파일시스템을 활용한 주소록 만들기

과제의 목적	파일 시스템 배우기

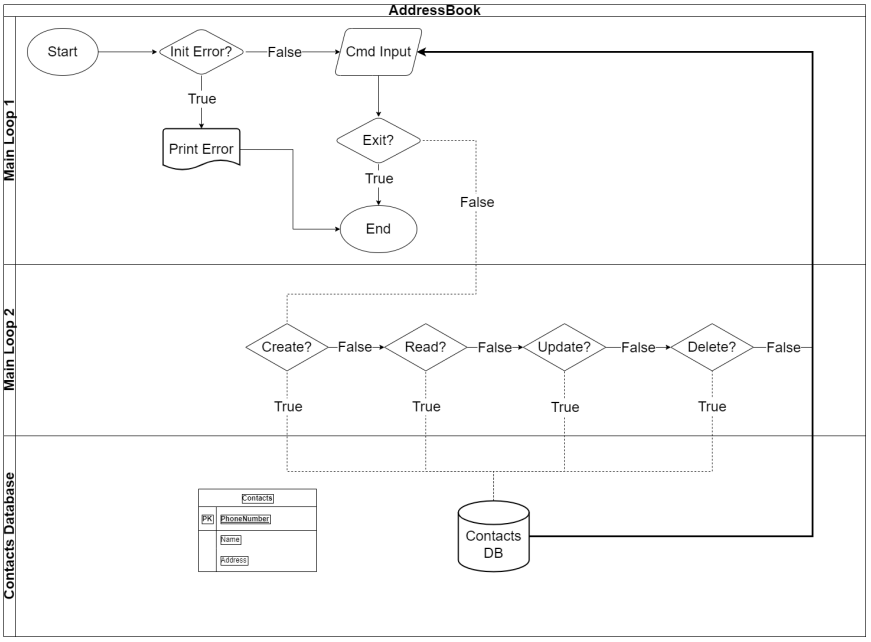
과제 개요

1. 이름, 휴대폰 번호, 주소를 관리할 수 있는 주소록 프로그램 작성
2. 해당 데이터는 파일로 관리
3. 메뉴 출력 및 화면 제어 로직은 Event loop 형식
4. 함수 포인터 테이블을 활용해 호출
5. 최소 1만 개 이상의 더미 데이터를 넣어서 운영
6. 파일 데이터를 인메모리에 들고있어선 안됨
7. SQL SELECT WHERE 스타일의 데이터 검색(Read) 기능 지원
8. 휴대폰 번호 중복 X(이름과 주소는 중복 가능)

사용한 로직

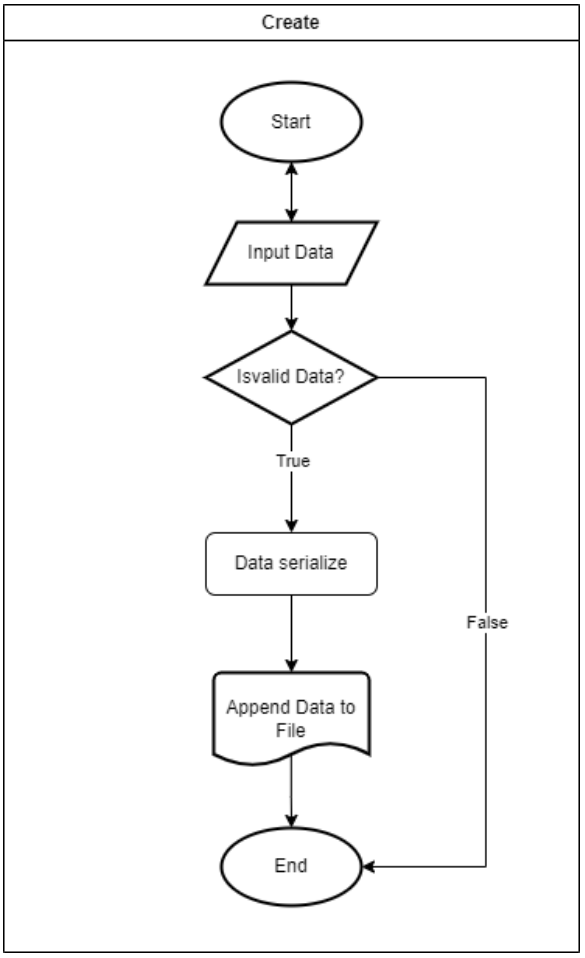
- ContactData 구조체 사용 (wchar_t *, wchar_t *, wchar_t *)
- 자체 라이브러리 사용
- 이벤트 함수들을 함수 포인터 배열을 이용해 쉽게 접근
- char * 대신 wchar_t *를 사용해 유니코드(한글) 지원
- .csv 파일을 사용해 각 컬럼을 콤마(,)로 구분
- db 파일을 utf-8 without BOM으로 설정해 예상치 못한 초기데이터 추출 해결
- Compare 함수와 qsort를 활용해 ContactData* (이름, 전화번호, 주소)데이터 정렬
- 출력되는 데이터는 이름 1순위, 전화번호 2순위로 정렬
- 정렬 및 인덱스 서칭에 이진 탐색을 활용
- 리스트가 아닌 포인터 배열만을 사용해 캐시 데이터의 공간적 지역성 극대화

+ 간단하게 create, update, delete 기능 추가



▼ Create

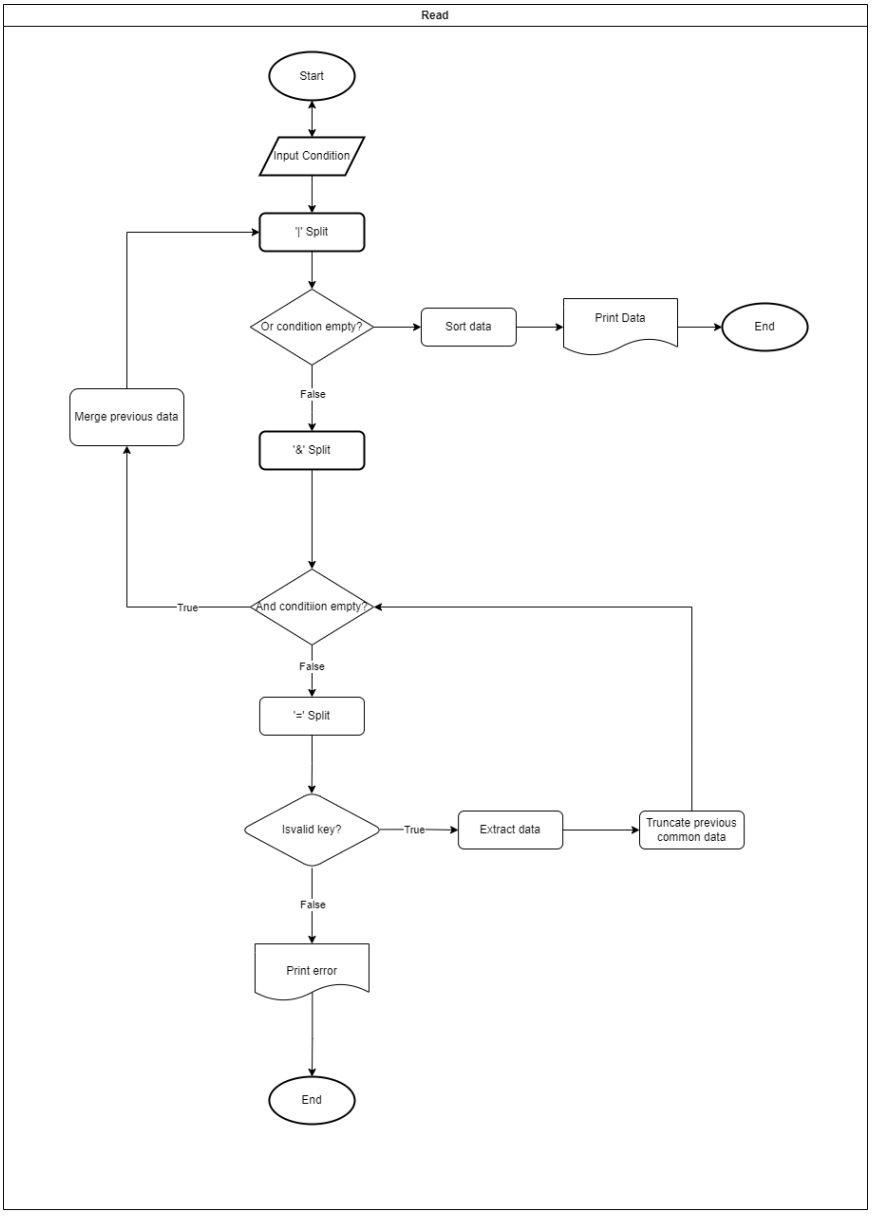
1. 이름, 전화번호, 주소 입력
2. 파일 데이터를 가져와 전화번호 순 정렬
3. 이진 탐색을 이용해 전화번호 중복 체크(중복일 시 종료)
4. 파일 끝에 데이터 추가



▼ **Read**

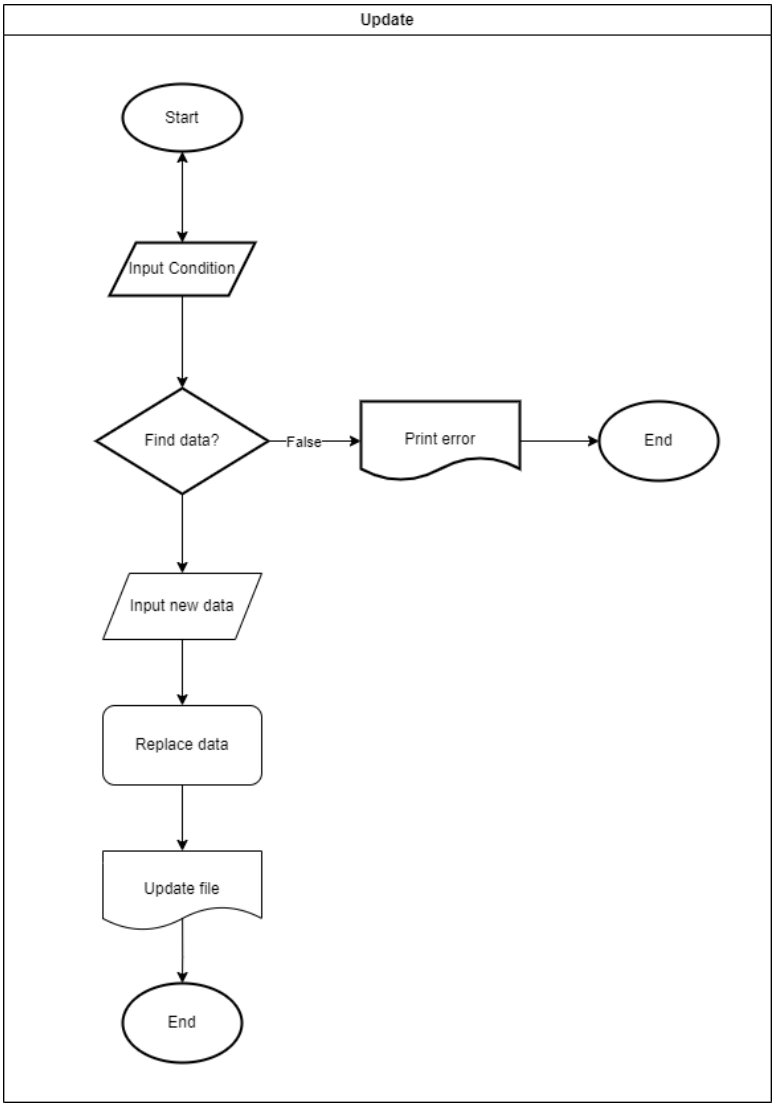
1. 조건 입력
2. or ('|') 기준 Split
3. | Split한 문자열만큼 and 함수 반복
4. 함수 내에서 and ('&') 기준 Split
5. & Split한 문자열만큼 조건 함수 반복
6. 함수 내에서 대입 연산자 ('=') 기준 Split
7. split된 key 유효성 검사 (유요하지 않은 key일 시 에러 출력 후 종료)
8. key와 value에 맞는 데이터를 파일에서 가져옴(없으면 continue)
9. 이진 탐색으로 해당 데이터를 가진 데이터 범위 측정(이름, 주소 중복 가능성 O)
10. 측정된 범위 배열화 후 리턴
11. & 연산에 맞춰서 이전 데이터와 중복되는 데이터만 추출(첫 데이터면 그대로 삽입) 후 반환

12. | 연산에 맞춰서 이전 데이터와 중복되는 데이터 Truncate(중복 안되는 데이터는 그대로 삽입)
13. 3-12 반복
14. 출력 전 오름차순 재정렬(1순위 이름, 2순위 전화번호)



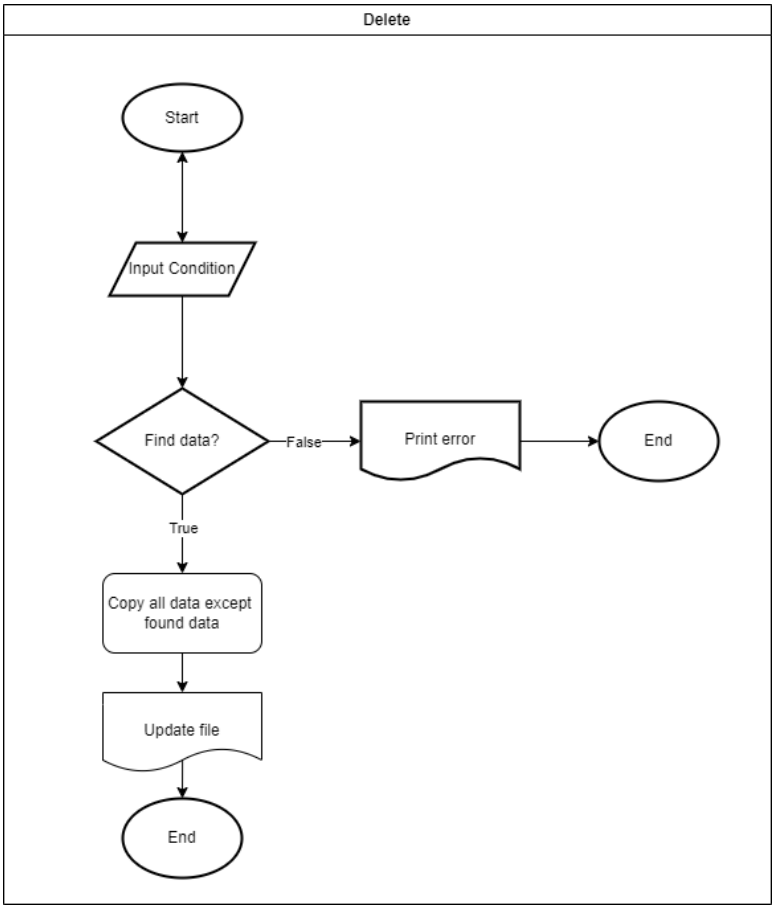
▼ Update

1. 조건 입력(전화번호)
2. 이진 탐색으로 해당 전화번호를 가진 데이터의 인덱스를 추출(없으면 종료)
3. 바꿀 데이터 입력(이름, 전화번호, 주소)
4. 해당 인덱스의 값만 바꾼 후 파일 업데이트



▼ **Delete**

1. 조건 입력(전화번호)
2. 이진 탐색으로 해당 전화번호를 가진 데이터의 인덱스를 추출(없으면 종료)
3. 기존 사이즈 - 1 만큼 할당 후 해당 인덱스인 경우를 제외하고 복사 반복
4. 복사된 데이터로 파일 업데이트



▼ 주요 함수들

```c
int binarySearch(void* arr, int start, int end, void* key, size_t elemSize, int (*func)(void*, void*)) {

    if (start > end)
        return -1;

    int mid = (end + start) / 2;
    void* elem = (char*)arr + (mid * elemSize);
    int funcRet = func(elem, key);
    if (funcRet == 0)
        return mid;
    else if (funcRet > 0)
        return binarySearch(arr, start, mid - 1, key, elemSize, func);
    else
        return binarySearch(arr, mid + 1, end, key, elemSize, func);
}
```

- int binarySearch(서칭할 배열, 시작 인덱스, 마지막 인덱스, 서칭 타겟 객체, 객체의 크기, 비교 함수)
- 이진 탐색 함수로써 값을 검색할 때 사용

```c
void SetContactRange(ContactData* contacts, size_t size, size_t pos, int (*func)(void*, void*), int* start, int* end, ContactData* key) {
    int low = 0, high = pos;

    while (low <= high) {
        int mid = (low + high) / 2;
        if (func(&contacts[mid], key) < 0) low = mid + 1;
        else high = mid - 1;
    }
    *start = low;

    low = pos;
    high = size;

    while (low <= high) {
        int mid = (low + high) / 2;
        if (func(&contacts[mid], key) > 0) high = mid - 1;
        else low = mid + 1;
    }
    *end = high;
}
```

- void SetContactRange(주소록 구조체 배열, 배열의 크기, 기준 인덱스, 비교 함수, *시작 인덱스, *마지막 인덱스)
- 이진 탐색 함수 활용하는 함수로써 주로 read 조건문으로 여러개의 값들을 반환해야할 때 정렬된 배열에서 기준 인덱스를 기준으로 시작 인덱스와 마지막 인덱스를 결정해주는 용도로 사용
  예를 들어
  read 조건문이 “이름 = 다혜” 인 경우 이름은 중복 가능이기에 여러개의 값들이 있을 수 있기에 이름 순으로 정렬시킨 주소록 구조체 배열에서 binarySearch 함수로 찾은 pos값을 기준으로 이름이 다혜로 되어있는 인덱스의 시작과 끝을 정해주는 유틸함수

▼ 참고 자료

https://blockdmask.tistory.com/392

https://woo-dev.tistory.com/232

https://stackoverflow.com/questions/36184019/is-it-possible-force-utf-8-in-a-c-program

https://www.unicode.org/charts/PDF/UAC00.pdf

https://programmerpsy.tistory.com/110

https://learn.microsoft.com/ko-kr/cpp/c-runtime-library/reference/setlocale-wsetlocale?view=msvc-170

https://yoongrammer.tistory.com/75

https://vlasovstudio.com/fix-file-encoding/

https://luckygg.tistory.com/229