전화번호부라면 전화번호가 빠지는 것은 말이 안 된다고 생각하여 필수적으로 받아야겠다고 구상했다.

나는 직장 칸에 꼭 이 사람의 직장만 쓰는 것이 아니라, 이 사람과의 관계를 나타내는 항목으로 자주 이용하였다. 따라서 직장 대신 관계 항목을 추가하여 사람의 특징을 좀 더 담을 수 있는 항목을 구성하였다.

전화번호부가 사람의 온라인 주소를 저장하여 이를 통해 상대방과 연결을 시켜주는 기능이라고 했을 때 가능한 주소는 여러 가지가 있는데 그중 하나가 이메일이라고 생각한다. 하지만 모든 사람의 이메일을 아는 것은 매우 귀찮은 일이라고 생각하여 필수적으로 받을 칸으로는 구성하지 않았다.

나의 연락처 추가 기능에서는 이 정도 메뉴가 다였다. 하지만 나는 추가로 긴급연락처 기능을 제작하기로 하였다.



긴급연락처란, '내가 문제가 생기면 여기로 연락을 남겨주십시오 ~' 라고 말할 수 있는 기능이다. 이 기능은 연락처에 필수적으로 필요한 항목은 아니지만 나의 건강에 문제가 생겨 응급실에 가거나, 아니면 핸드폰을 분실했을 때 나에게 연락하는 방법으로 쓰인다. 이러한 경우를 고려했을 때 쓴다면 꽤 쓸모 있는 기능이라고 볼 수 있다.

보통 연락처에 있는 전화번호를 다시 한번 저장하게 되면 이런 현상이 나타난다.



위 사진은 이미 저장 되어있는 내 번호를 다시 추가하기 위해 번호를 입력해 본 것이다. 이미 저장된 번호는 번호가 이미 저장되었다고 뜨게 된다. 나는 이러한 특징을 이메일과 이름까지 확장하여, 같은 이름이나 전화번호, 이메일은 이미 저장된 항목이 있다고 출력하게 만들겠다고 구상했다. 하지만 이름은 동명이인과 같은 특수한 케이스가 있어서 저장은 가능하지만, 알려주는 방식으로 구성하고, 전화번호와 이메일은 같은 경우가 없도록 구상했다.

2) 전화번호부 검색 기능

전화번호를 검색하기 위해 검색창에 들어가면 검색어를 입력하는 칸이 하나 뜬다.



이 검색창에는 특정 항목만 검색되는 것이 아니라, 연락처, 관계, 이메일 등 추가 메뉴에서 구성한 어떤 것을 입력해도 연관된 목록들이 다 나오게 된다.

심지어 이름의 일부나 전화번호의 일부 등 문자열의 일부만 검색해도 나오는 것을 보아, 나도 연락처 검색 기능에서 아무거나 검색해도 관련된 항목들은 다 나올 수 있도록 설정해야겠다고 생각했다. 또한 스페이스 등의 기호를 검색했을 때는 다음과 같은 창이 나왔다.



‘ ’에 대한 결과 없음

맞춤법을 확인하거나 새로운 검색을 시도하십시오.

검색된 목록이 없었을 때, 검색된 항목이 없다고 말해주는 기능은 생각해 보면 당연할지도 모른다. 이러한 표시가 뜨지 않게 되면, 잠깐 렉 먹었나? 등의 생각을 할 수도 있기 때문이다. 따라서 나도 검색된 항목이 없으면 없다고 말해주는 친절한 프로그램을 만들어야겠다고 구상했다.

다음 메뉴인 전화번호 수정, 삭제는 이 메뉴에 함께 넣었다.

내 연락처에서 전화번호 수정과 삭제는 전화번호 검색이 선행되어야 가능하다. 따라서 나도 삭제와 수정을 할 때 검색을 선행시킨 후 항목을 선정하여 수정 또는 삭제할 수 있는 기능을 만들어야겠다고 생각했다.

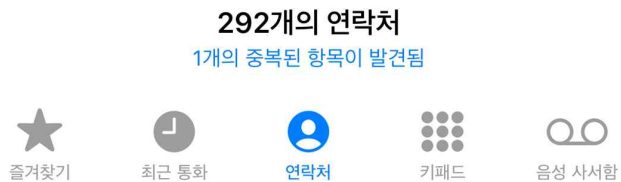
수정 기능에서는 추가 기능과 마찬가지로 동일한 이름, 연락처, 이메일이 이미 있는지 검사하는 과정을 넣었다. 또한 수정과 삭제 기능 후에 정상적으로 수정, 삭제 기능이 되었다는 안내 메시지를 넣어 혼동을 방지해야겠다고 생각했다.



3) 전화번호부 출력 기능

전화번호부 출력은 2가지로 구성했다. 1. 전체 전화번호부 출력, 2. 긴급연락처 출력이다.

전화번호 출력 기능은 보통 사전식으로 출력하기 때문에, 효율적인 관리를 위하여 모든 메뉴 이전에 이름을 기준으로 정렬하는 정렬 함수를 선행시켜야겠다고 생각했다.



또한 연락처 출력에는, 총 몇 항목이 있는지 알려주는 기능이 있는 것을 보았다. 이를 알면 연락처의 제대로 된 저장 여부를 확인할 수 있어서 출력 과정 중 목록의 개수를 알려주도록 구성했다.

긴급연락처는, 긴급 상황일 때 빠르게 연락이 가능하도록 출력 메뉴를 따로 구성했다.

2. 전화번호부 코드와 순서도를 통한 과제 해결 과정 탐색

1) 전화번호부 구축을 위한 데이터 관리 방법

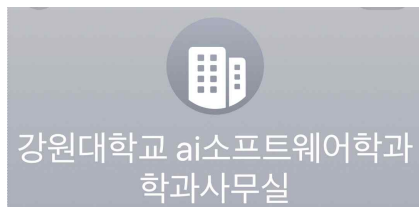
전화번호부 코드를 제작하기 위하여, 나는 무슨 데이터형을 사용하여 각 항목을 저장할까에 관한 생각을 했다.

가장 먼저 생각난 것이 배열이다. 배열은 인덱스 번호를 통하여 변수들의 이름을 각각 저장하지 않아도 총개수만 안다면 반복문을 통한 접근이 빨라서 배열을 사용하겠다고 생각했다.

또한 나는 이름, 연락처, 관계, 이메일, 긴급연락처 저장 여부라는 총 5가지 멤버를 사용하므로 구조체를 통해 같은 인덱스로 다른 5가지의 멤버를 한꺼번에 다룰 수 있게 만들어야겠다고 생각했다. 따라서 나의 데이터형을 다루는 구조체 정의는 다음과 같다.

```
struct base {  
    char name[40]; // 이름  
    char num[12]; // 전화번호  
    char relation[40]; // 관계  
    char mail[30]; // 메일  
    char sos[2]; // 긴급연락처 여부 안내  
};
```

이름과 관계는 서로 비슷한 개념으로 쓸 수 있으므로 같은 크기만큼 지정했다. 처음엔 사람 이름은 보통 3칸이니까 10칸 정도로 지정하면 부족하지 않겠지? 라고 생각하며 내 연락처를 확인했다.



나는 연락처를 저장할 때, 사람 이름만 저장하는 것이 아니라 기관명 등 특징도 다 이름에 저장하는 특성이 있던 것이다. 이러한 특수 경우를 고려했을 때 40칸을 기준으로 배열의 크기를 설정했다.

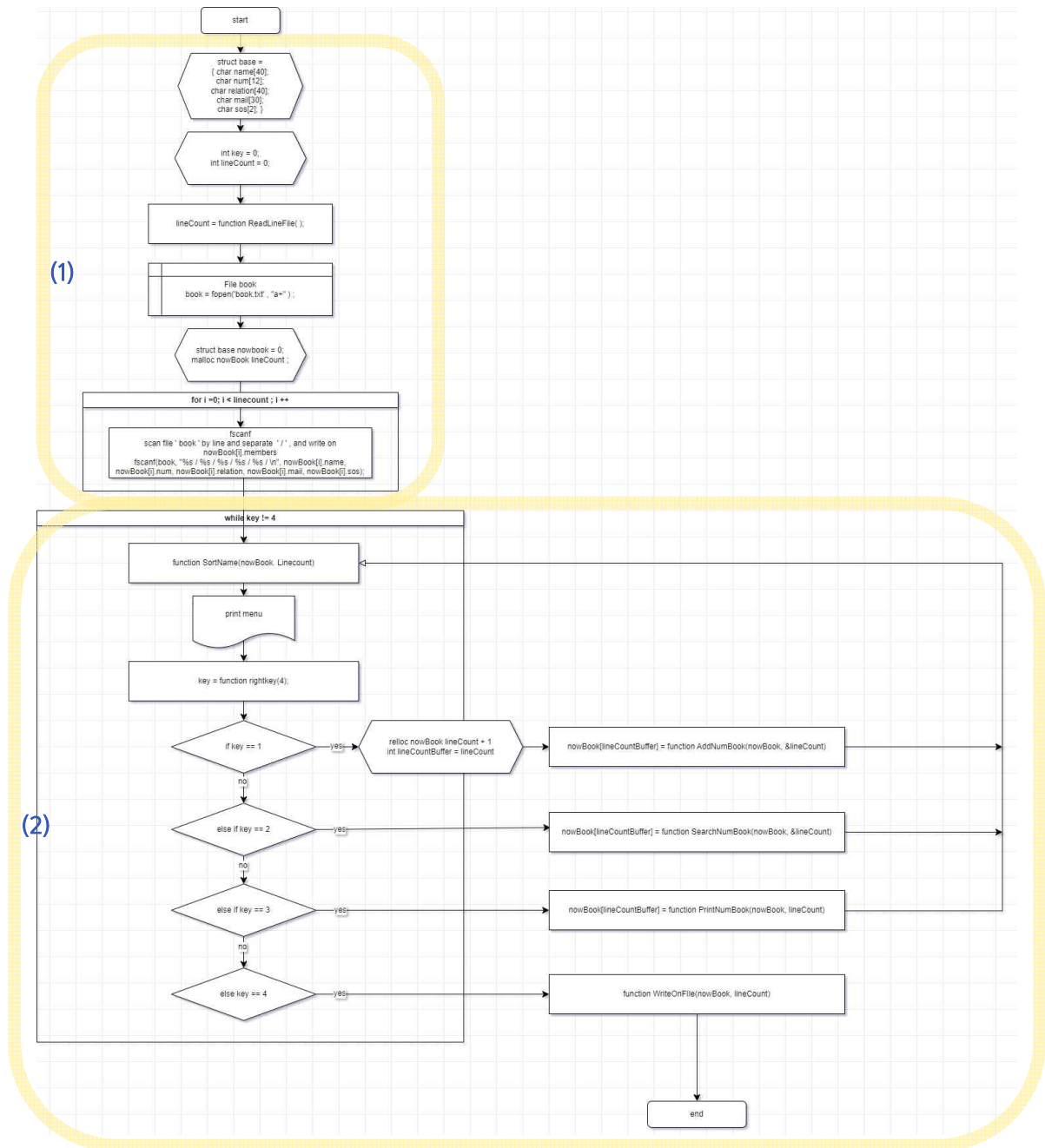
전화번호는 숫자 11자리와 널 문자를 저장할 공간인 12 자리로 지정했다. 또한 문자열 비교 함수를 사용하기 위해 정수형이 아닌 문자형으로 데이터형을 받았다.

이메일은 한국인이 한국 이름을 영어로 쳤을 때의 최대 공간인 9 와, 생년월일을 쳤을 때의 6자리, 강원대학교 메일 주소를 기준으로 뒤에 들어가는 @kangwon.ac.kr 의 14자리, 널 문자 1자리를 고려하여 30을 크기로 잡았다.

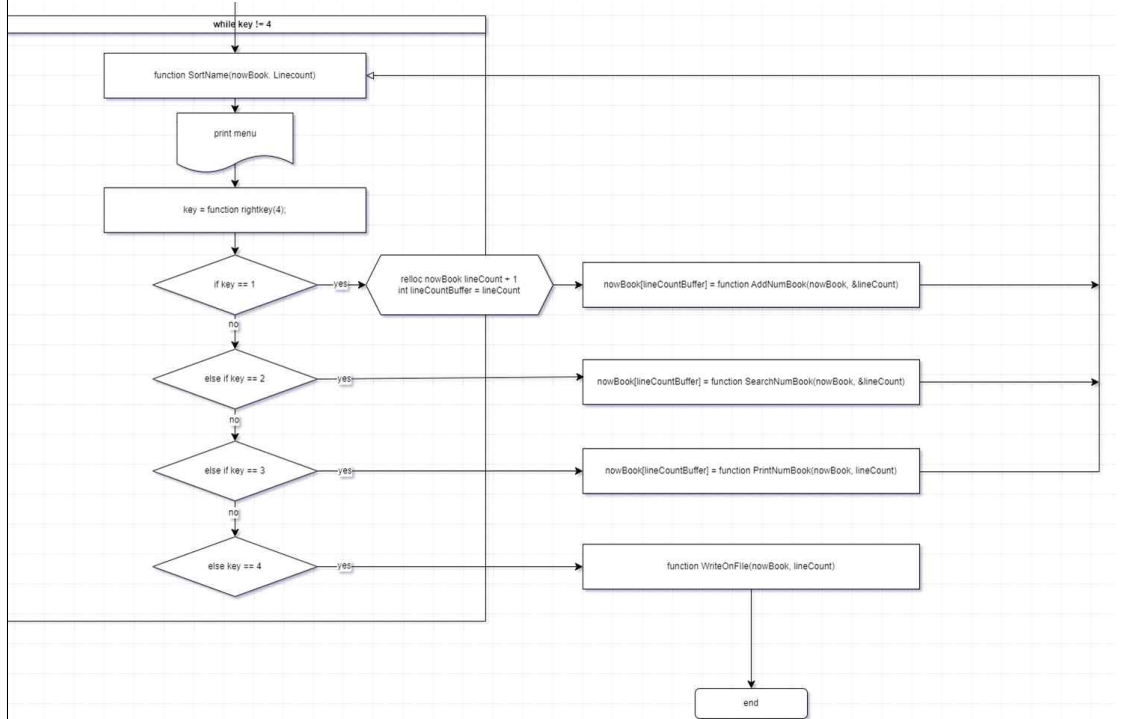
과제의 기존 요구사항인 종료 후 재실행 후에도 기존의 항목이 남아 있어야 하는 텍스트 파일을 어떻게 활용해야 할지도 고민했다. 초기 구성은 프로그램 진행 도중에도 추가, 수정, 삭제되는 항목이 파일에 바로 업데이트될 수 있도록 파일 자체를 움직이는 방식을 선택했다. 하지만 이는 파일을 계속하여 열었다 닫았다 하며 주 메모리에서 보조 메모리에 계속해서 접근하여 효율적인 방식이 아니라는 판단이 들었다. 따라서 시작할 때 파일에서 읽고, 종료할 때 파일에 쓰는 방식을 선택했다.

2) 전체 진행 과정

파일의 전체 진행 과정을 나타낸 순서도는 다음과 같다.



(2)



```
while (key != 4) {
    SortName( nowBook, lineCount); // 이름 정렬 함수

    printf("%s", "===== 메뉴 ===== \n");
    printf("%s\n%s\n%s\n%s\n", "01. 추가", "02. 검색 과 수정/삭제", "03. 출력", "04. 종료 및 저장");

    key = rightkey(4) ;

    if (key == 1) {
        int lineCountBuffer = lineCount;
        nowBook = (struct base*) realloc( nowBook, sizeof(struct base) * (lineCount + 1) );
        nowBook[lineCountBuffer] = AddNumBook(nowBook, &lineCount);

        ScreenClear();
    }
    else if (key == 2) {
        nowBook = SearchNumBook(nowBook, &lineCount);

        ScreenClear();
    }
    else if (key == 3) {
        PrintNumBook(nowBook, lineCount);

        ScreenClear();
    }
    else if (key == 4) {
        printf("%s", "===== 04. 종료 및 저장 ===== \n");
        printf("%s", "... 데이터를 'book.txt' 파일에 저장합니다. \n");
        WriteOnFile(nowBook, lineCount);
        ScreenClear();

        free(nowBook); // 동적메모리 해제
        nowBook = NULL;
    }
}
```

전화번호부 메뉴를 진행한다. 출력 과정을 통해 메뉴에 대한 항목 키와 각 메뉴가 무엇을 하는 지 간단히 쓴다.

위 함수는 4. 종료 및 저장을 누를 때까지 반복한다.

SortName 함수를 통해 가지고 있는 전화번호부를 사람 이름을 기준으로 사전식 정렬한다.

rightkey() 함수를 통해 메뉴에서 쓰이는 1부터 4 사이의 숫자만 입력받는다.

만약, key == 1 이면, 추가 메뉴로 진입한다. 추가 메뉴에서 AddNumBook 함수는 줄 수(lineCount, 연락처의 총 개수)를 + 1 하여 변화시킬 수 있다.(정상 추가가 된 경우만) 따라서 함수 실행 전에 lineCount를 lineCountBuffer 에 담아 함수 실행 후 리턴 되는 추가된 newRecord 을 담을 nowBook 구조체 배열 변수의 인덱스 번호로 이용한다. 인덱스 번호는 0부터 시작하므로, 추가 메뉴를 종료한 후 추가한 구조체를 담을 인덱스 번호는 1이 추가되기 전의 lineCount 가 되어야 하기 때문이다.

key == 2 이면 검색 및 수정 삭제 메뉴로 진입한다. 수정 및 삭제는 SearchNumBook 함수를 이용한다. 수정 및 삭제 후 SearchNumBook 함수의 결과인 수정된 nowBook 값을 nowBook에 저장한다.

만약, key == 3이면 출력 메뉴로 진입한다. 출력은 PrintNumBook 함수를 이용한다.

key == 4면 프로그램을 종료 및 저장한다. 저장한다는 출력 메시지가 뜨고, WriteOnFile 함수를 통해 nowBook 구조체 배열 안의 내용들을 파일에 옮긴다. 그 후 할당된 동적 메모리를 해제하며 프로그램이 종료된다.

매 메뉴의 끝에는 화면을 clrer 하는 ScreenClear(); 함수가 존재한다.

```
#include "numbook.h"

void ScreenClear( void ) {

    printf("... 아무 키나 눌러주세요.\n"); // 메뉴를 완수했을 때, 메뉴가 끝났음을 알리기 위한 화면 클리어 함수
    getchar(); // 키를 누르게 하여 화면이 바로 지워지는것을 방지함

    printf("작업 저장중"); // 작업 저장중이라는 안내메세지 출력

    for (int i = 0 ; i < 3; i++) {
        printf("%c", '.');
        Sleep(800); // 0.8초씩 3번 점을 출력하여 화면이 바로 지워지는것을 방지함
    }

    system("cls"); // 화면을 깨끗하게 지우는 함수
}
```

위 함수는 아무 문자를 받아들인 후,

작업 저장 중 이라는 멘트와 함께 0.8초에 한 번씩 점이 3번 출력된다.

그 이후 화면을 깨끗하게 지운다.

프로그램에 기능을 추가할수록, 화면이 난잡해진다는 단점이 있었다.

따라서 화면을 지우는 기능을 통해 프로그램의 가독성을 높였다.

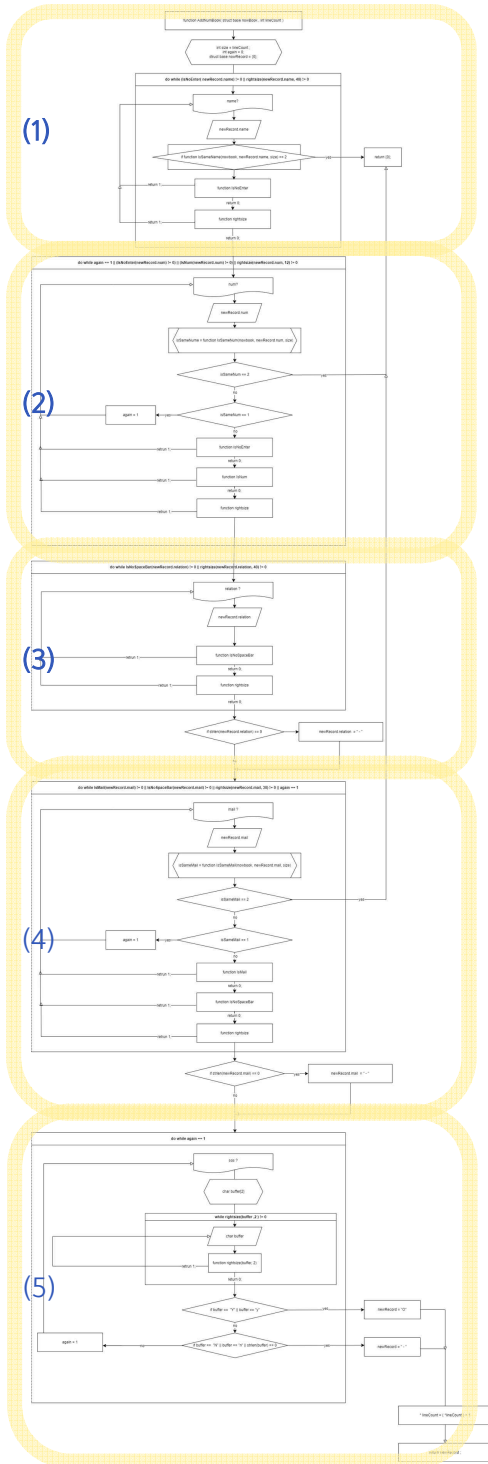
또한 프로그램의 진행이 너무 빠르게 되면 오타나 메뉴와 맞지 않는 것을 입력하는 등의 생각지도 못한 입력 실수가 발생했다. 따라서 이러한 점을 방지하게 위해 0.8초씩 3번의 시간 기말 과제를 두어 프로그램 진행이 빠른 것을 잠시 막는 역할을 하였다.

3) 주요 함수 진행 과정

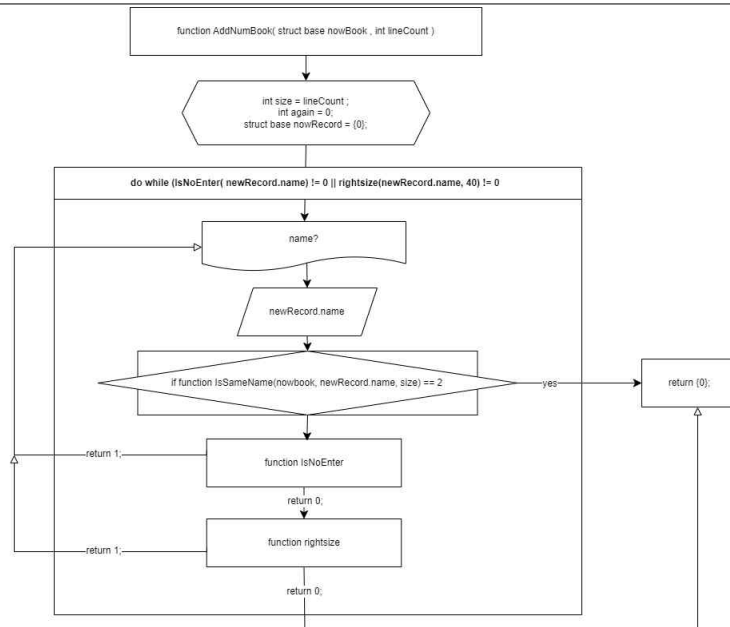
파일의 전체 진행 과정을 나타낸 순서도는 위에서 보았다. 이제는 메뉴에 중요하게 사용되는 함수에 대해 보도록 하자

(1) AddNumBook 함수

AddNumBook 함수의 전체 순서도는 다음과 같다. 총 (5) 가지 단계로 나눌 수 있다.



(1)



```
#include "numbook.h"
// 전화번호부에 새로운 연락처를 저장하는 함수
struct base AddNumBook(struct base* nowBook, int* lineCount) {
    int size = *lineCount; // 포인터를 직접 사용하지 않도록, 줄 수를 새로운 변수에 입력 받음
    int again = 0; // 동일한 목록이 이미 전화번호부에 존재할 때 다시 입력받게 하는 변수 선언
    struct base newRecord = { 0 }; // 새로운 연락처를 입력받을 구조체 변수 선언
    printf("%s", "----- 이, 추가 ----- \n");
    // 이름 추가
    do {
        strcpy(newRecord.name, "00"); // 구조체를 초기화
        printf("%s", "이름? \n");
        scanf("%[^\n]s", newRecord.name);
        getchar();
        if (IsSameName(nowBook, newRecord.name, size) == 2) { // 기존의 목록에 입력한 이름이 존재할 때, 새로운 이름을 입력하지 않겠다고 선택하면 메뉴 화면으로 보냄
            return { 0 };
        }
    } while (IsNoEnter(newRecord.name) != 0 || rightsize(newRecord.name, 40) != 0);
    // 이름에 공백이 없는지, 크기에 맞게 입력했는지 검사하고 아니면 다시 입력하게 함
    // 전화번호부 추가
}
```

맨 처음 전화번호 입력에 들어가게 되면, 입력을 받을 구조체 변수 newRecord를 선언한다. 또한 again 변수를 0으로 초기화 한 후 선언하여, 이후 while문에서 again == 1 일 때 계속 반복하게 하는 조건식 용도의 변수를 만든다.

이후 이름 추가부터 순차적으로 이루어진다. 구조체의 각 멤버별로 요구하는 예외 처리 사항이 달라서 while문 안의 함수는 다 다르다.

먼저 do-while문의 시작에서는 이름? 이라는 안내 메시지가 출력되고 공백과 스페이스바, 기호 등을 포함한 이름 문자열을 newRecord.name 에 입력받는다. 이후 if 문에서 IsSameName 함수를 통해, 작성한 이름 문자열이 이미 전화번호부 목록에 있는지 검사한다. 만약 있다면 IsSameName 함수 내부에서 그대로 저장할 것인지, 아니면 저장 하지 않을 것인지 물어본다.

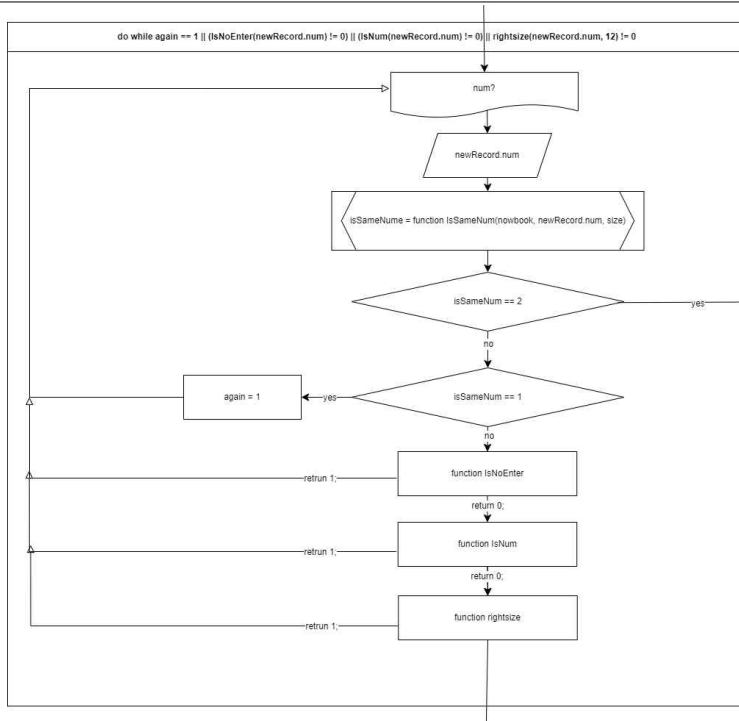
저장 하지 않는다고 하면 {0} 값을 보내 AddNumBook 함수에서 탈출하게 되며 저장 한다고 하면 그대로 진행한다.

while문의 조건식 단계에서는 입력된 문자열이 제대로 된 값인지 검사한다.

IsNoEnter 함수를 통해 공백, 스페이스바를 입력했을 경우 다시 입력하도록 한다.

rightsized 함수를 통해 40개 이상의 문자를 입력했을 때 다시 입력하도록 한다.

(2)



```
do {
    again = 0;
    strcpy(newRecord.num, "00");
    printf("%s", "전화번호? ");
    scanf("%12s", newRecord.num);
    getchar();
    int isSameNum = isSameNum(nowBook, newRecord.num, size); // 기존의 목록과 새로 입력받은 전화번호를 검사하고 값을 리턴받음

    if (isSameNum == 2) { // 기존의 전화번호와 같은 목록이 존재하여 새로운 전화번호를 입력하지 않겠다고 선택하면 메뉴 화면으로 보냄
        return { 0 };
    }
    else if (isSameNum == 1) { // 기존의 전화번호와 같은 목록이 존재하여 새로운 전화번호를 입력하겠다고 하면 새로 입력하게 함
        again = 1;
    }
} while (again == 1 || (isNoEnter(newRecord.num) != 0) || (isNum(newRecord.num) != 0) || (rightsize(newRecord.num, 12) != 0));
// 전화번호에 공백이 존재하는지, 숫자로만 입력했는지, 크기에 맞게 입력했는지 검사하고 아니면 다시 입력하게 함
```

이름 추가가 완료되면 같은 원리로 전화번호가 추가되게 된다.

형식은 다르지 않다. 전화번호를 입력하라는 안내 메시지가 출력된다. 공백, 스페이스, 특수기호 등을 포함한 문자열을 newRecord.num 에 받는다.

이미 전화번호부에 같은 전화번호가 있는지 검사하기 위해

isSameNum 변수에 isSameNum 함수를 사용하여 리턴값을 집어넣는다.

만일 이미 같은 전화번호가 전화번호부에 있고, 다른 번호를 입력하겠다고 하면 isSameNum 에는 1이 들어간다.

만일 이미 같은 전화번호가 전화번호부에 있고, 다른 번호를 입력하지 않겠다고 하면 isSameNum 에는 2가 들어가고 추가 메뉴는 저장되지 않고 종료된다.

입력한 전화번호와 같은 내용인 전화번호가 전화번호부에 없으면 0이 들어가며, 이후 단계를 수행한다.

while문의 조건식에서는 입력된 문자열이 전화번호의 형태가 맞는지 검사한다.

먼저 isNoEnter함수를 통해 문자열에 엔터나 스페이스바가 들어갔는지 검사하고 있다면 다시

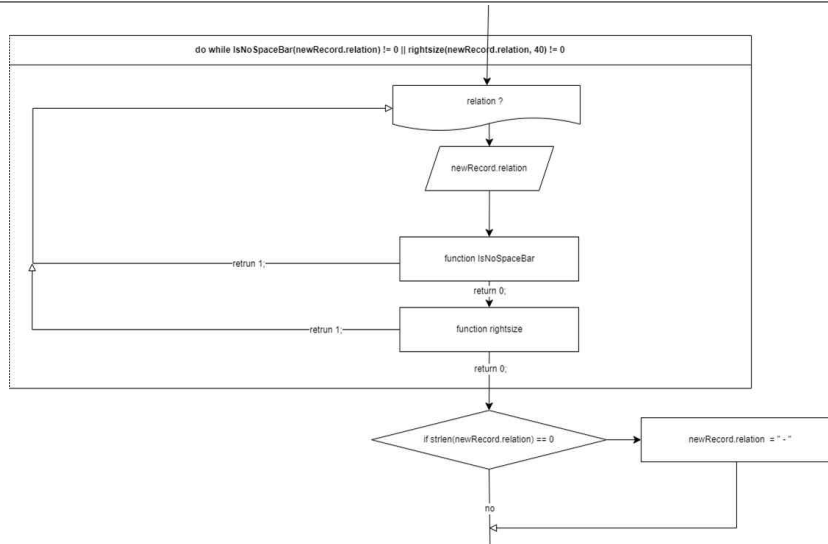
입력하게 한다. IsNum 함수를 통해 문자열에 한글이나 숫자가 아닌 것이 포함되었는지 검사하고 있다면 다시 입력하게 한다.

rightsize 함수를 통해 12자리 이상의 전화번호를 받았는지 검사한다. 입력 자릿수가 초과하면 다시 입력하라고 한다.

전화번호는 사람마다 입력하는 형태가 다를 수 있다. 000-0000-0000라고 입력하는 사람, 000000000000 라고 입력하는 사람 등 다양한 형태가 있다. 이렇게 된다면 저장할 때 통일성이 없으므로 무조건 숫자만 입력받도록 했다.



(3)



```
do {
    strcpy(newRecord.relation, " ");
    printf("%s", "관계? (생략을 원할 시 엔터를 누르세요.)\n"); // 관계는 필수적으로 입력을 받지 않고 엔터를 입력하면 '-' 으로 대체시킬
    scanf("%[^\n]", newRecord.relation);
    getchar();
} while (isNoSpaceBar(newRecord.relation) != 0 || rightsize(newRecord.relation, 40) != 0); // '-' 문자가 존재하거나 크기에 맞지 않게 입력하면 다시 입력하게 함
if (strlen(newRecord.relation) == 0) { // 엔터를 입력했을 때, '-' 으로 대체함
    strcpy(newRecord.relation, "-");
}
```

전화번호 입력이 끝나면 같은 원리로 관계 입력이 진행된다. 관계 항목은 필수적으로 입력을 받지 않고 엔터를 입력하면 공백을 의미하는 '-'으로 바뀌게 했다.

do-while문에 진입하게 되면 먼저 관계? 라는 안내 메시지를 출력하여 공백과 스페이스를 포함한 문자열을 받는다.

그 후, 입력된 문자열을 검사한다.

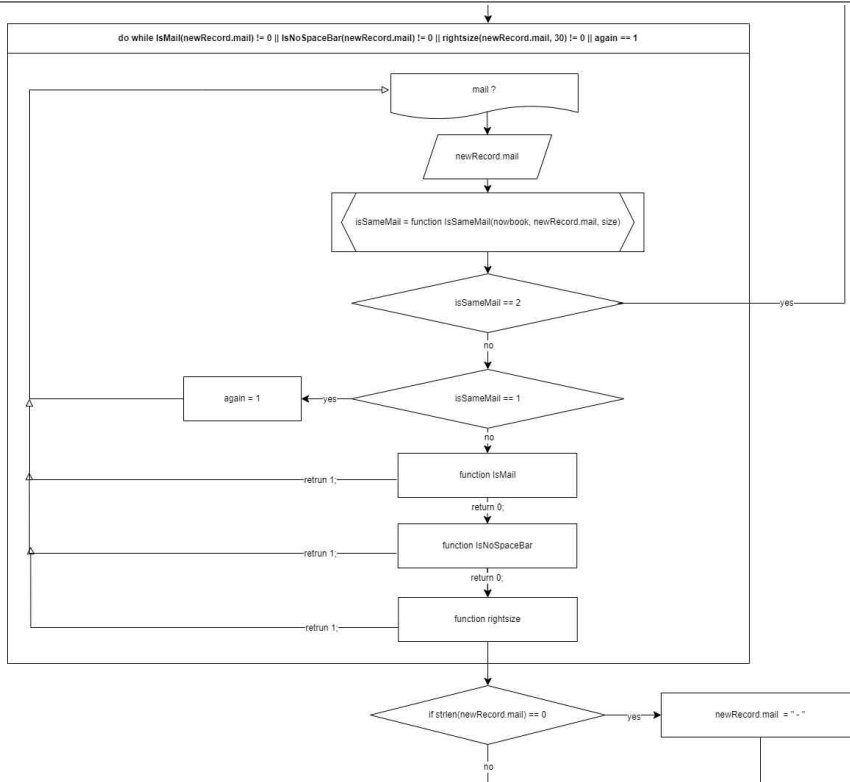
IsNoSpaceBar 함수를 통해 문자열 안에 " " 스페이스바가 포함되어 있는지 검사하고, 포함되었다면 다시 입력받도록 한다. rightsize 함수를 통해 40칸 이상의 문자열이 입력되었는지 검사하고 이상이라면 다시 입력받도록 한다.

위의 조건식을 만족하였다면 newRecord.relation 변수의 문자열 길이를 검사한다.

만약 공백 문자 (엔터) 만 있다면 값은 0이고 아니면 0이 아닌 값을 가진다.

따라서 0일 때, 엔터만 입력되었다면 생략하겠다는 뜻으로 간주하고 '-' 값을 대신하여 넣는다.

(4)



```
// 이메일 추가
do {
    again = 0;
    strcpy(newRecord.mail, "\0");
    printf("%s", "이메일? (생략을 원할 시 엔터를 누르세요.)\n"); // 이메일은 필수적으로 입력을 받지 않고 엔터를 누르면 - 으로 대체시킬
    scanf("%[^\n]s", newRecord.mail);
    getchar();
    int isSameMail = isSameMail(nowBook, newRecord.mail, size); // 목록에 이미 같은 이메일이 있는지 검사하고 리턴값을 저장함
    if (isSameMail == 2) { // 목록에 같은 메일이 있고 새로 입력하지 않는다고 하면 메뉴 화면으로 보냄
        return { 0 };
    }
    else if (isSameMail == 1) { // 목록에 같은 메일이 있고 새로 입력한다고 하면 다시 입력하게 함
        again = 1;
    }
} while (isMail(newRecord.mail) != 0 || isNoSpaceBar(newRecord.mail) != 0 || rightsize(newRecord.mail, 30) != 0 || again == 1);
// 메일의 형식에 맞추어 입력 했는지, 공백은 없는지, 사이즈를 맞게 입력했는지 검사함

if (strlen(newRecord.mail) == 0) { // 만약에 엔터를 누르면 하이픈으로 대체시킬
    strcpy(newRecord.mail, "-");
}
```

관계 추가를 끝내게 되면 이메일 추가 반복문으로 진입한다.

이메일도 관계 추가와 마찬가지로 필수 입력이 아니며 엔터를 누르게 되면 '-' 로 대체하게 했다.

먼저 반복문에 진입하면 메일을 입력하라는 출력문이 뜬다.

메일을 공백을 포함한 문자열로 입력받고 newRecord.mail 변수에 저장된다.

전화번호부에 이미 같은 이메일이 존재하는지 검사하기 위해

isSameMail 변수에 isSameMail 함수의 리턴값을 저장한다. isSameMail 함수는 입력한 이메일과 동일한 이메일이 목록에 존재하고, 다른 이메일을 입력한다고 하면 리턴값이 1이고, again 변수를 1로 바꾸어 while문의 조건식을 통해 다시 입력을 받게 된다. 동일한 이메일이

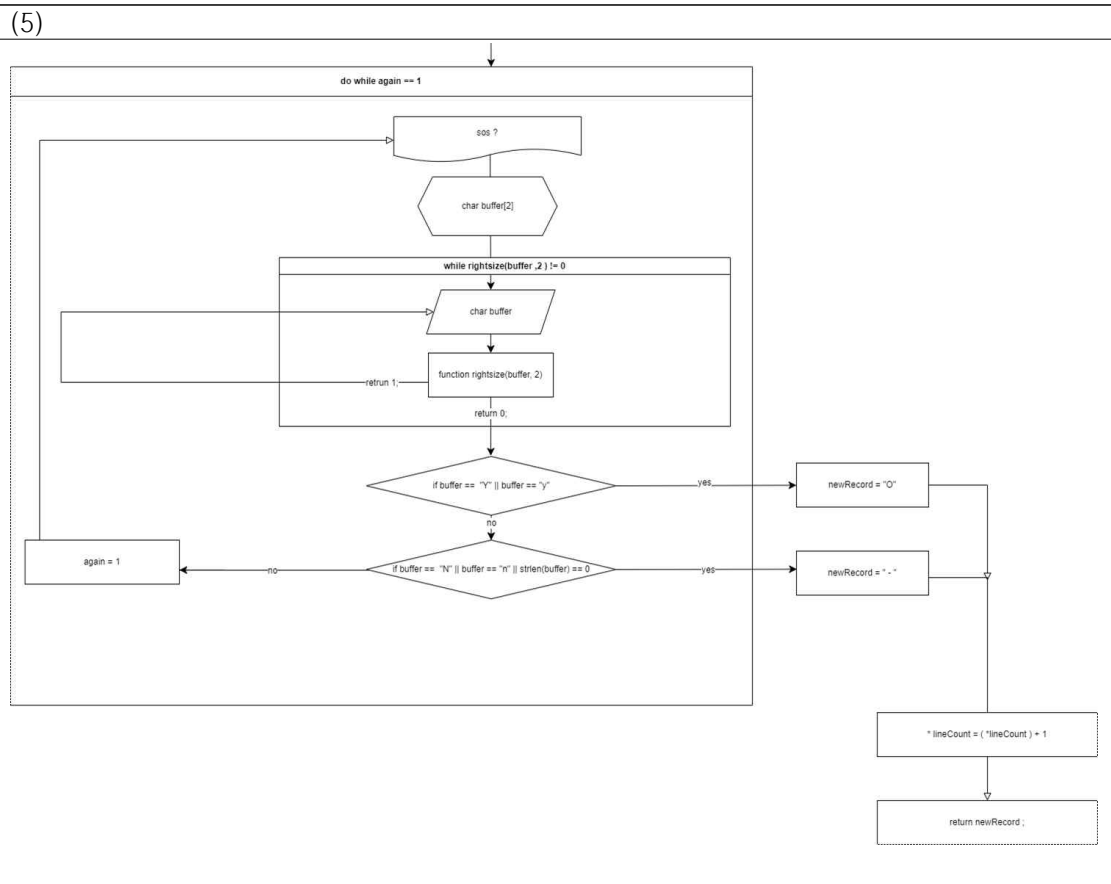
존재하고 다른 이메일을 입력하지 않겠다고 하면 {0}을 리턴하고 추가한 것들이 저장되지 않은 상태로 메뉴 화면으로 돌아간다.

동일한 이메일이 존재하지 않으면 IsSameMail 함수의 리턴값은 0이다.

위 과정을 완료하면, while문의 조건식을 통해 메일에 맞는 형식으로 문자열이 입력되었는지 검사한다. 먼저 IsMail 함수를 통해, 메일에 '@' 와 '.' 기호가 입력되었는지 확인한다. 기호가 입력되지 않았다면 메일 형식이 아닌 거로 간주하고 다시 입력받는다.

그 후 rightsize 함수를 통해 메일이 30자 이상으로 입력되었는지 검사하고, 30자 이상이라면 다시 입력받는다.

위 과정을 모두 완료했다면 문자열의 길이를 검사하여, 문자열의 길이가 0이면 (엔터가 입력되었다면) '-' 문자로 대체한다.



```

// 긴급 연락처 등록 여부
do {
    again = 0;
    strcpy(newRecord.sos, "0");
    printf("%s", "\n긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요) \n[Y/N]\n"); // 긴급 연락처 등록 여부를 물어봄
    char buffer[2] = { 0 }; // 등록 여부를 입력받을 문자열을 지정함
    do {
        strcpy(buffer, "0");
        scanf("%c", buffer);
        getchar();
    } while ( rightsize(buffer, 2) != 0 ); // 너무 많은 문자를 입력하면 다시 입력하라고 안내함

    if (strcmp(buffer, "Y") == 0 || strcmp(buffer, "y") == 0) { // 지정 한다고 하면 긴급 연락처 여부 항목에 0를 입력함
        strcpy(newRecord.sos, "0");
    }
    else if (strcmp(buffer, "N") == 0 || strcmp(buffer, "n") == 0 || strlen(buffer) == 0) { // 지정 안한다고 하면 - 을 입력함
        strcpy(newRecord.sos, "-");
    }
    else {
        printf("... 올바른 값을 입력하세요."); // y 와 n을 제외한 공백 스페이스바 등 다른 값을 입력하면 다시 입력하라고 출력
        again = 1;
    }
} while ( again == 1);
printf("===== \n");
printf("%-20s %-20s %-20s %-20s %-20s\n", "이름", "전화번호", "관계", "메일", "긴급연락처 등록 여부");
printf("===== \n");
printf("%-20s %-20s %-20s %-20s %-20s\n", newRecord.name, newRecord.num, newRecord.relation, newRecord.mail, newRecord.sos);
printf("===== \n");
printf("%s\n", "... 저장 완료!"); // 저장을 다 했으면, 저장을 완료했다고 출력함
*lineCount = (*lineCount) + 1; // 저장을 다 했으면, 파일의 줄 수(입력받은 전화번호의 개수) 를 증가시킴

return newRecord; // 입력받은 구조체를 돌려줌
}

```

이메일까지 완료되었다면 반복문 1의 안에서 긴급연락처 등록 여부를 물어보는 안내를 출력한다.

후에 입력받은 값을 저장할 char buffer[2] = {0}; 배열을 형성한다.

반복문 1의 안, 반복문 2의 조건식에 rightsize 함수를 통해, 입력받은 buffer 의 크기를 검사한다. 크기가 2 이상이면 다시 입력받도록 한다.

반복문 2 이후에 조건식에서는 입력받은 문자가 무엇을 의미하는지 검사한다.

만일 buffer 가 'Y' 또는 'y'면,

긴급연락처로 등록하겠다는 표시로 받아들인다. 따라서 newRecord.sos의 값을 "O" 로 바꾼다.

만일 buffer 가 'N' 또는 'n' 또는 엔터면 등록을 하지 않겠다는 표시로 받아들인다. 따라서 newRecord.sos의 값을 '-'로 바꾼다.

이 이외의 값이 들어온다면 올바른 값을 입력받도록 반복문 1을 다시 진행한다.

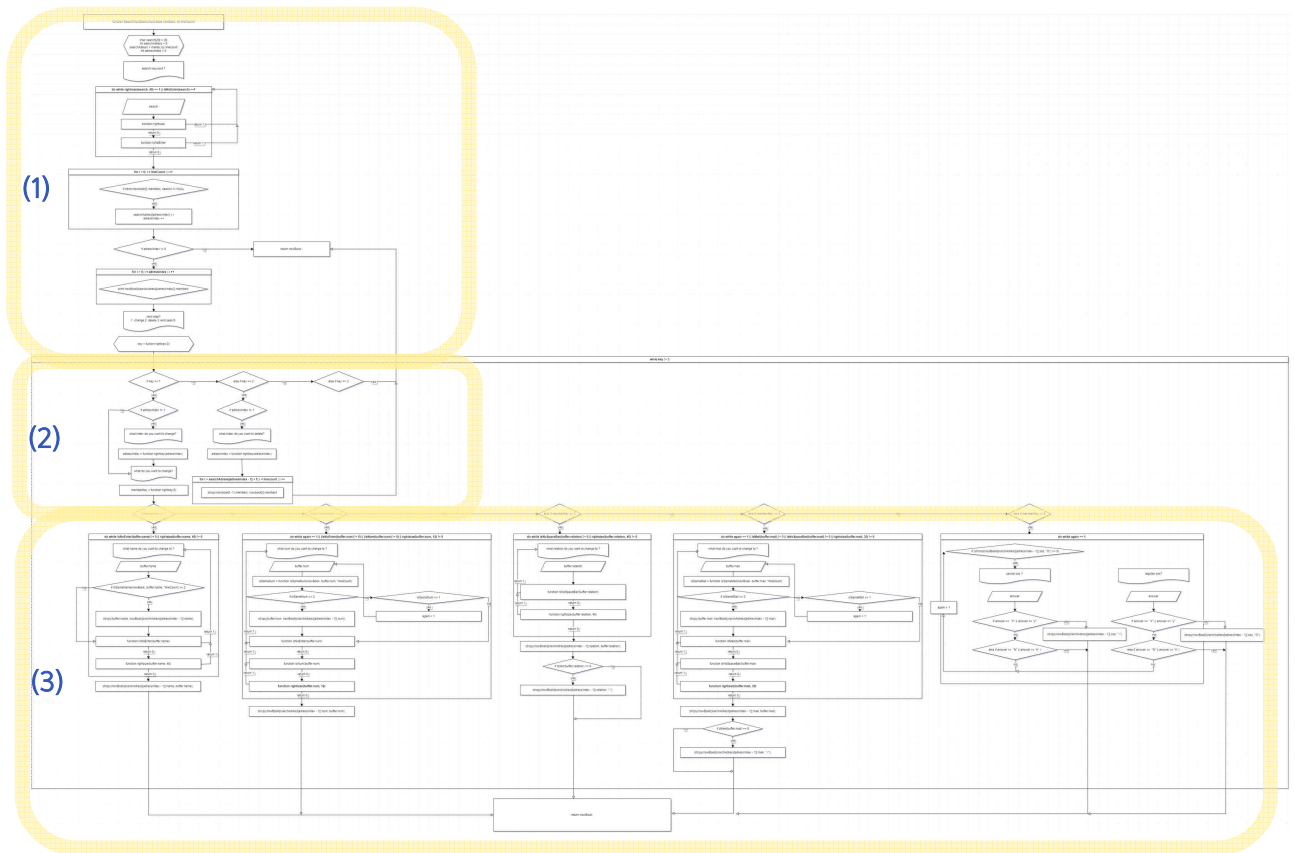
이후 긴급연락처 등록 여부까지 진행이 끝나게 되면, 연락처에 추가할 값을 모두 입력받은 것이다. 입력한 값을 확인할 수 있도록 출력한다.

저장을 완료했다면, 총연락처의 개수를 뜻하는 lineCount 변수를 1 증가시킨다. 함수 내부에서 증가시킨 것이지만 포인터 변수이기 때문에 함수에서 벗어난 후에도 lineCount의 값은 1 증가해 있다.

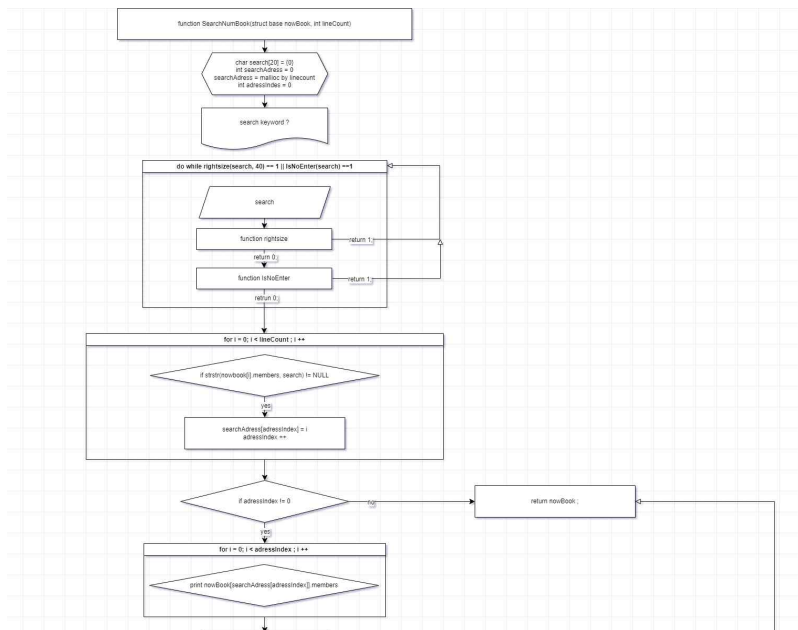
이후 입력한 newRecord를 함수를 호출한 부분에 리턴 하여 기존의 전화번호부에 저장할 수 있도록 한다.

(2) SearchNumBook 함수

SearchNumBook 함수의 전체 순서도는 다음과 같다. 총 (3) 가지 단계로 나눌 수 있다.



(1)



```
#include "novBook.h"

struct base SearchNumBook(struct base novBook, int* lineCount) {
    printf("2s", "===== 02. 검색 과 수정/삭제 ===== \n");

    char search[40] = { 0 };
    int* searchAdress = 0; // 검색어와 일치하는 항목의 주소(배열번호)를 담은 검색어주소배열 설정
    searchAdress = (int*)malloc(sizeof(int) * (*lineCount)); // 검색어와 일치하는 항목은 최대 전체 항목 수와 같으므로 동적메모리 설정
    int adressIndex = 0; // 검색어와 일치하는 항목의 배열번호의 순번을 담은 검색어주소배열순번 설정
    // 검색 시작
    printf("2s\n", "... 검색어?");

    do {
        strcpy(search, "00");
        scanf("2L%0s", search);

        char check = 0;
        do {
            check = getchar();
        } while (check != '\n');

    } while ( ( rightsize(search, 40) == 1) || !isNoEnter(search) == 1); // 검색어가 20자 이내인지, 엔터나 스페이스바는 없는지 검사

    for (int i = 0; i < *lineCount; i++) { // 검색어와 항목의 이름, 전화번호, 관계, 이메일을 부분 문자열 비교를 통해 일부라도 일치한다면 모두 출력함
        if ((strstr(novBook[i].name, search) != NULL) || (strstr(novBook[i].num, search) != NULL) || (strstr(novBook[i].relation, search) != NULL) || (strstr(novBook[i].mail, search) != NULL)) {
            searchAdress[adressIndex] = i; // 검색되는 항목의 배열 번호를 searchAdress 배열에 담음
            adressIndex++; // 한번 찾을 때마다 adressIndex의 번호를 증가시킴 따라서 adressIndex는 검색된 총 항목의 개수를 포함
        }

        printf("... 총 20개의 목록 검색 \n", adressIndex); // 검색어와 일치하는 항목을 전체 출력함

        if (adressIndex != 0) { // 만약 검색된 항목이 1개 이상이라면
            printf("----- \n");
            printf("2-5s 2-20s 2-20s 2-20s 2-20s 2-20s \n", "번호", "이름", "전화번호", "관계", "메일", "이메일연락처 등록 여부");
            printf("----- \n");
            for (int i = 0; i < adressIndex; i++) {
                printf("2-5s 2-20s 2-20s 2-20s 2-20s 2-20s \n", i + 1, novBook[searchAdress[i]].name, novBook[searchAdress[i]].num, novBook[searchAdress[i]].relation, novBook[searchAdress[i]].mail, novBook[searchAdress[i]].sos);
            }

            printf("2s", "----- 다음 작업 ===== \n"); //수정, 삭제, 검색종료에서 원하는 작업을 고를
            printf("01. 수정 02. 삭제 03. 검색 종료\n");
            int key = 0; // 다음 작업의 메뉴 번호를 저장할 변수 선언
            key = rightkey();
        }
    }

    return novBook;
}
```

SearchNumBook 함수가 호출을 받게 되면, 검색어를 입력 받을 search[20] 변수, 검색된 항목의 인덱스 번호를 담은 searchAdress 변수, serachAdress의 인덱스값을 표현할 adressIndex 변수를 선언한다.

검색된 항목이 많을 경우를 대비하여 설정한 searchAdress 변수는, 전화번호에 있는 항목이 모두 검색 될 경우를 대비하여 malloc 동적메모리 함수를 이용하여 lineCount 즉, 전화번호의 총 개수만큼 int 형의 메모리를 할당한다.

이후 검색어를 입력하라는 안내 메시지를 출력한다.

먼저 do-while 반복문1 안의 do-while 반복문 2에서 검색어를 입력받는다. 검색어는 반복문 2의 조건식에서 형식에 맞는지 검사된다.

rightsize 함수를 통해 입력받은 search의 문자열이 40 이상인지 검사한다. 이상이라면, 다시 입력받는다. 이후 IsNoEnter 함수를 이용해 엔터나 스페이스바가 입력되었는지 검사하고, 입력

되었다면 다시 입력받는다.

반복문 2에서 제대로 된 값을 search에 입력받으면, for문을 사용하여 인덱스 번호를 0부터 전화번호의 총 개수인 lineCount 전까지 점점 증가시키는 반복문3을 통해 검색어와 일치하는 항목이 있는지 검사한다.

일치하는 항목을 검사할 때는 strstr 함수를 이용했다. strstr(문자열1, 문자열2) 함수는 문자열 2가 문자열 1에 포함되어있을 때, 주소값을 반환해주는 함수로, NULL 이면 반환된 값이 없다는 의미로 쓰여졌다. 따라서 nowBook의 모든 멤버 (긴급연락처등록여부 제외)와 입력받은 search 문자열을 비교하여, search 문자열이 nowBook에 일부 포함되어있다면 그 인덱스 번호를 searchAdress[adressIndex] 에 담는다. 이때, 인덱스 번호를 담은 후, adressIndex ++ 연산자를 통해 adressIndex가 검색된 전화번호의 총 개수를 의미하도록 했다.

검색이 완료가 되었다면,

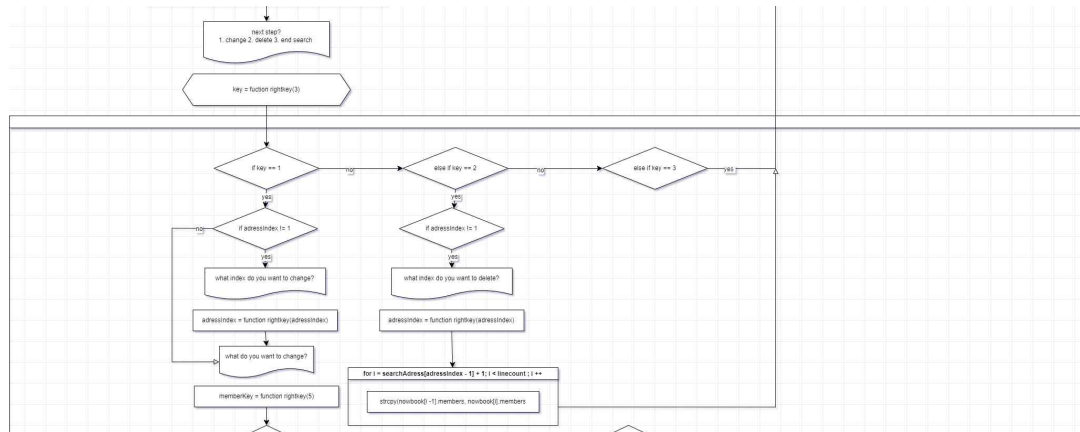
만일 검색된 항목이 1개 이상이라면 검색된 모든 항목을 for문 반복문 2를 통해 출력하도록 했다. 검색된 항목은 searchAdress[adressIndex] 를 nowBook 배열의 숫자로 이용하면 간편하게 출력할 수 있다.

검색된 항목이 없다면 nowBook을 리턴 해 검색 프로그램을 종료한다.

이후 다음 작업인 1. 수정, 2. 삭제, 3. 검색 종료 를 출력하고

메뉴에 맞는 값을 key 값으로 rightkey(3) 함수를 이용하여 1부터 3까지의 값만 입력받도록 했다.

(2)



```
// 삭제 시작
else if (key == 2) {
    if (adressIndex != 1) {
        printf("%s\n", "... 삭제를 원하는 항목의 번호?"); //검색된 항목이 2개 이상이면 삭제할 항목의 번호를 입력받음
        adressIndex = rightkey(adressIndex);
    }

    for (int i = searchAdress[adressIndex - 1] + 1; i <= lineCount; i++) { // 항목의 이전은 유지하고, 항목의 이후를 한칸씩 당겨서 선택한 항목을 없애지게 만들
        strcpy(nowBook[i - 1].name, nowBook[i].name);
        strcpy(nowBook[i - 1].num, nowBook[i].num);
        strcpy(nowBook[i - 1].relation, nowBook[i].relation);
        strcpy(nowBook[i - 1].mail, nowBook[i].mail);
        strcpy(nowBook[i - 1].sos, nowBook[i].sos);

        *lineCount = (*lineCount) - 1; // 삭제를 완료하면 줄수를 한칸 줄임
    }

    printf("%s\n", "... 삭제 완료! \n");
    free(searchAdress); // 동적메모리 해제
    searchAdress = NULL;
    return nowBook;
}

else if (key == 3) { // 검색 종료 항목을 누르면 메뉴 화면으로 돌아가게 함
    printf("... 검색 종료\n");
    free(searchAdress); // 동적메모리 해제
    searchAdress = NULL;
    return nowBook;
}
```

key 값에서 2. 삭제, 3. 검색 종료를 선택 하였을 때를 먼저 다뤄보려고 한다.

먼저 key 값이 2. 삭제일 때 어떻게 진행되는지 살펴보자.

adressIndex가 검색된 항목의 총 개수를 의미한다는 것은 전 단계에서 설명 하였다.

만약에, 검색된 항목이 1개가 아니라면, 검색된 항목 중에서 삭제를 할 항목을 선택해야 한다.

따라서 rightkey 함수를 이용하여 검색된 항목의 번호 내에서 1개를 선택하도록 하였다.

선택한 번호를 adressIndex에 담았다.

삭제할 번호를 선택 하였다면, 삭제를 해야한다.

나는 삭제를, 삭제할 인덱스의 이전 nowBook은 건들지 말고, 이후의 nowBook을 한칸씩 당겨 마지막 인덱스 번호를 지우는 방식을 통해 마치 지워진 것처럼 보이게 만들었다.

한 칸씩 당기는 방식은 반복문을 이용했다. adressIndex가 뜻하는 것은 내가 고른 항목의 번호이다. 배열의 번호는 0부터 시작하지만 항목의 번호는 1부터 시작하기 때문에, 사실 내가 고른 항목의 번호는 실제 인덱스 번호와 1 차이가 난다. 따라서 searchAdress[adressIndex - 1] 은 내가 고른 항목의 nowBook 배열의 번호이다. 이제 이 번호 이후부터 1칸씩 당겨야 하므로, nowBook 배열의 번호는 searchAdress[adressIndex - 1] 부터, 전체 항목의 개수인

lineCount 전 까지이다.

이 배열의 번호 동안 nowBook[i]의 모든 멤버의 값을 한 칸 앞인 nowBook[i-1]로 복사했다. 삭제가 완료 되었다면 전화번호부의 총 항목 수는 1개 없어졌기 때문에 lineCount를 1개 줄였다. 비록 함수 내부에서 이루어졌지만 lineCount를 포인터로 받았기 때문에 호출한 함수에서도 lineCount는 1칸 없어진다.

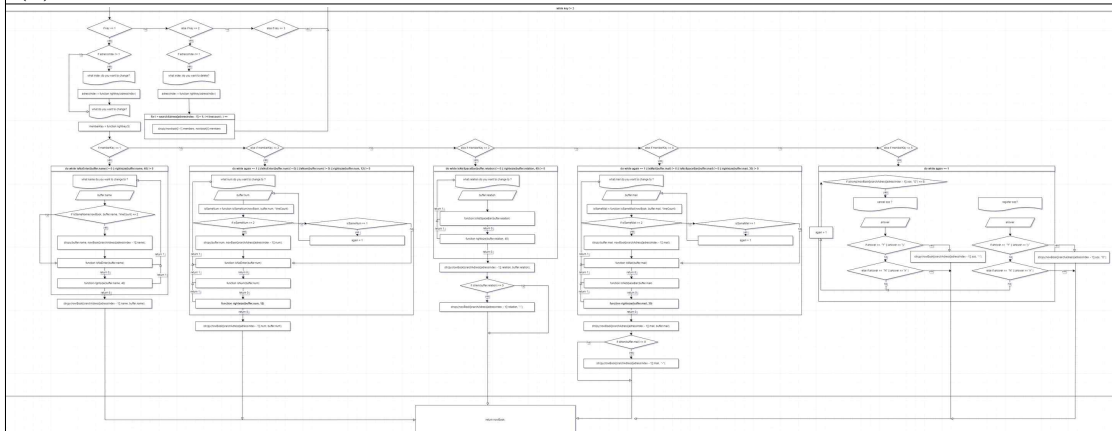
삭제가 완료 되었다면, 삭제가 되었다고 출력하여 혼동을 방지한다.

이후 SearchNumBook 함수를 호출한 곳에 삭제가 완료된 nowBook을 리턴 한다.

다음은 key값이 3.검색 종료를 의미할 때를 알아보자.

검색을 종료하면, 종료했다고 알려주고 nowBook을 리턴 했다.

(3)



```
if (key == 0) {
    struct book buffer = {0;};
} else if (key == 1) {
    if (addressIndex != 1) {
        printf("수정 할 내용? 현재 내용 : ", nowBook[srarchAddress[addressIndex - 1]].name);
        addressIndex = rightKey(addressIndex); // 수정을 원하는 번호를 입력받음
    }
    printf("%s %20s %20s %20s %20s %20s", "번호", "이름", "연희번호", "관계", "필수", "인원번호의 등록 여부");
    printf("%d %20s %20s %20s %20s %20s", 1, nowBook[srarchAddress[addressIndex - 1]].num, nowBook[srarchAddress[addressIndex - 1]].name, nowBook[srarchAddress[addressIndex - 1]].relation, nowBook[srarchAddress[addressIndex - 1]].sex);
    printf("수정 할 내용");
    memberKey = 0;
} else if (memberKey == 0) {
    memberKey = rightKey(5); // 무슨 필드를 수정할지 입력받는 변수
} else if (memberKey == 1) { // 값을 수정하는 것은 입력과 같은
    do {
        printf("수정 할 내용? 현재 내용 : ", nowBook[srarchAddress[addressIndex - 1]].name);
        scanf("%s", buffer.name);
        if (isSameNum(nowBook, buffer.name, *lineCount) == 2) { // 수정할 내용이 이미 연희번호 존재하면 다시 입력시키거나 메뉴로 돌려보냄
            strcpy(buffer.name, nowBook[srarchAddress[addressIndex - 1]].name);
        } while (again == 1 || (IsNoEnter(buffer.name) != 0) || (IsNum(buffer.name) != 0) || rightsize(buffer.name, 40) != 0);
        strcpy(nowBook[srarchAddress[addressIndex - 1]].name, buffer.name);
    }
}
```

```
else if (memberKey == 2) {
    int again = 0;
    do {
        again = 0;
        printf("%s %s\n", "... 수정 할 내용? 현재 내용 : ", nowBook[srarchAddress[addressIndex - 1]].num);
        scanf("%i", &buffer.num);
        getchar();
        int isSameNum = isSameNum(nowBook, buffer.num, *lineCount);
        if (isSameNum == 2) {
            strcpy(buffer.num, nowBook[srarchAddress[addressIndex - 1]].num);
        } else if (isSameNum == 1) {
            again = isSameNum;
        } while (again == 1 || (IsNoEnter(buffer.num) != 0) || (IsNum(buffer.num) != 0) || rightsize(buffer.num, 12) != 0);
        strcpy(nowBook[srarchAddress[addressIndex - 1]].num, buffer.num);
    }
} else if (memberKey == 3) {
    do {
        printf("%s %s\n", "... 수정 할 내용? 현재 내용 : ", nowBook[srarchAddress[addressIndex - 1]].relation);
        scanf("%s", &buffer.relation);
        getchar();
    } while (IsNoSpaceBar(buffer.relation) != 0 || rightsize(buffer.relation, 40) != 0);
    strcpy(nowBook[srarchAddress[addressIndex - 1]].relation, buffer.relation);
    if (strlen(buffer.relation) == 0) {
        strcpy(nowBook[srarchAddress[addressIndex - 1]].relation, "-");
    }
}
```


key 값이 1. 수정 일때를 다뤄보고자 한다.

flowchart 와 코드 구성은 매우 길지만 돌아가는 모양만 알면, 이해가 가능할것이라 생각한다. 먼저 수정을 하게 된다면, 삭제와 마찬가지로 항목의 수인 adressIndex 가 2 이상이라면 (1이 아니라면) 수정을 원하는 항목의 번호를 골라야 한다. 항목의 번호는 rightkey 함수를 이용하여 adressIndex 에 답았다.

현재 고를 수 있는 수정 가능한 멤버는

1. 이름 2. 연락처 3. 관계 4. 이메일 5. 긴급 연락처 등록 여부 이다.

수정 할 멤버의 번호를 rightkey 함수를 이용하여 memberKey 에 입력받는다.

먼저 memberKey가 1. 이름 수정일 경우이다.

do-while 조건문 안에서

무슨 이름으로 바꾸고 싶은지 출력하고, 바꿀 이름을 buffer.name에 입력받는다.

IsSameName 함수를 통해 바꿀 이름이 이미 연락처에 있는지 검사한다.

만일 이미 연락처에 존재한다면, 존재해도 저장을 할것인지 아니면 저장하지 않고 종료할 것인지 물어본다.

종료 할 것이라면, 기존의 이름인 nowBook[searchAdress[adressIndex -1]].name을 buffer.name 에 덮어씌운다. 작성한 이름을 지워버리는 것이다.

이후 while문의 조건식에서 이름이 제대로 된 형식인지 검사한다.

IsNoEnter 함수를 통해 공백이나 스페이스바가 있는지 검사한다. 있다면 바꾸고 싶은 이름을 다시 입력받게 한다.

rightsize를 통해 이름의 크기가 40자 이상인지 검사한다. 이상이라면 다시 입력받게 한다.

이렇게 완료가 되었다면, 수정된 이름인 buffer.name을

nowBook[searchAdress[adressIndex -1]].name에 덮어씌우고

수정이 완료된 nowBook을 리턴 한다.

두 번째로 memberKey가 2. 연락처 수정일 경우이다.

반복문 내부에서 무슨 전화번호로 바꾸고 싶은지 출력하고, 바꿀 전화번호를 buffer.num에 입력받는다.

IsSameNum 함수를 통해 바꿀 전화번호가 이미 연락처에 있는지 검사한다. 리턴 값을 isSameNum 변수에 넣는다.

만일 이미 연락처에 존재하고, 다른 전화번호를 입력하겠다고 하면 isSameNum은 1이 된다.

isSameNum가 1이면 반복문 처음으로 돌아가 다시 바꿀 전화번호를 입력한다.

만일 이미 연락처에 존재하고, 다른 전화번호를 입력 하지 않겠다고 하면 isSameNum은 2가 된다.

isSameNum가 2이면 전화번호 수정을 멈추고 기존의 연락처인

nowBook[searchAdress[adressIndex -1]].num을 buffer.num에 복사한다. 역시 입력한 전화번호를 지워버리는 것이다.

이후 while문의 조건식에서 입력받은 연락처가 제대로 된 연락처 형식인지 확인한다.

IsNoEnter 함수를 통해 엔터나 스페이스바가 입력 되었는지 검사하고, 입력 되어있다면 다시 수정할 전화번호를 입력받도록 한다.

IsNum 함수를 통해 입력받은 전화번호가 한글이 아닌 숫자로만 이루어졌는지 확인한다. 만일 숫자가 아닌 값이 있다면 다시 수정할 전화번호를 입력받도록 한다.

rightsize 함수를 통해 12자리 이상 입력했는지 확인한다. 만일 12자리 이상으로 입력했다면 다시 수정할 전화번호를 입력받도록 한다.

이렇게 확인이 완료가 되었다면, 입력받은 buffer.num을

nowBook[searchAdress[adressIndex -1]].num 로 복사한다. 그리고 수정된 전화번호부인



numbook을 리턴한다.

세 번째로 memberKey가 3. 관계 수정일 경우이다.

반복문 내부에서 무슨 관계로 수정하고 싶은지 출력하고, 바꿀 관계를 buffer.relation 에 입력 받는다. while문의 조건식에서 IsNoSpaceBar 함수를 통해 관계 안에 스페이스바가 있는지 검사한다. 있다면 다시 수정하고 싶은 내용을 입력받는다.

rightsize 함수를 통해 관계가 40자 이상으로 작성되었는지 확인한다. 40자 이상이라면 다시 입력받도록 한다.

이후 buffer.relation 안에 있는 내용을 nowBook[searchAdress[adressIndex -1]].relation 으로 복사한다.

nowBook[searchAdress[adressIndex -1]].relation의 문자열 길이가 0(엔터)인지 확인하고, 0이면 '-' 로 바꾼다.

수정된 nowbook을 리턴 한다.

네 번째로 memberKey가 4. 이메일 수정일 경우이다.

반복문 내부에서 무슨 이메일로 수정하고 싶은지 출력하고, 바꿀 이메일을 buffer.mail 에 입력 받는다.

IsSameMail 함수를 통해 연락처에 이미 동일한 이메일이 존재하는지 확인하고 리턴값을 isSameMail 변수에 넣는다.

만일 이미 동일한 이메일이 존재하고, 다른 이메일을 입력하겠다고 하면 isSameMail은 1이 된다.

isSameMail가 1이면 반복문 처음으로 돌아가 다시 바꿀 이메일을 입력한다.

만일 이미 동일한 이메일이 존재하고, 다른 이메일을 입력 하지 않겠다고 하면 isSameMail은 2가 된다.

isSameMail 가 2이면 이메일 수정을 멈추고 기존의 이메일인

nowBook[searchAdress[adressIndex -1]].mail을 buffer.mail에 복사한다. 역시 입력된 이메일을 지우는 과정이다.

이후 반복문의 조건식 내부를 통해 이메일이 형식에 잘 맞춰 있는지 검사한다.

IsMail 함수를 통해 입력받은 메일에 '@'와 '.'이 있는지 검사한다. 없다면 다시 입력받는다.

IsNoSpaceBar 함수를 통해 입력받은 메일에 스페이스가 있는지 검사하고, 있다면 다시 입력 받는다.

rightsize 함수를 통해 입력받은 메일의 글자수가 30 이상인지 검사하고, 이상이라면 다시 입력 받는다.

이후 nowBook[searchAdress[adressIndex -1]].mail 에 buffer.mail을 복사하고

만일 nowBook[searchAdress[adressIndex -1]].mail의 문자열 길이가 0이라면(엔터), '-' 값으로 바꾸어 저장한다.

이후 수정된 연락처인 nowBook을 리턴 한다.



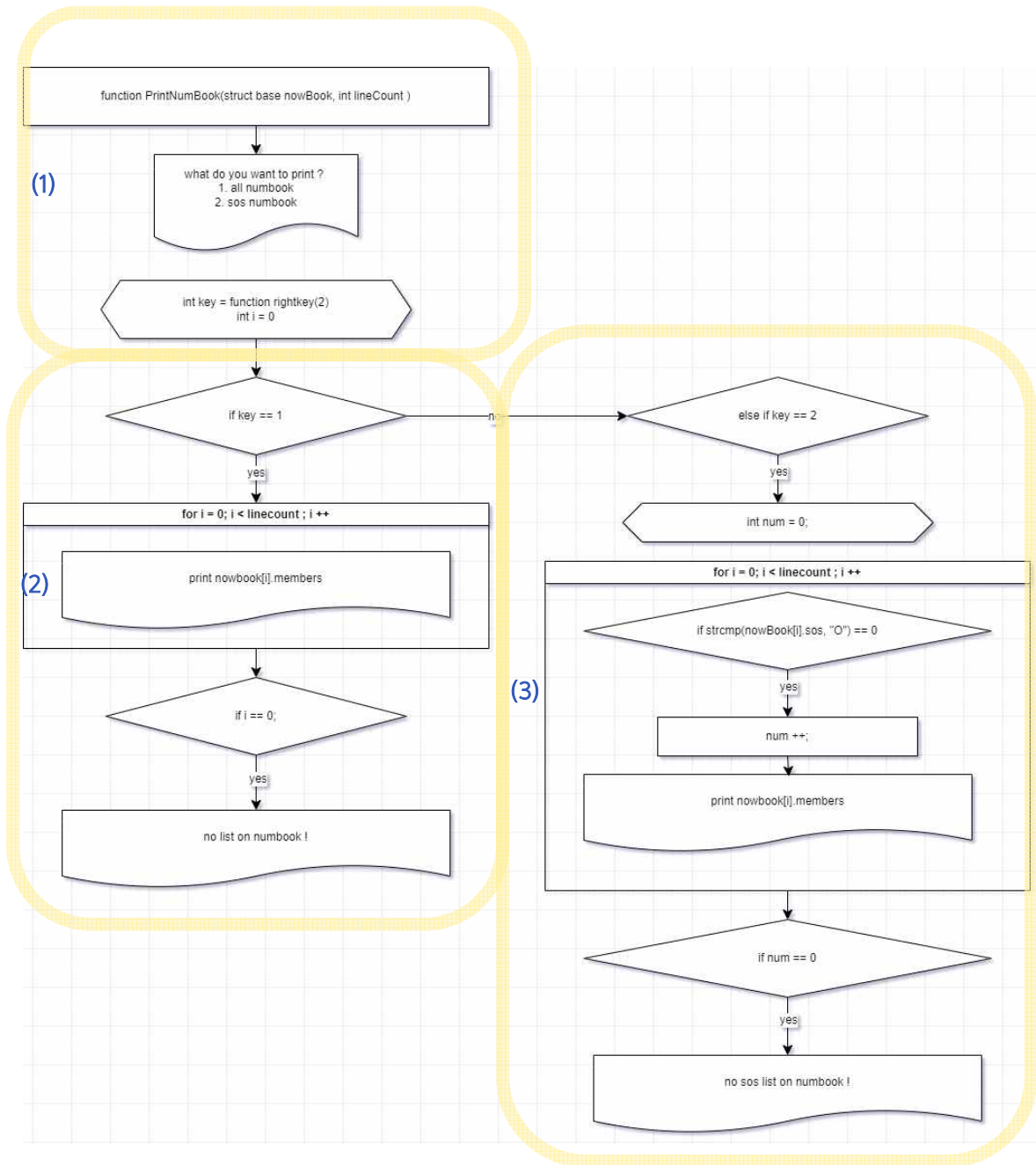
마지막으로 memberKey가 5. 긴급연락처 지정 여부 수정일 경우이다.
반복문 안에서,
만약 nowBook[searchAdress[adressIndex -1]].sos 가 “O” 의 값을 가진다면
긴급연락처 지정을 취소 할건지 물어보고, answer 변수에 입력 받는다.
answer가 ‘Y’또는 ‘y’라면, nowBook[searchAdress[adressIndex -1]].sos의 값을 ‘-’로 바꾸어 저장한다.
만약 answer가 ‘N’또는 ‘n’라면, 아무것도 하지 않는다.
answer가 그 이외의 것이라면 다시 answer 을 입력받는다.

만약 nowBook[searchAdress[adressIndex -1]].sos 가 “O” 의 값을 가지지 않는다면
긴급연락처로 지정 할건지 물어보고, answer 변수에 입력 받는다.
answer가 ‘Y’또는 ‘y’라면, nowBook[searchAdress[adressIndex -1]].sos의 값을 ‘O’로 바꾸어 저장한다.
만약 answer가 ‘N’또는 ‘n’라면, 아무것도 하지 않는다.
answer가 그 이외의 것이라면 다시 answer을 입력받는다.

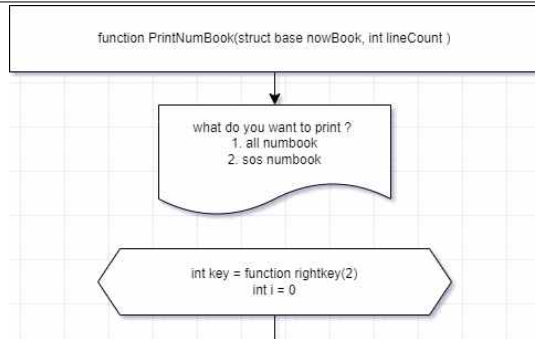
이후 수정된 nowBook을 리턴한다.

(3) PrintNumBook 함수

PrintNumBook 함수의 전체 순서도는 다음과 같다. 총 (3) 가지 단계로 나눌 수 있다.



(1)



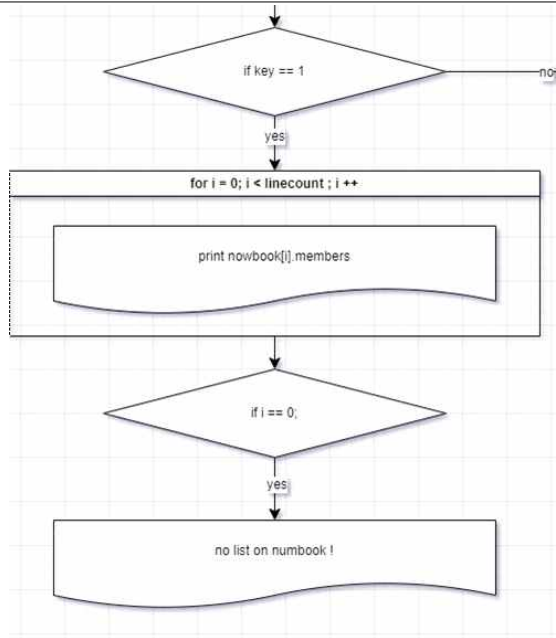
```
void PrintNumBook(struct base* nowBook, int lineCount) { // 전화번호부의 목록 출력
    printf("%s", "===== 03. 출력 ===== \n");
    printf("... 무엇을 출력 하시겠습니까? \n01. 전체 전화번호부 출력\n02. 긴급 연락처 출력\n");
    int key = rightkey(2); // 잘 입력했는지 검사하는 함수
```

`PrintNumBook` 함수에 들어가게 되면

먼저 무슨 전화번호부를 출력 할 것인지 물어본다.

그리고 출력할 목록의 번호를 `rightkey` 함수를 이용하여 `key`에 입력 받는다.

(2)



```
if (key == 1) { // 전체 전화번호부 출력을 입력했을때
    int i = 0;
    printf("===== \n");
    printf("%-5s %-20s %-20s %-20s %-20s %-20s\n", "번호", "이름", "전화번호", "관계", "메일", "긴급연락처 등록 여부");
    printf("===== \n");
    for (i = 0; i < lineCount; i++) { // 반복문을 이용하여 줄 수 만큼 전화번호부를 출력함
        printf("%-5d %-20s %-20s %-20s %-20s %-20s\n", i + 1, nowBook[i].name, nowBook[i].num, nowBook[i].relation, nowBook[i].mail, nowBook[i].sos);
    }
    printf("===== \n");
    printf("%s %d %s", "... 총 ", lineCount, " 항목 출력\n");
    if (i == 0) { // 만약에 출력된 전화번호부가 없으면, 연락처가 없다고 출력함
        printf("\n... 전화번호부에 연락처가 없습니다. \n");
    }
}
```

만일 key가 1 이라면,

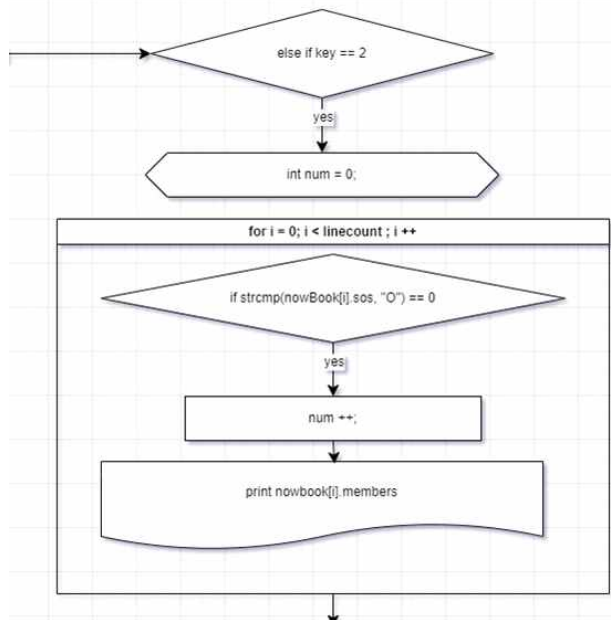
반복문을 이용하여, 인덱스 번호를 0부터 연락처의 총 개수인 lineCount 전 까지 증가시켜 출력한다.

출력을 완료한 후 lineCount 의 값 자체를 출력시켜, 몇 개의 항목이 출력되었는지 알린다.

만약 아무것도 출력되지 않았을 경우 (인덱스 번호가 0 그대로인 경우)

연락처가 없다는 안내 메시지를 출력한다.

(3)

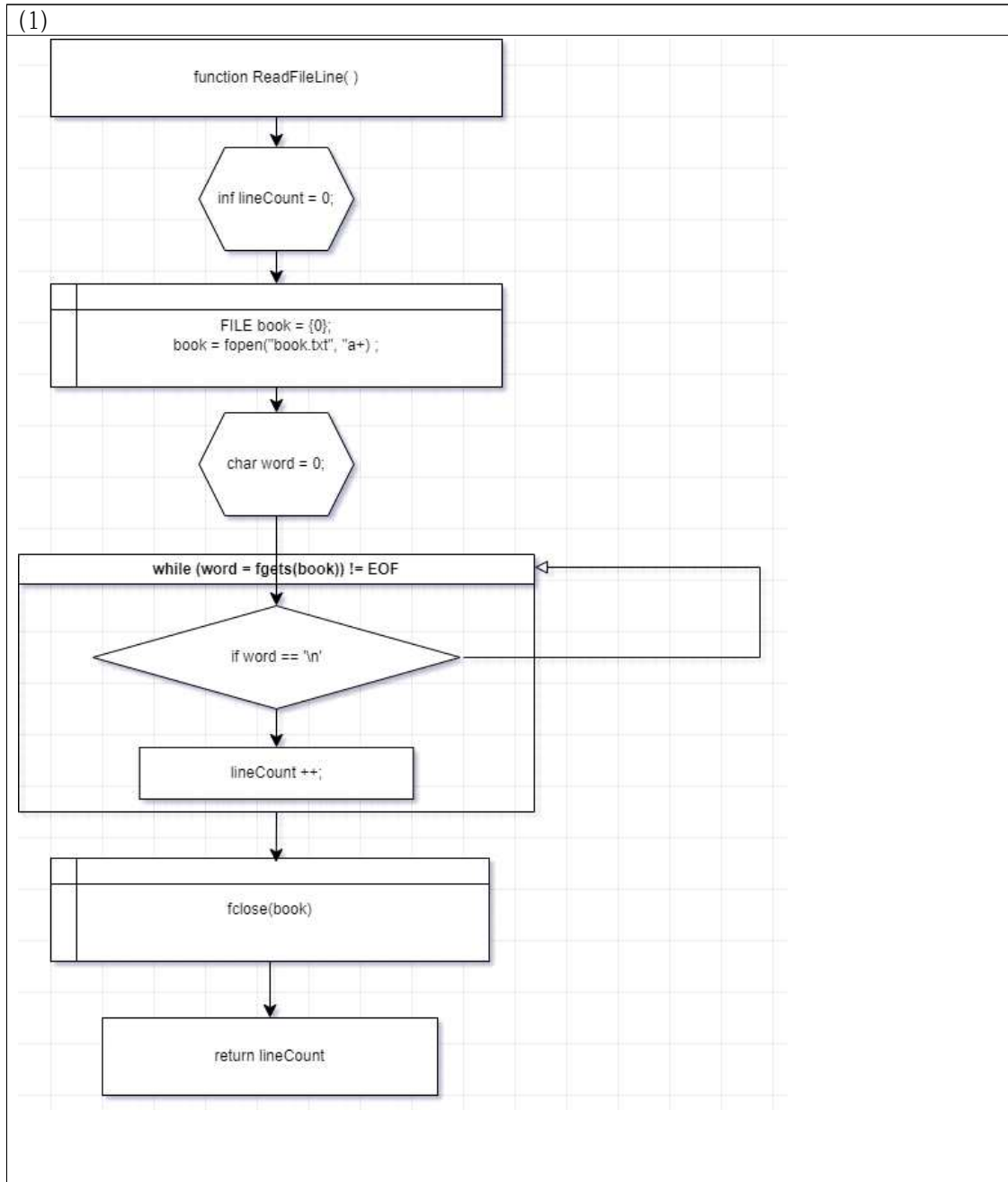


```
else if (key == 2) { // 긴급 연락처 출력을 입력했을 때
    int num = 0;
    printf("=====\\n");
    printf("%-5s %-20s %-20s %-20s %-20s\\n", "번호", "이름", "전화번호", "관계", "메일", "긴급연락처 등록 여부");
    printf("=====\\n");
    for (int i = 0; i < lineCount; i++) {
        if (strcmp(nowBook[i].sos, "O") == 0) { // 긴급 연락처에 'O' 표시가 있는 연락처만 출력함
            num++;
            printf("%-5d %-20s %-20s %-20s %-20s\\n", num, nowBook[i].name, nowBook[i].num, nowBook[i].relation, nowBook[i].mail, nowBook[i].sos);
        }
    }
    printf("=====\\n");
    printf("%s %d %s", "... 총 ", num, " 항목 출력\\n");
    if (num == 0) { // 만약에 연락처가 없으면 연락처가 없다고 표시함
        printf("\\n... 전화번호부에 저장된 긴급 연락처가 없습니다. \\n");
    }
}
```

만일 key가 2 이라면,
반복문을 이용하여, 인덱스 번호를 0부터 연락처의 총 개수인 lineCount 전 까지 증가시켜
조건문에 strcmp를 이용하여, numbook[i].sos의 값이 “O” 인 항목만 출력한다.
이때 출력된 항목의 개수를 나타내는 num 변수를 지정하여
항목이 1개 출력될 때마다 num ++ 의 값을 늘린다.
출력을 완료한 후 num 의 값 자체를 출력시켜, 몇 개의 항목이 출력되었는지 알린다.
만약 아무것도 출력되지 않았을 경우 (num이 0 그대로인 경우)
저장된 긴급 연락처가 없다는 안내 메시지를 출력한다.

(4) ReadFileLine 함수

ReadFileLine 함수의 순서도는 다음과 같다.



```
#include "numbook.h"

int ReadFileLine() {
    int lineCount = 0;
    FILE* book = { 0 };
    book = fopen("book.txt", "a+"); //전화번호부 파일 열기 없다면 만들기

    char word = 0; // txt파일을 문자 1개별로 검사하는 변수

    // txt 파일의 줄 수 count
    while ((word = fgetc(book)) != EOF) // 파일의 끝에 올 때까지 한글자씩 입력받아 검사
    {
        if (word == '\n') { // 줄바꿈 문자가 있다면 lineCount 를 증가시킴
            lineCount++;
        }
    }
    fclose(book); // 전화번호부 파일 닫기

    return lineCount; // 줄 수를 리턴함
}
```

ReadFileLine은, nowBook 구조체 배열에 동적 메모리를 할당하기 위해 선행된다.

총 연락처의 개수가 몇 개인지 파악하는 함수이다.

이후의 WriteOnFile 함수의 내용을 보면 알겠지만, 파일에 연락처를 저장할 때 1 항목에 1 줄로 저장을 했다.

따라서 파일 내부의 엔터 문자('\n')를 카운트 한다면 그것이 연락처의 총 개수가 되는 것이다. 먼저 파일을 fopen 함수를 사용하여 열어준다.

다음으로, 파일 내부를 한 글자씩 검사할 word 변수를 지정한다.

fgetc 함수는 파일을 한 글자씩 읽어준다. 만일, 파일에 끝에 다다르게 된다면 EOF 가 저장되게 되므로, EOF가 저장되기 전까지 반복문을 사용하여 파일에서 한 글자씩 읽는다.

만일 word 가 '\n' 이라면, 엔터키를 입력 받은 것으로 lineCount (전화번호의 총 개수)를 ++ 연산자를 이용해 1개씩 늘린다.

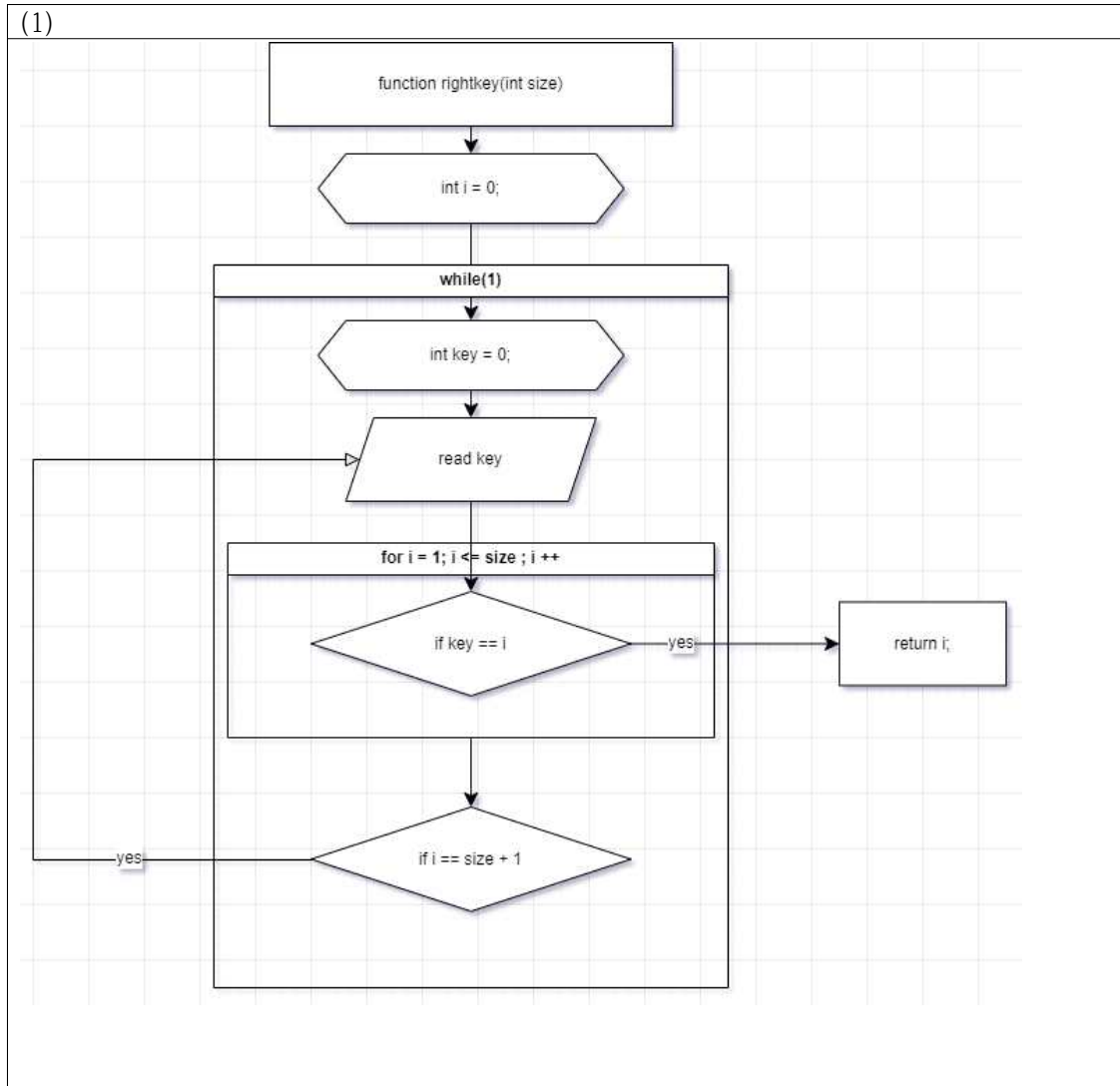
줄 수를 다 확인했다면, 파일을 닫고 줄 수를 리턴한다.

WriteOnFile의 순서도는 다음과 같다.



4) 서브 함수 및 예외처리 함수 진행 과정

(1) rightkey 함수



```

#include "numbook.h"

int rightkey(int size) {
    int i = 0;
    while (1) {
        char check = 0;
        int key = 0;

        scanf("%d", &key); // 정수형으로 한글자를 입력받음
        while (check != '\n') {
            check = getchar();
        }

        for (i = 1; i <= size; i++) {
            if (key == i) {
                return i;
                break;
            }
        }

        if (i == size + 1) {
            printf("... 메뉴와 맞는 숫자를 입력하세요.\n");
        }
    }
}

```

프로그램을 진행하다 보면, 많은 메뉴를 마주치고 그때마다 메뉴의 번호를 입력 받는 상황이 발생한다.

많은 메뉴가 발생하다 보니 항상 맞는 번호를 작성 하였는지 예외 처리를 해줘야 하는 것이 불편했다.

따라서 따로 함수를 만들었다.

size는 메뉴의 마지막 번호를 함수 호출에서 받는다.

먼저 무한루프 반복문에서 진행한다. 맞는 키를 입력하기 전까지는 이 함수를 나갈 수 없다.

반복문 내부에서 정수형의 key 변수를 선언하고 한 글자를 입력 받는다.

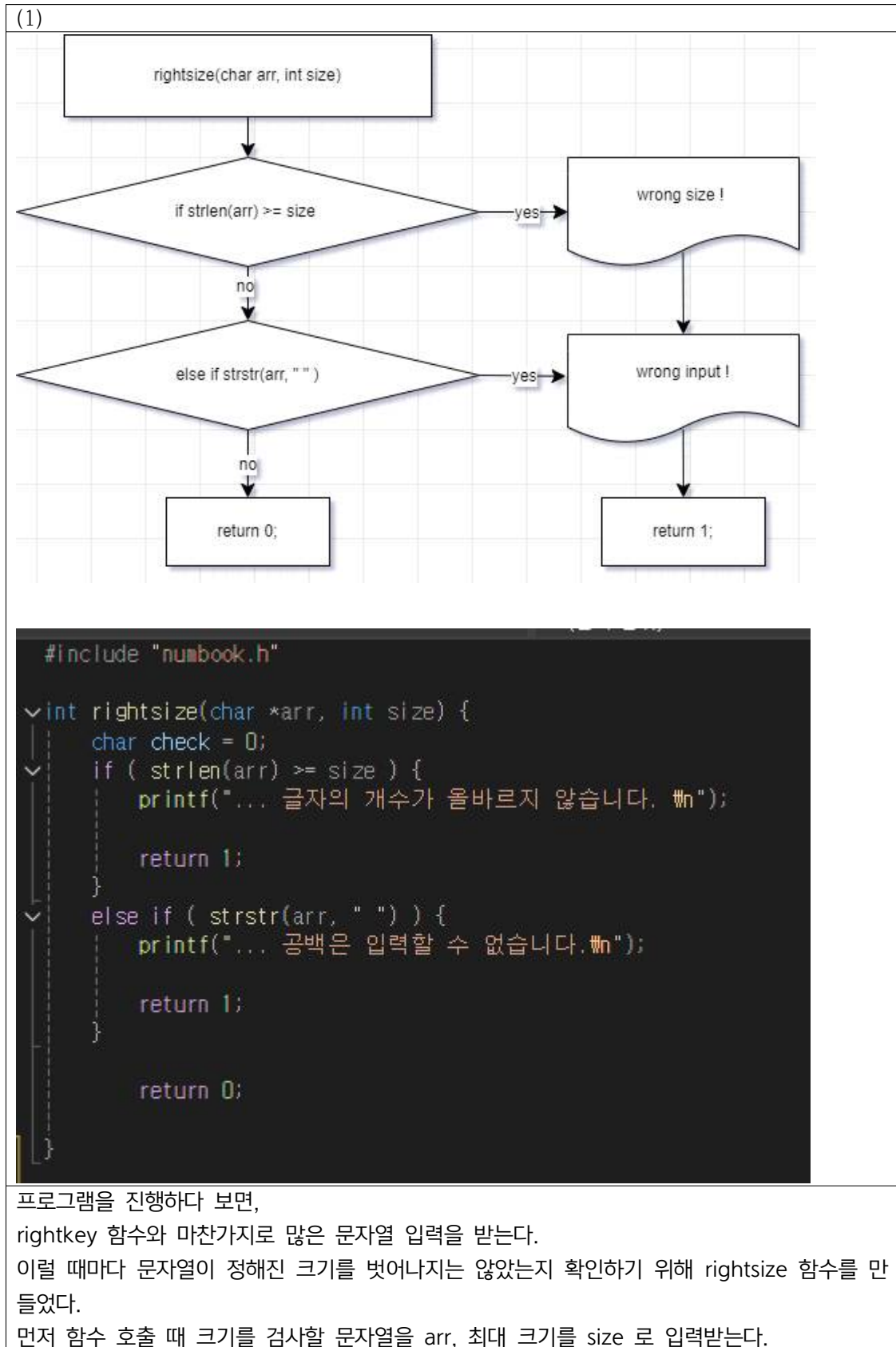
for 반복문 내부에서, i를 1부터 size 와 같거나 작을 때까지 증가시켰을 때,

i와 입력받은 정수형 변수 key 가 일치하는 순간, i 의 값을 리턴 값으로 보낸다. 이때 함수에서 탈출이 가능하다.

만일 for 반복문을 다 완료 하였는데도 리턴 되지 않고 i의 값이 size + 1이 되면,

다시 입력하라는 메시지를 출력하고 다시 입력받는다.

(2) rightsize 함수



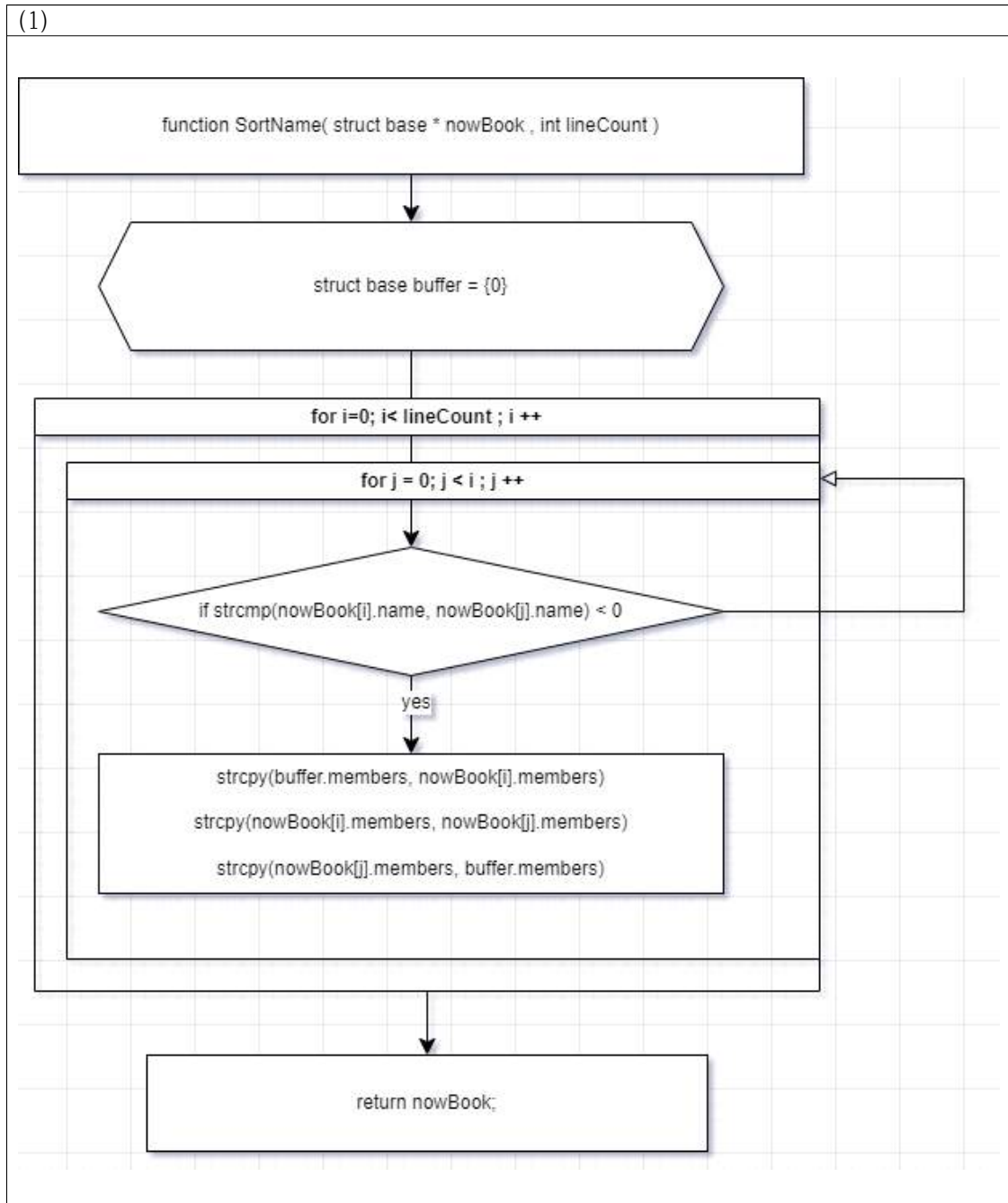
함수에 들어가게 되면, strlen를 이용하여 arr의 길이를 측정하고 size 보다 같거나 크면 1의 값을 반환한다. 문자열은 끝에 널 문자를 포함해야 하는데, strlen은 널 문자 전까지의 길이만 측정하기 때문에 size와 같아서는 안되는 점을 이용했다.

만약 arr 내부에 “ ” 기호를 포함했을 때도 예외 처리로 1을 반환했다.

스페이스바를 파일 입출력에서 다루기 힘들었기 때문에 매 문자열에 “ ”를 입력 받지 않았다.

만일 arr가 size 내부의 길이이고, “ ”가 없다면 0값을 반환하여 함수의 정상 종료를 알렸다.

(3) SortName 함수




```

#include "numbook.h"

struct base * SortName(struct base * nowBook , int lineCount) {
    struct base buffer = { 0 };
    for (int i = 0; i < lineCount; i++) {
        for (int j = 0; j < i; j++) {
            if ( strcmp(nowBook[i].name , nowBook[j].name) < 0 ) { // 사전순으로 이름 문자열을 검사하는 strcmp 함수 이용 , 만약에 값이 음수면 사전순이 아니다
                strcpy(buffer.name, nowBook[i].name);
                strcpy(buffer.num, nowBook[i].num);
                strcpy(buffer.relation, nowBook[i].relation);
                strcpy(buffer.mail, nowBook[i].mail);
                strcpy(buffer.sos, nowBook[i].sos);

                strcpy(nowBook[i].name, nowBook[j].name);
                strcpy(nowBook[i].num, nowBook[j].num);
                strcpy(nowBook[i].relation, nowBook[j].relation);
                strcpy(nowBook[i].mail, nowBook[j].mail);
                strcpy(nowBook[i].sos, nowBook[j].sos);

                strcpy(nowBook[j].name, buffer.name);
                strcpy(nowBook[j].num, buffer.num);
                strcpy(nowBook[j].relation, buffer.relation);
                strcpy(nowBook[j].mail, buffer.mail);
                strcpy(nowBook[j].sos, buffer.sos);
                buffer = { 0 };
            }
        }
    }
    return nowBook;
}

```

프로그램 진행 시에 출력이 되는 부분은 2개의 메뉴이다.

2. 검색 및 수정 과 3. 전화번호부 출력이다.

이때마다 원활하게 수정 및 출력하기 위해 전화번호부의 메뉴에 돌아올 때 마다 전화번호부를 이름을 기준으로 사전식으로 정렬하여 관리하도록 했다.

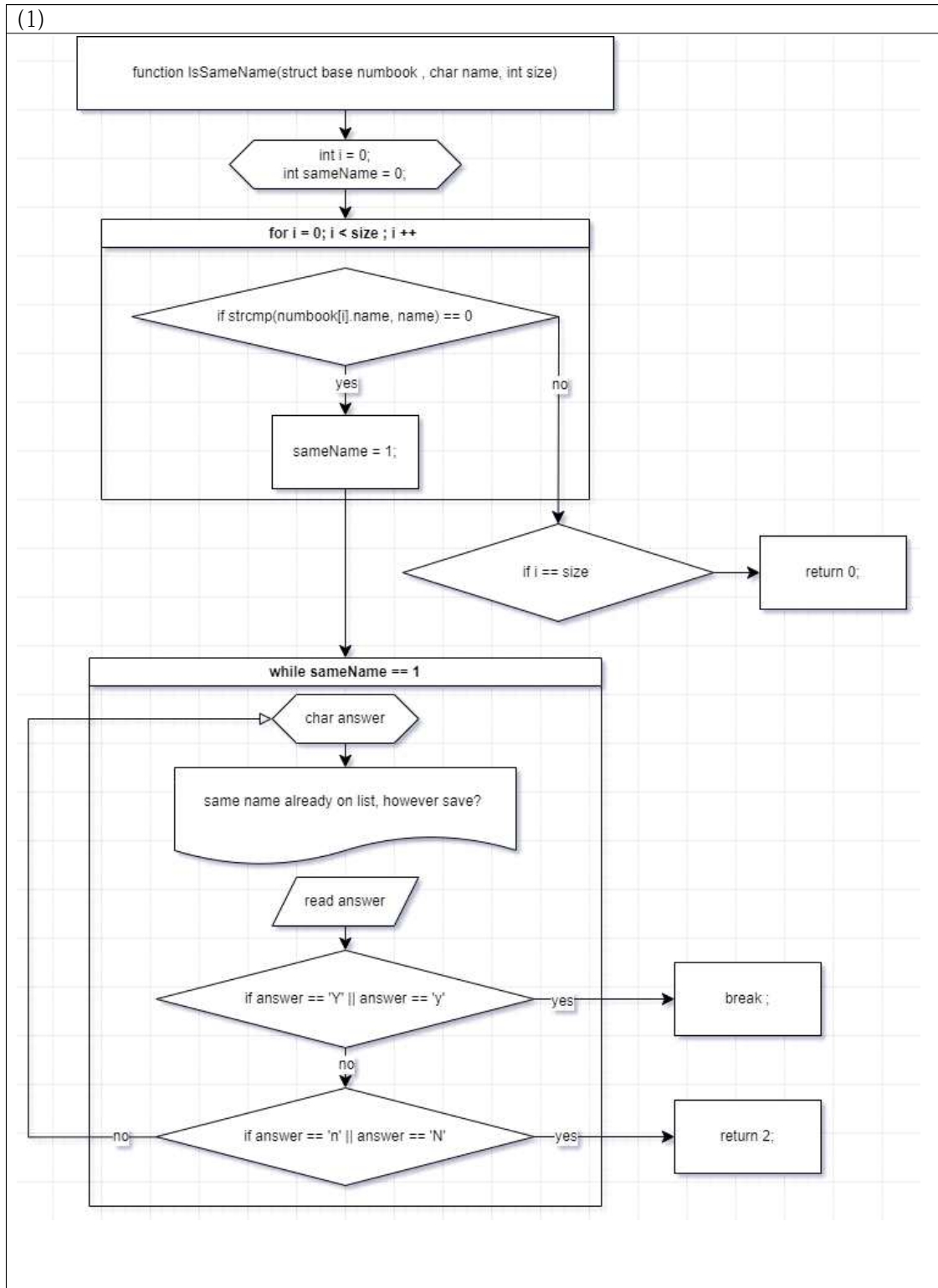
함수의 호출에서 현재 전화번호부와 전체 개수를 받는다.

중첩된 반복문을 이용하여,

strcmp 는 두 문자열이 같으면 0을 반환 해 주지만, 사전 순으로 되어있는지 검사도 해주는 것을 이용했다.

strcmp는 문자열을 두 개 입력받는다. 만약 앞의 문자열이 사전순으로 앞서 있다면 음수의 값을 반환한다. 따라서 nowBook[i] 와 nowBook[j] 의 값을 비교했을 때, 양수 또는 0이면 사전순으로 정렬되어있다는 뜻이고 음수이면 사전순으로 정렬되어있지 않다는 뜻이다. (j < i) 이기 때문에 j는 i보다 위치상 앞서있다. 이런 경우 두 값을 buffer를 이용하여 nowBook[i]를 buffer 에 옮기고, nowBook[j]의 값을 nowBook[i]로 옮기고, 다시 buffer 의 값을 nowBook[j]에 옮겼다. swap 함수 같은 개념이다.

(4) IsSameName 함수



```

#include "numbook.h"

int sameName = 0;

int IsSameName(struct base* nowbook, char* name, int size) {
    int i = 0;
    for ( i = 0; i < size; i++) {
        if ( strcmp(nowbook[i].name, name) == 0 ) {
            sameName = 1;
            break;
        }
    }
    if (i == size) {
        return 0;
    }

    while (sameName == 1) {
        char answer = 0;
        char check = 0;

        printf("... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까? \n[Y/N]");

        scanf(" %c", &answer);

        do { check = getchar();
        } while (check != '\n');

        if (answer == 'y' || answer == 'Y') {
            printf("\n... [ %s ] 을/를 저장합니다. \n", name);
            sameName = 0;
            break;
        }
        else if (answer == 'n' || answer == 'N') {
            printf("\n... 메뉴로 돌아갑니다. \n");
            sameName = 0;
            return 2;
        }
        else {
            printf("\n... 올바른 문자를 입력 해 주세요. \n");
        }
    }
}

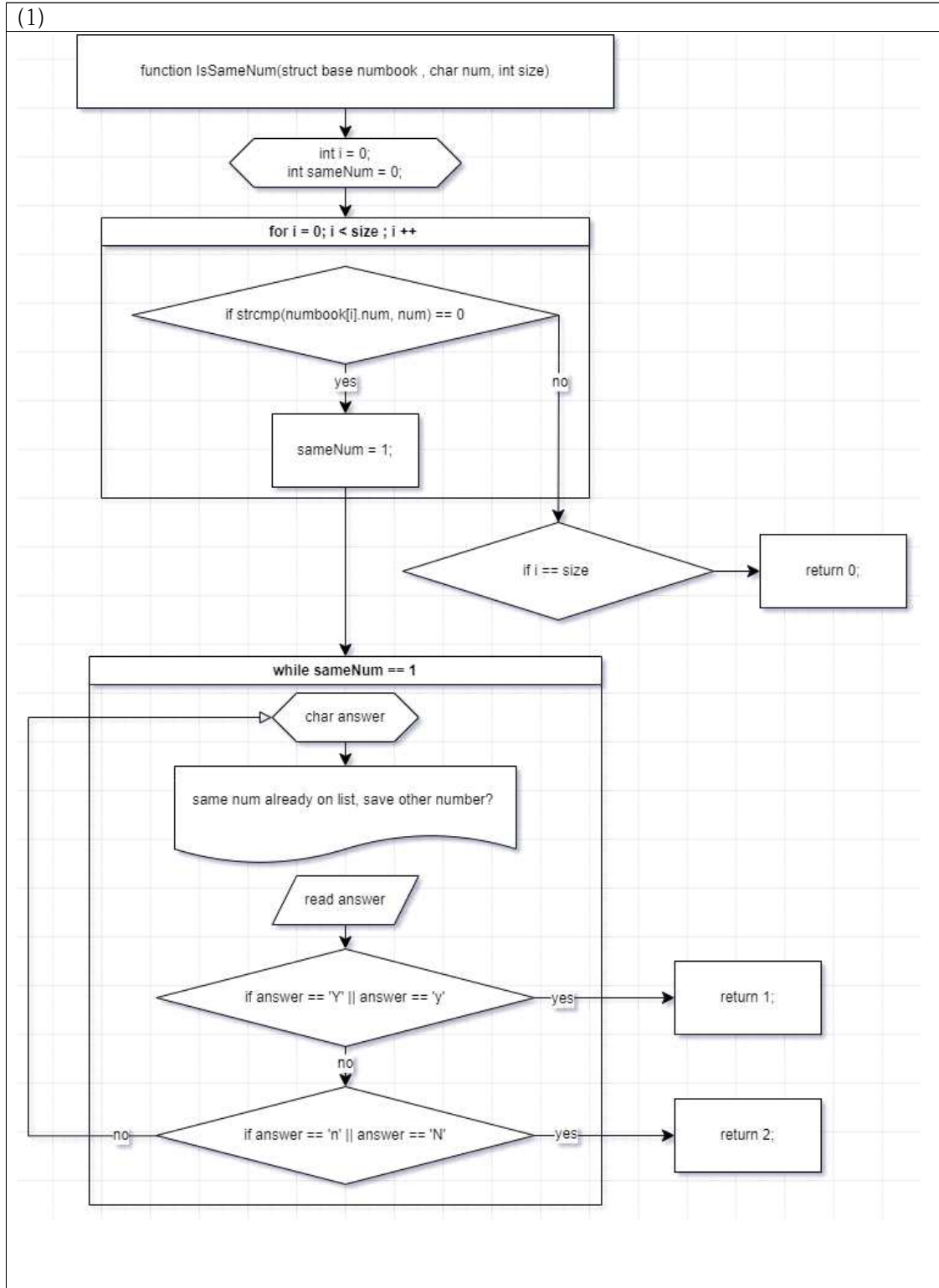
```

IsSame XXX 함수는 기존의 연락처에 입력한 항목이 존재하는지 검사를 도와주는 함수다.
만약 IsSameName 이면 기존의 연락처에 앞으로 입력할 이름이 존재하는지 검사를 도와준다.
IsSameName 함수는 현재의 연락처를 nowbook, 입력할 이름을 name, 전체 전화번호부의 개수를 size로 입력 받는다.
동일한 문자열을 발견하는 방법은 반복문과 strcmp 함수를 이용했다.
만약 동일한 문자열을 발견했을 경우를 위하여 sameName 변수를 0으로 초기화 하여 만들었다. 발견했을 경우, 이 값은 1로 바뀐다.

sameName 가 1일 때,
전화번호에 동일한 이름이 존재한다는 것을 알리고, 그래도 저장 할지 아니면 저장을 취소할지 물어본다.
대답을 입력 받는 변수는 answer로 설정했다.
만일 answer 이 'Y'나 'y'의 값을 가지면 , 그대로 저장하고 리턴 값으로 0 값을 보낸다.

answer 이 'N'나 'n'의 값을 가지면 , 저장을 취소하는 리턴 코드인 2를 보냈다.
이외의 값을 입력하면 대답을 다시 입력하도록 했다.

(5) IsSameNum 함수



```

int IsSameNum(struct base* nowbook, char * num, int size) {
    int i = 0;
    int sameNum = 0;
    for (i = 0; i < size; i++) {
        if ( strcmp( nowbook[i].num , num ) == 0 ) {
            sameNum = 1;
            break;
        }
    }
    if (i == size ) {
        return 0;
    }

    while (sameNum == 1) {
        char answer = 0;
        char check = 0;

        printf("... 전화번호부에 동일한 전화번호가 존재합니다. 다른 전화번호를 저장 하시겠습니까? \n[Y/N]\n");
        scanf("%c", &answer);
        while (check != '\n') {
            check = getchar();
        }

        if (answer == 'y' || answer == 'Y') {
            printf("\n... 저장을 원하는 전화번호를 다시 입력하세요.\n");
            sameNum = 0;
            return 1;
        }
        else if (answer == 'n' || answer == 'N') {
            printf("\n... 메뉴로 돌아갑니다. \n");
            sameNum = 0;
            return 2;
        }
        else {
            printf("\n... 올바른 문자를 입력 해 주세요.\n");
        }
    }
}

```

IsSame XXX 함수는 기존의 연락처에 입력한 항목이 존재하는지 검사를 도와주는 함수다.
만약 IsSameNum 이면 기존의 연락처에 앞으로 입력할 전화번호가 존재하는지 검사를 도와준다.

IsSameNum 함수는 현재의 연락처를 nowbook, 입력할 전화번호를 num, 전체 전화번호부의 개수를 size로 입력 받는다.

동일한 문자열을 발견하는 방법은 반복문과 strcmp 함수를 이용했다.

만약 동일한 문자열을 발견했을 경우를 위하여 sameNum 변수를 0으로 초기화 하여 만들었다. 발견했을 경우, 이 값은 1로 바뀐다.

sameNum 가 1일 때,

전화번호에 동일한 전화번호가 존재함을 알리고 다른 전화번호를 저장 할지, 아니면 저장을 취소할지 물어본다.

전화번호는 고유하기 때문에 중복 저장은 불가하게 설정했다.

대답을 입력 받는 변수는 answer로 설정했다.

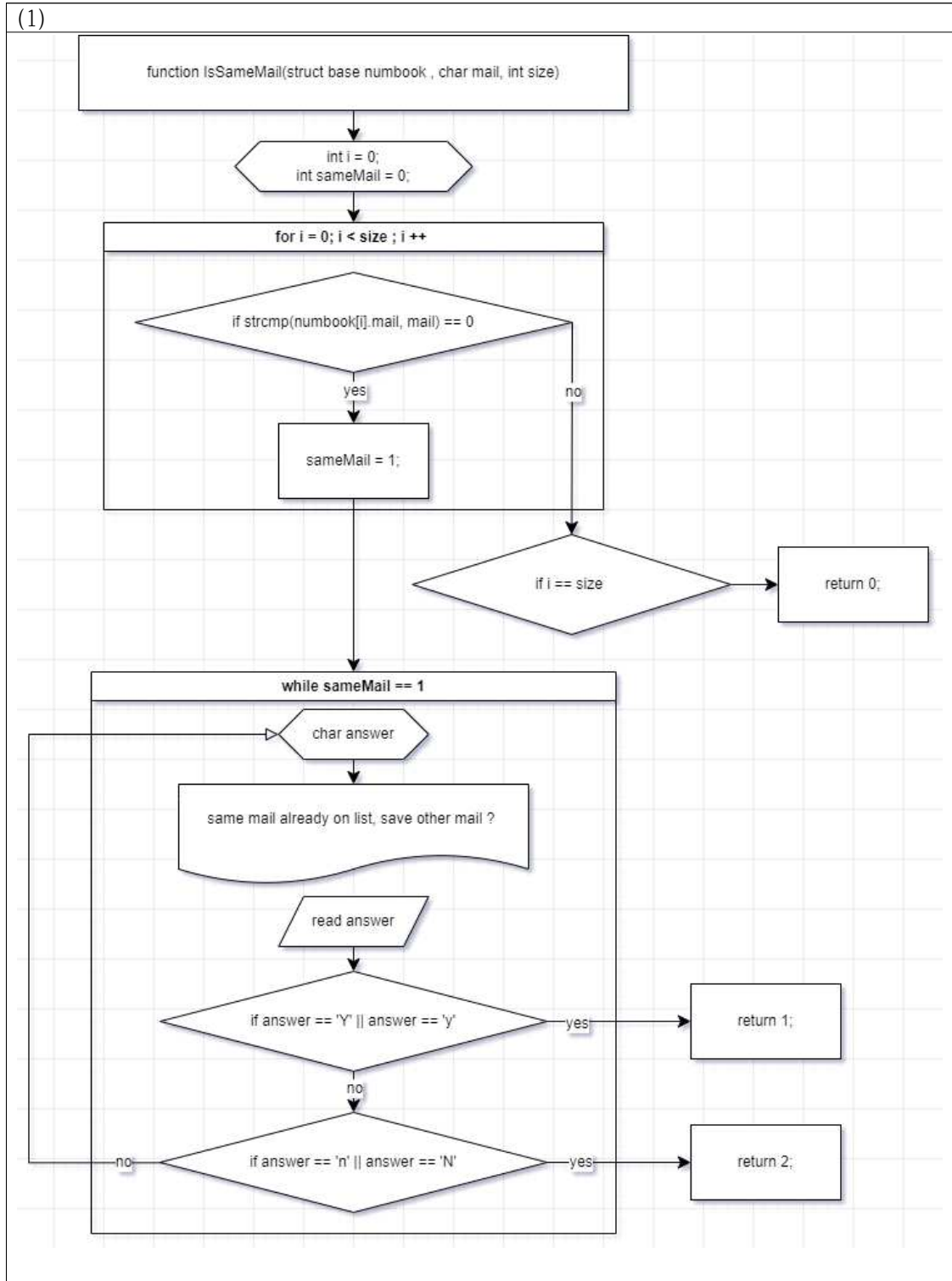


만일 answer 이 'Y'나 'y'의 값을 가지면 , 다른 전화번호를 입력받게 하는 리턴값인 1을 보낸다.

answer 이 'N'나 'n'의 값을 가지면 , 저장을 취소하는 리턴 코드인 2을 보냈다.

이외의 값을 입력하면 대답을 다시 입력하도록 했다.

(6) IsSameMail 함수



```

int sameMail = 0;
int IsSameMail(struct base* nowbook, char* mail, int size) {
    int i = 0;
    for (i = 0; i < size; i++) {
        if (strcmp(nowbook[i].mail, mail) == 0) {
            sameMail = 1;
            break;
        }
    }

    if (i == size) {
        return 0;
    }

    if (sameMail == 1) {
        printf("... 전화번호부에 동일한 이메일이 존재합니다. 다른 이메일을 저장 하시겠습니까? \n[Y/N]");
    }

    while (sameMail == 1) {
        char answer = 0;
        char check = 0;

        scanf("%c", &answer);
        do {
            check = getchar();
        } while (check != '\n');

        if (answer == 'y' || answer == 'Y') {
            printf("\n... 저장을 원하는 이메일을 다시 입력하세요.\n");
            sameMail = 0;
            return 1;
        }

        else if (answer == 'n' || answer == 'N') {
            printf("\n... 메뉴로 돌아갑니다. \n");
            sameMail = 0;
            return 2;
        }

        else {
            printf("\n... 올바른 문자를 입력 해 주세요.\n");
        }
    }
}

```

IsSame XXX 함수는 기존의 연락처에 입력한 항목이 존재하는지 검사를 도와주는 함수다.
만약 IsSameMail 이면 기존의 연락처에 앞으로 입력할 이메일이 존재하는지 검사를 도와준다.
IsSameMail 함수는 현재의 연락처를 nowbook, 입력할 이메일을 mail, 전체 전화번호부의 개수를 size로 입력 받는다.

동일한 문자열을 발견하는 방법은 반복문과 strcmp 함수를 이용했다.

만약 동일한 문자열을 발견했을 경우를 위하여 sameMail 변수를 0으로 초기화 하여 만들었다. 발견했을 경우, 이 값은 1로 바뀐다.

sameMail 가 1일 때,

전화번호부에 동일한 이메일이 존재함을 알리고 다른 이메일을 저장 할지, 아니면 저장을 취소 할지 물어본다.

이메일 또한 고유하기 때문에 중복 저장은 불가하게 설정했다.

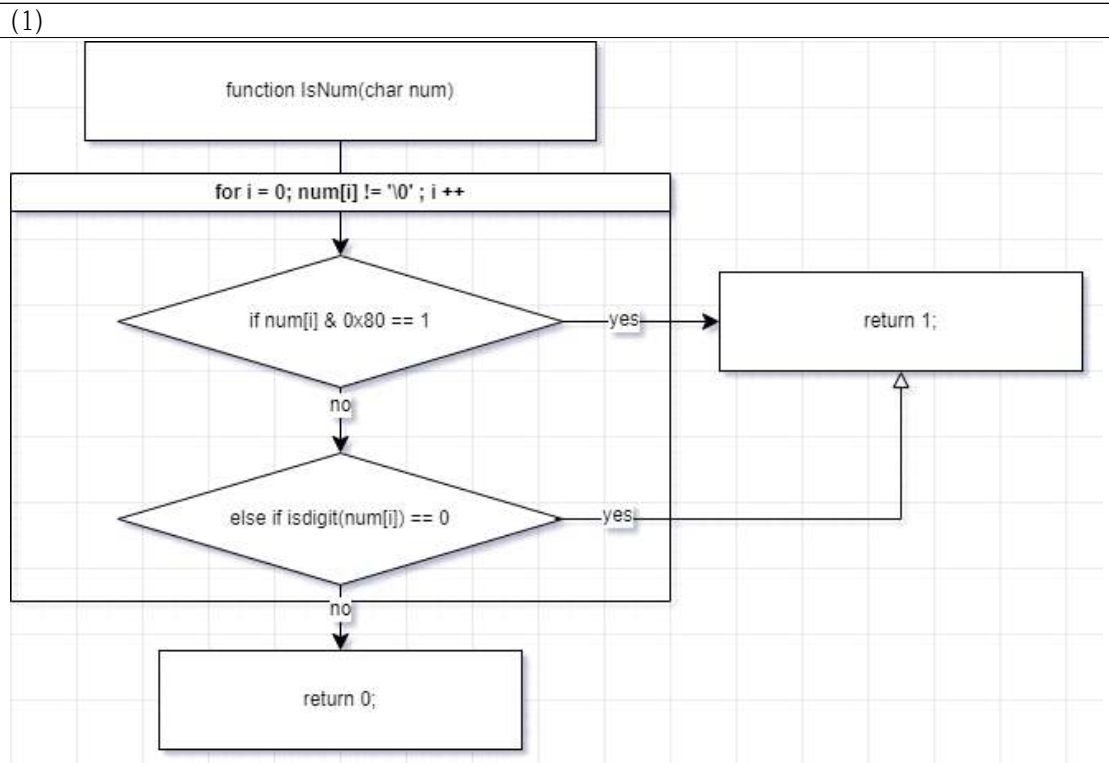
대답을 입력 받는 변수는 answer로 설정했다.

만일 answer 이 'Y'나 'y'의 값을 가지면 , 다른 이메일을 입력받게 하는 리턴값인 1을 보낸다.



answer 이 'N'나 'n'의 값을 가지면 , 저장을 취소하는 리턴 코드인 2을 보냈다.
이외의 값을 입력하면 대답을 다시 입력하도록 했다.

(7) IsNum 함수



```

int IsNum(char* num) {
    for (int i = 0; num[i] != '\0'; i++) { // null 문자까지 확인
        if (num[i] & 0x80) { // 한글인 경우
            printf("... 숫자로만 입력하세요. \n");
            return 1; // 오류 코드 반환
        }
        else if ( isdigit(num[i]) == 0 ) { // 숫자가 아닐 경우(영어), 특수기호일 경우
            printf("... 숫자로만 입력하세요. \n");
            return 1; // 오류 코드 반환
        }
    }
    return 0;
}
  
```

IsNum은 함수에 전해지는 문자열 num 이 숫자로만 구성되었는지 검사한다.

먼저 캐릭터 형의 변수로 숫자를 입력 받았을 때, 각 숫자의 아스키 코드값에 대응 시켜 숫자면 0이 아닌 값, 숫자가 아니면 0을 리턴 해주는 isdigit 함수를 이용해서 숫자인지 검사를 했는데, 이 경우 한글이 입력되면 오류가 뜨는 것을 확인했다.

따라서 한글인 경우를 따로 구분했다.

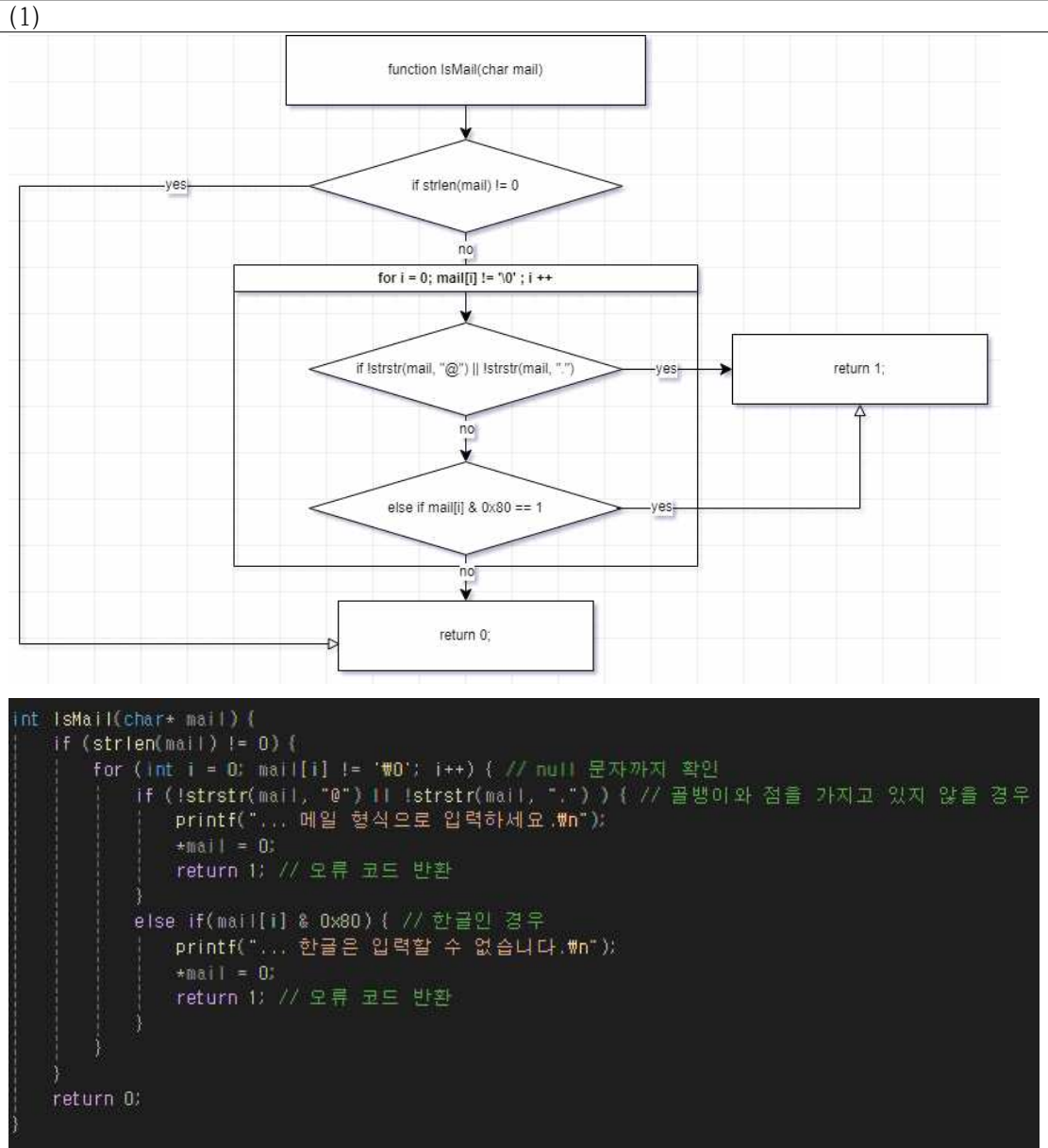
한글을 입력할 때는 0X80 (1000 0000) 보다 큰 값이 들어간다고 한다.

따라서 & 연산자로 입력받은 문자열과 비트 연산을 했을 때 (1000 0000) 인 1이 들어가면 한

글, 아니면 한글이 아니게 된다. 이 개념을 이해 하는것에 어려움이 있었지만 한글의 저장 방법을 안다면 활용할 수 있는 개념이라고 생각한다.

반복문을 이용하여 문자열 num이 널 문자(문자열의 끝)를 지시하기 전까지의 배열을 한 자리씩 조건문에 맞추어 검사한다.

(8) IsMail 함수



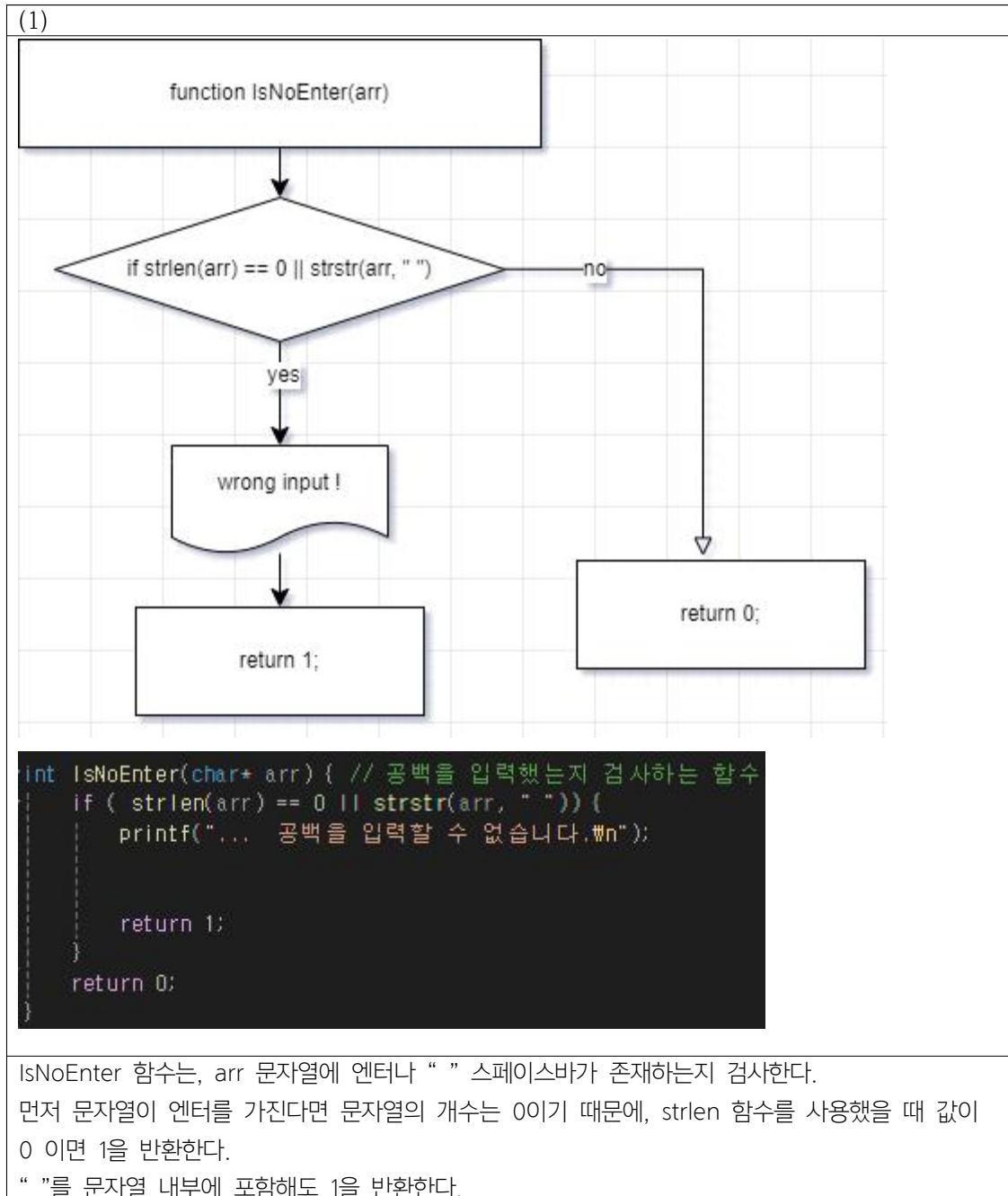
IsMail 함수는 입력된 mail 이 메일 형식에 맞는지 검사한다.

먼저 메일에는 '@'와 '.' 기호가 들어가므로, 이 기호가 들어가지 않으면 1을 반환한다.

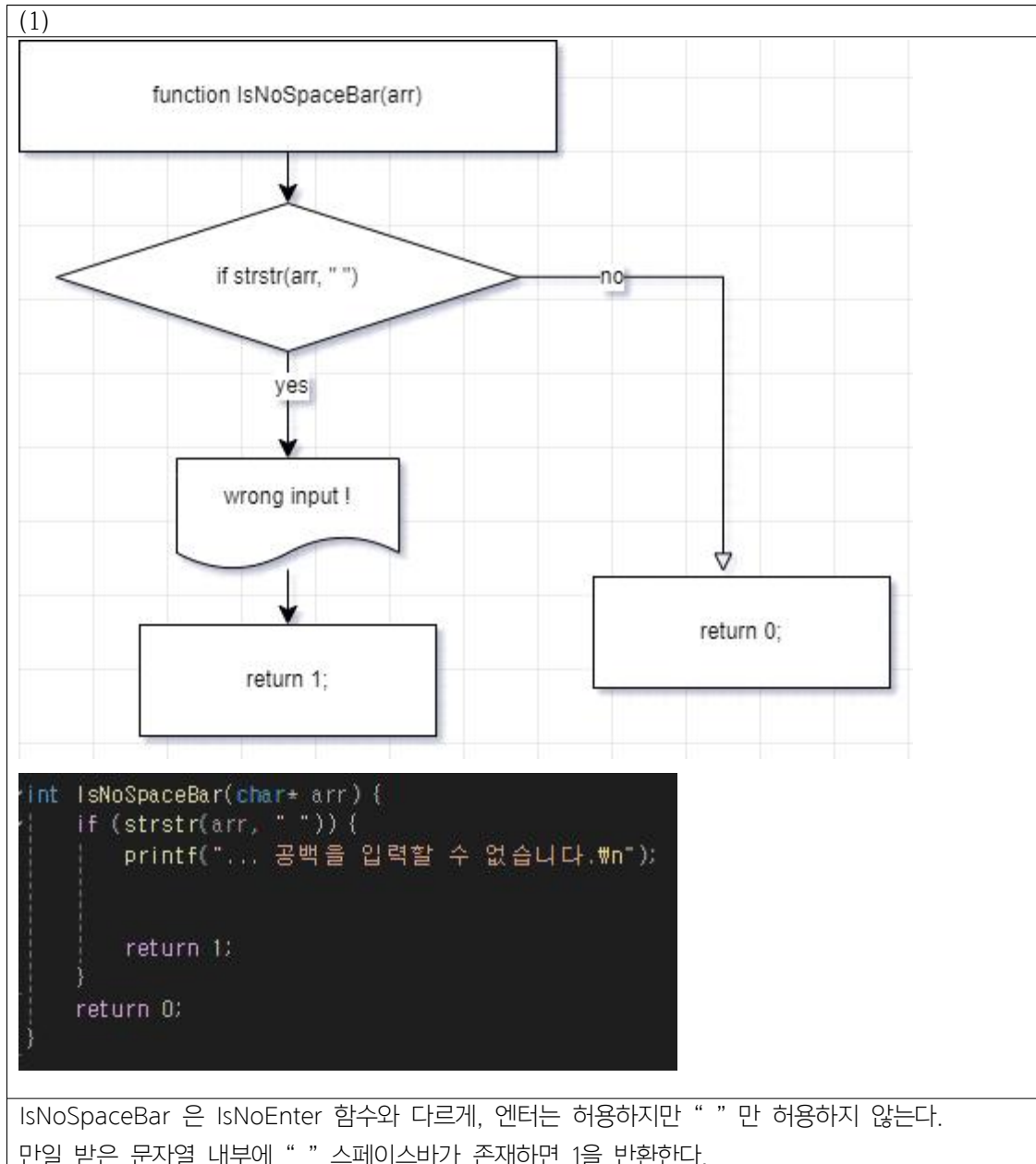
또한 IsNum 함수와 마찬가지로 한글을 입력하면 1을 반환하여 다시 입력 하도록 해준다.

반복문을 이용하여 문자열 mail이 널 문자(문자열의 끝)를 지시하기 전까지의 배열을 한 자리씩 조건문에 맞추어 검사한다.

(9) IsNoEnter 함수

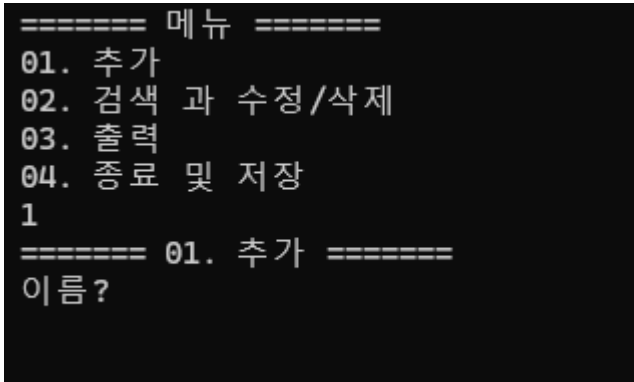


(10) IsNoSpaceBar 함수



3. 전화번호부 코드 진행 예시

메뉴 번호	작동 예시
0. 진입	<p>프로그램 실행 시</p> <pre> ... 전화번호부를 세팅합니다. ... 'book.txt' 파일에서 데이터를 읽는 중... ... 데이터 읽기 완료 ! ... 전화번호부를 실행하려면 아무 키나 눌러주세요. </pre>
	<p>아무 키나 눌렀을 때</p> <pre> ===== 메뉴 ===== 01. 추가 02. 검색 과 수정 /삭제 03. 출력 04. 종료 및 저장 </pre>
	<p>잘못된 메뉴 번호를 입력했을 때</p> <pre> ===== 메뉴 ===== 01. 추가 02. 검색 과 수정 /삭제 03. 출력 04. 종료 및 저장 안녕하세요 ... 메뉴와 맞는 숫자를 입력하세요. 5 ... 메뉴와 맞는 숫자를 입력하세요. 0 ... 메뉴와 맞는 숫자를 입력하세요. abcd ... 메뉴와 맞는 숫자를 입력하세요. </pre>

<p>1. 추가</p>	<p>추가 메뉴 진입 시</p>  <pre> ===== 메뉴 ===== 01. 추가 02. 검색 과 수정 /삭제 03. 출력 04. 종료 및 저장 1 ===== 01. 추가 ===== 이름? </pre>
--------------	--

잘못된 이름 입력 시

```
===== 01. 추가 =====
이름?
안녕하세요반갑습니다저는강원대학교에다니는하승연이라고합니다잘부탁드립니다!!!!!!반가워요정말!!!!!!!!!!!!!!!!!!!!!!
... 글자의 개수가 올바르지 않습니다.
이름?
스페 이스 바를 입력 해 보자
... 공백을 입력할 수 없습니다.
이름?
```

제대로 된 이름 입력 시

```
이름?
하승연
전화번호?
|
```

이미 입력된 이름을 저장할 경우

```
===== 01. 추가 =====
이름?
강원대학교
... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까?
[Y/N]
```

이미 입력된 이름을 저장할 경우 - y를 누를 경우

```
===== 01. 추가 =====
이름?
강원대학교
... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까?
[Y/N]y
... [ 강원대학교 ] 을/를 저장합니다.
전화번호?
```

이미 입력된 이름을 저장할 경우 - n를 누를 경우

```
===== 01. 추가 =====
이름?
강원대학교
... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까?
[Y/N]n
... 메뉴로 돌아갑니다.
... 아무 키나 눌러주세요.
```



잘못된 전화번호 입력 시

```
전화번호?  
010-1234-5678  
... 숫자로만 입력하세요.  
전화번호?  
010123456789  
... 글자의 개수가 올바르지 않습니다.  
전화번호?  
안녕하세요  
... 숫자로만 입력하세요.  
전화번호?  
010 1234 5678  
... 공백을 입력할 수 없습니다.  
전화번호?
```

올바른 전화번호 입력 시

```
전화번호?  
01048250209  
관계? (생략을 원할 시 엔터를 누르세요.)
```

이미 입력된 전화번호를 저장하는 경우

```
전화번호?  
033  
... 전화번호부에 동일한 전화번호가 존재합니다. 다른 전화번호를 저장 하시겠습니까?  
[Y/N]
```

이미 입력된 전화번호를 저장하는 경우 - y를 입력하는 경우

```
... 전화번호부에 동일한 전화번호가 존재합니다. 다른 전화번호를 저장 하시겠습니까?  
[Y/N]  
y  
... 저장을 원하는 전화번호를 다시 입력하세요.  
전화번호?
```

이미 입력된 전화번호를 저장하는 경우 - n를 입력하는 경우

```
... 전화번호부에 동일한 전화번호가 존재합니다. 다른 전화번호를 저장 하시겠습니까?  
[Y/N]  
n  
... 메뉴로 돌아갑니다.  
... 아무 키나 눌러주세요.
```

잘못된 관계 입력 시

관계? (생략을 원할 시 엔터를 누르세요.)
너는누구니?나는하승연이라고해만나서정말반가워!!!!!!!!!!앞으로잘지내보자야
호!!!!!!!!!!!!
... 글자의 개수가 올바르지 않습니다.
관계? (생략을 원할 시 엔터를 누르세요.)
스페 이스 바를 입력 해 보자
... 공백을 입력할 수 없습니다.
관계? (생략을 원할 시 엔터를 누르세요.)

생략을 하여 입력하는 경우

관계? (생략을 원할 시 엔터를 누르세요.)
메일? (생략을 원할 시 엔터를 누르세요.)

올바른 입력을 하는 경우

관계? (생략을 원할 시 엔터를 누르세요.)
학과친구
메일? (생략을 원할 시 엔터를 누르세요.)

	<p>생략을 하여 입력하는 경우</p> <pre> 메일? (생략을 원할 시 엔터를 누르세요.) 긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요) [Y/N] </pre> <p>잘못된 값을 입력하는 경우</p> <pre> 메일? (생략을 원할 시 엔터를 누르세요.) 안녕하세요 ... 메일 형식으로 입력하세요. 메일? (생략을 원할 시 엔터를 누르세요.) 스페 이스 바 ... 메일 형식으로 입력하세요. 메일? (생략을 원할 시 엔터를 누르세요.) naver.com ... 메일 형식으로 입력하세요. 메일? (생략을 원할 시 엔터를 누르세요.) </pre> <p>올바른 메일 값을 입력하는 경우</p> <pre> 메일? (생략을 원할 시 엔터를 누르세요.) kangwan033@naver.com 긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요) [Y/N] </pre>
--	---

	<p>이미 입력한 메일 값을 저장하는 경우</p> <pre> 메일? (생략을 원할 시 엔터를 누르세요.) kangwan033@naver.com ... 전화번호부에 동일한 이메일이 존재합니다. 다른 이메일을 저장 하시겠습니까? [Y/N] </pre> <p>이미 입력한 메일 값을 저장하는 경우 - y를 입력할 경우</p> <pre> 메일? (생략을 원할 시 엔터를 누르세요.) kangwan033@naver.com ... 전화번호부에 동일한 이메일이 존재합니다. 다른 이메일을 저장 하시겠습니까? [Y/N]y ... 저장을 원하는 이메일을 다시 입력하세요. 메일? (생략을 원할 시 엔터를 누르세요.) </pre> <p>이미 입력한 메일 값을 저장하는 경우 - y를 입력할 경우</p> <pre> 메일? (생략을 원할 시 엔터를 누르세요.) kangwan@naver.com ... 전화번호부에 동일한 이메일이 존재합니다. 다른 이메일을 저장 하시겠습니까? [Y/N]n ... 메뉴로 돌아갑니다. ... 아무 키나 눌러주세요. </pre>
--	---

잘못된 값을 긴급연락처에 등록하는 경우

```

긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요)
[Y/N]
abcd
... 글자의 개수가 올바르지 않습니다.
안녕
... 글자의 개수가 올바르지 않습니다.
a
... 올바른 값을 입력하세요.
긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요)
[Y/N]

```

생략된 값을 입력하는 경우

```

긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요)
[Y/N]
=====
이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
김강원                0331230009                학과친구                kangwan033@naver.com -
=====
... 저장 완료!
... 아무 키나 눌러주세요.

```

y를 입력하는 경우

```

긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요)
[Y/N]
y
=====
이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
경찰서                112                경찰                -                0
=====
... 저장 완료!
... 아무 키나 눌러주세요.

```

n을 입력하는 경우

```

긴급 연락처로 등록 하시겠습니까? (생략을 원할 시 엔터를 누르세요)
[Y/N]
n
=====
이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
구급대원                119                응급실                -                -
=====
... 저장 완료!
... 아무 키나 눌러주세요.

```

출력 (관계, 메일이 엔터 값으로 생략 된 경우 , 긴급연락처가 등록 된 경우)

이름	전화번호	관계	메일	긴급연락처 등록 여부
이승연	01048250209	-	-	0

... 저장 완료!
... 아무 키나 눌러주세요.

출력 (관계, 메일의 정상 값, 긴급연락처로 등록되지 않은 경우)

이름	전화번호	관계	메일	긴급연락처 등록 여부
강원대학교	033	학교	knu@naver.com	-

... 저장 완료!
... 아무 키나 눌러주세요.



2. 검색

검색 메뉴 진입 시

```
===== 메뉴 =====
01. 추가
02. 검색 과 수정/삭제
03. 출력
04. 종료 및 저장

2
===== 02. 검색 과 수정/삭제 =====
... 검색어?
```

목록에 없는 검색어 입력 시

```
===== 02. 검색 과 수정/삭제 =====
... 검색어?
!
... 총 0개의 목록 검색!
... 아무 키나 눌러주세요.
```

목록이 많은 검색어 입력 시

```
===== 02. 검색 과 수정/삭제 =====
... 검색어?
하
... 총 5개의 목록 검색!
=====
번호  이름              전화번호              관계              메일              긴급연락처  등록  여부
=====
1   김하나              01099982223          -                  -                  -                  -
2   박하나              01092748385          -                  -                  -                  -
3   주하나              01099927773          -                  -                  -                  -
4   하승연              01048250209          나자신            ha020206@naver.com  -                  -
5   한하나              01036462473          -                  -                  -                  -
=====
===== 다음 작업 =====
01. 수정
02. 삭제
03. 검색 종료
```

이름 수정 시

- 김하나를 하나김으로 변경

```

===== 02. 검색 과 수정/삭제 =====
... 검색어?
하나
... 총 4개의 목록 검색!

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 김하나                01099982223                -                -                -
2 박하나                01092748385                -                -                -
3 주하나                01099927773                -                -                -
4 한하나                01036462473                -                -                -
=====

===== 다음 작업 =====
01. 수정
02. 삭제
03. 검색 종료
1
... 수정을 원하는 목록의 번호?
1

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 김하나                01099982223                -                -                -

... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
1
... 수정 할 내용? 현재 내용 : 김하나
하나김
... 작업을 저장합니다.

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 하나김                01099982223                -                -                -

... 아무 키나 눌러주세요.

```

- 하나김을 이미 존재하는 이름으로 변경

```
===== 02. 검색 과 수정/삭제 =====
... 검색어?
하나
... 총 4개의 목록 검색!

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 박하나                01092748385                -                -                -
2 주하나                01099927773                -                -                -
3 하나김                01099982223                -                -                -
4 한하나                01036462473                -                -                -
=====

===== 다음 작업 =====
01. 수정
02. 삭제
03. 검색 종료
1
... 수정을 원하는 목록의 번호?
3

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 하나김                01099982223                -                -                -
=====

... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
1
... 수정 할 내용? 현재 내용 : 하나김
강원대학교
... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까?
[Y/N]
```

- - y를 눌렀을 때

```
... 수정을 원하는 목록의 번호?
3

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 하나김                01099982223                -                -                -
=====

... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
1
... 수정 할 내용? 현재 내용 : 하나김
강원대학교
... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까?
[Y/N]y

... [ 강원대학교 ] 을/를 저장합니다.
... 작업을 저장합니다.

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 강원대학교                01099982223                -                -                -
=====

... 아무 키나 눌러주세요.
```

- - n을 눌렀을 때

```
... 수정을 원하는 목록의 번호?
3

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 하나김                01099982223                -                -                -
=====

... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
1
... 수정 할 내용? 현재 내용 : 하나김
강원대학교
... 전화번호부에 동일한 이름이 존재합니다. 그래도 저장 하시겠습니까?
[Y/N]n

... 메뉴로 돌아갑니다.
... 작업을 저장합니다.

=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1 하나김                01099982223                -                -                -
=====

... 아무 키나 눌러주세요.
```

전화번호 수정 시 -033을 123으로 변경

```

강원
... 총 2개의 목록 검색!
=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1   강원대학교            033                학교                kangwan@naver.com    0
2   강원대학교학과사무실  03392837463        -                  -                  0
=====
===== 다음 작업 =====
01. 수정
02. 삭제
03. 검색 종료
1
... 수정을 원하는 목록의 번호?
1
=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1   강원대학교            033                학교                kangwan@naver.com    0
=====
... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
2
... 수정 할 내용? 현재 내용 : 033
123
... 작업을 저장합니다.
=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1   강원대학교            123                학교                kangwan@naver.com    0
=====
... 아무 키나 눌러주세요.

```

-123을 이미 존재하는 전화번호로 변경

- y를 눌렀을 때

```

... 수정을 원하는 목록의 번호?
1
=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1   강원대학교            123                학교                kangwan@naver.com    0
=====
... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
2
... 수정 할 내용? 현재 내용 : 123
01048250209
... 전화번호부에 동일한 전화번호가 존재합니다. 다른 전화번호를 저장 하시겠습니까?
[Y/N]
y
... 저장을 원하는 전화번호를 다시 입력하세요.
... 수정 할 내용? 현재 내용 : 123

```

- n을 눌렀을 때

```

01048250209
... 전화번호부에 동일한 전화번호가 존재합니다. 다른 전화번호를 저장 하시겠습니까?
[Y/N]
n
... 메뉴로 돌아갑니다.
... 작업을 저장합니다.
=====
번호 이름                전화번호                관계                메일                긴급연락처 등록 여부
=====
1   강원대학교            123                학교                kangwan@naver.com    0
=====
... 아무 키나 눌러주세요.

```


관계 수정 시

- 나자신 을 우리엄마딸로 변경

```

1
=====
번호 이름              전화번호              관계              메일              긴급연락처 등록 여부
=====
1 하승연              01048250209        나자신              ha020206@naver.com -
=====
... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
3
... 수정 할 내용? 현재 내용 : 나자신
우리엄마딸
... 작업을 저장합니다.
=====
번호 이름              전화번호              관계              메일              긴급연락처 등록 여부
=====
1 하승연              01048250209        우리엄마딸          ha020206@naver.com -
=====
... 아무 키나 눌러주세요.
  
```

이메일 수정 시

- kangwon@naver.com 을 kang033@naver.com 으로 변경

```

... 수정을 원하는 목록의 번호?
1
=====
번호 이름              전화번호              관계              메일              긴급연락처 등록 여부
=====
1 강원대학교          123                학교              kangwan@naver.com 0
=====
... 수정 할 항목?
1. 이름
2. 전화번호
3. 관계
4. 메일
5. 긴급 연락처
4
... 수정 할 내용? 현재 내용 : kangwan@naver.com
kang033@naver.com
... 작업을 저장합니다.
=====
번호 이름              전화번호              관계              메일              긴급연락처 등록 여부
=====
1 강원대학교          123                학교              kang033@naver.com 0
=====
... 아무 키나 눌러주세요.
  
```

- 이미 존재하는 이메일로 변경

- y를 눌렀을 때

```

... 수정 할 내용? 현재 내용 : kang033@naver.com
ha020206@naver.com
... 전화번호부에 동일한 이메일이 존재합니다. 다른 이메일을 저장 하시겠습니까?
[Y/N]y
... 저장을 원하는 이메일을 다시 입력하세요.
... 수정 할 내용? 현재 내용 : kang033@naver.com
  
```

- n을 눌렀을 때

```

... 전화번호부에 동일한 이메일이 존재합니다. 다른 이메일을 저장 하시겠습니까?
[Y/N]n
... 메뉴로 돌아갑니다.
... 작업을 저장합니다.
=====
번호 이름              전화번호              관계              메일              긴급연락처 등록 여부
=====
1 강원대학교          010123            -                  kang033@naver.com -
=====
... 아무 키나 눌러주세요.
  
```

긴급 연락처 등록 여부를 수정하는 경우

- 이미 등록 된 경우 등록 해제

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	하승연	01012340987	-	ha020206@naver.com	0

... 수정 할 항목?

1. 이름

2. 전화번호

3. 관계

4. 메일

5. 긴급 연락처

5

... 긴급 연락처에서 해제 하시겠습니까?

[Y/N]

y

... 작업을 저장합니다.

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	하승연	01012340987	-	ha020206@naver.com	-

... 아무 키나 눌러주세요.

- 등록이 안된 경우 등록으로 수정

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	강원대학교	010123	-	kang033@naver.com	-

... 수정 할 항목?

1. 이름

2. 전화번호

3. 관계

4. 메일

5. 긴급 연락처

5

... 긴급 연락처로 등록 하시겠습니까?

[Y/N]

y

... 작업을 저장합니다.

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	강원대학교	010123	-	kang033@naver.com	0

... 아무 키나 눌러주세요.

삭제

... 총 3개의 목록 검색!

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	강원대학교	010123	-	kang033@naver.com	-
2	강하나	01099928374	-	-	-
3	강호원	01029383838	-	-	-

===== 다음 작업 =====

01. 수정

02. 삭제

03. 검색 종료

2

... 삭제를 원하는 목록의 번호?

1

... 삭제 완료!

... 아무 키나 눌러주세요.



	<p>검색 종료 시</p> <pre> ===== 다음 작업 ===== 01. 수정 02. 삭제 03. 검색 종료 3 ... 아무 키나 눌러주세요. </pre>
3. 출력	<p>출력 메뉴 진입 시</p> <pre> ===== 메뉴 ===== 01. 추가 02. 검색 과 수정 /삭제 03. 출력 04. 종료 및 저장 3 ===== 03. 출력 ===== ... 무엇을 출력 하시겠습니까? 01. 전체 전화번호부 출력 02. 긴급 연락처 출력 </pre> <p>잘못된 메뉴 번호를 입력했을 때</p> <pre> ===== 03. 출력 ===== ... 무엇을 출력 하시겠습니까? 01. 전체 전화번호부 출력 02. 긴급 연락처 출력 안녕하세요 ... 메뉴와 맞는 숫자를 입력하세요. 3 ... 메뉴와 맞는 숫자를 입력하세요. </pre> <p>빈 연락처인 상태에서 전체 전화번호부 출력</p> <pre> 번호 이름 전화번호 관계 메 일 긴급연락처 등록 여부 ===== ... 총 0 항목 출력 ... 전화번호부에 연락처가 없습니다. ... 아무 키나 눌러주세요. </pre>

빈 연락처인 상태에서 긴급 연락처 출력

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
... 총 0 항목 출력					
... 전화번호부에 저장된 긴급 연락처가 없습니다.					
... 아무 키나 눌러주세요.					

연락처가 있는 상태에서 전체 전화번호부 출력

===== 03. 출력 =====					
... 무엇을 출력 하시겠습니까?					
01. 전체 전화번호부 출력					
02. 긴급 연락처 출력					
1					
번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	강원대학교	033	학교	kangwan@naver.com	0
2	강원대학교 학과사무실	03392837463	-	-	0
3	경찰서	112	-	-	0
4	김하나	01099982223	-	-	-
5	박하나	01092748385	-	-	-
6	아빠	01098763456	-	-	-
7	엄마	01023459876	-	-	0
8	주하나	01099927773	-	-	-
9	하승연	01048250209	남자친	ha020206@naver.com	-
10	한하나	01036462473	-	-	-
... 총 10 항목 출력					
... 아무 키나 눌러주세요.					

연락처가 있는 상태에서 긴급 연락처 출력

번호	이름	전화번호	관계	메일	긴급연락처 등록 여부
1	강원대학교	033	학교	kangwan@naver.com	0
2	강원대학교 학과사무실	03392837463	-	-	0
3	경찰서	112	-	-	0
4	엄마	01023459876	-	-	0
... 총 4 항목 출력					
... 아무 키나 눌러주세요.					

4. 종료

종료 메뉴 진입 시

===== 메뉴 =====					
01. 추가					
02. 검색 과 수정/삭제					
03. 출력					
04. 종료 및 저장					
4					
===== 04. 종료 및 저장 =====					
... 데이터를 'book.txt' 파일에 저장합니다.					
... 아무 키나 눌러주세요.					

4. 과제 후 느낀 점과 보완 사항

먼저 프로그래밍 과제는 작은 규모의 프로그래밍 과제를 해결 해본 적은 있어도, 이렇게 요구사항이 많은 과제를 처음 접해봤다.

순서도를 제대로 설정하지 않고 마인드맵처럼 손글씨를 작성하면서 과제를 해결했는데, 그렇게 코드를 작성하다 보니 예상보다 코드의 양이 방대하고 처리할게 많았다.

내가 마인드맵으로 그린 순서도의 모양과 실제 코드에 차이가 발생하는 것이다. 예를 들어 출력 함수에서는 전체를 출력하자 라는 초반 설계와 다르게, 긴급 연락처라는 기능을 추가하는 순간 긴급 연락처 출력은 내 순서도에 없었는데,, 하며 당황하게 되는 것이다.

처음에는 모든 코드를 main 소스 파일 하나에서 작업했다. 작업을 하면 할수록 발생하는 예외 처리, 추가 해야 하는 기능들이 머릿속에서 계속 되었고, 내 코드를 내가 못 읽는 지경에 이르러서 모든 함수를 소스파일 하나하나에 분류했다. 이렇게 함수를 분류하고 보니 함수의 기능이 겹치는게 너무 많았다. 끝까지 해결을 해봤는데 완성된 지금 보면 아직도 겹치는게 꽤 있다.

이게 다 순서도 작성의 순서가 잘못되어서 벌어진 일이라고 생각한다.

구상 -> 설계 -> 보완 단계가 아니라

설계 -> 보완 -> 구상 -> 보완 으로 반복되고 있었다.

프로그래밍 과제를 완수한 지금은 앞으로 어떻게 프로그래밍을 해야할지 대략 감이 잡히는 것 같다.

c언어를 집중적으로 공부할 수 있는 좋은 기회였던 것 같다.

아쉬운 점이 있다면, 예외 처리에 집중한 나머지 기능을 더 추가하지 못한 것이 아쉽고,

c언어에 대한 기본 지식이 많이 부족한것에 한계를 느낀다. 1학년 1학기 때 컴퓨터 사고력 강좌를 수강하고 거의 1년 만에 다시 c언어를 마주친 것인데 물론, 함수를 보면 이런 함수구나 기억은 나지만 함수가 작동하는 순서가 가끔 기억이 안난다.

사용자 정의 함수를 호출하면 호출한 대로 리턴 값이 주어지는데, 나는 사용자 정의 함수를 변수처럼 써놓고 어?? 왜 값이 두 번 출력되지?? 이런 적도 있다.

따라서 나의 코드에서 보완할 점은 다음과 같다.

1. 코드를 간결하게 쓰자. 순서도를 작성하지 않고 머릿속으로 구상하는 프로그래밍은 중복과 쓸모없는 코드를 양성시킨다.
2. 예외 처리를 확실하게 하자. 모든 루트에서 리턴 값이 있어야 하는데, 가끔 한 루트에서 리턴 값이 없어 오류가 뜨는 부분이 많다.
3. 효과적인 데이터 관리에 대해 고민하자. 데이터구조 수강 중인 지금은, 아직 데이터를 어떻게 저장해야 하는지 도무지 모르겠어서 친숙한 개념인 구조체 배열을 썼다. 근데 배열 인덱스 번호를 관리하는게 머리가 너무 아프다. 주소 값을 기용하여 편리하게 접근할 수 있는 방향을 좀 더 고민하자.

