

소프트웨어공학 1차 과제



소 속 : 공학 대학

과 목 명 : 소프트웨어공학

학 과 : AI소프트웨어학과

학 번 : 202320078

이 름 : 김세현

제 출 일 : 2024.09.29

목차

- 1. 요구분석 ----- 1p~2p
- 2. 설계 ----- 3p~7p
- 3. 구현코드 설명 -----8p~10p
- 4. 결과분석 ----- 11p~14p

전화번호부의 필수요소

(편리성, 무결성, 사용자친화성)

전화번호부 프로그램은 말 그대로 전화번호를 저장하는 프로그램입니다. 때문에 데이터베이스의 3요소인 무결성, 일관성, 보안성 중 최소한 무결성과 일관성이 성립되어야 합니다.

뿐만 아니라 사용자가 잘못된 값을 입력하거나 중복되는 입력을 할 경우에 이를 바로잡아주는 역할을 하거나, 저장된 데이터를 검색할 때 사용자의 편의성을 높여주는 단어검색 기능과 갑작스러운 종료에도 데이터를 보호하는 기능 등 전자기기에 친숙하지 않은 분들 또한 간편하게 사용할 수 있는 프로그램 이어야 된다고 생각하였습니다.

전화번호부의 필수기능

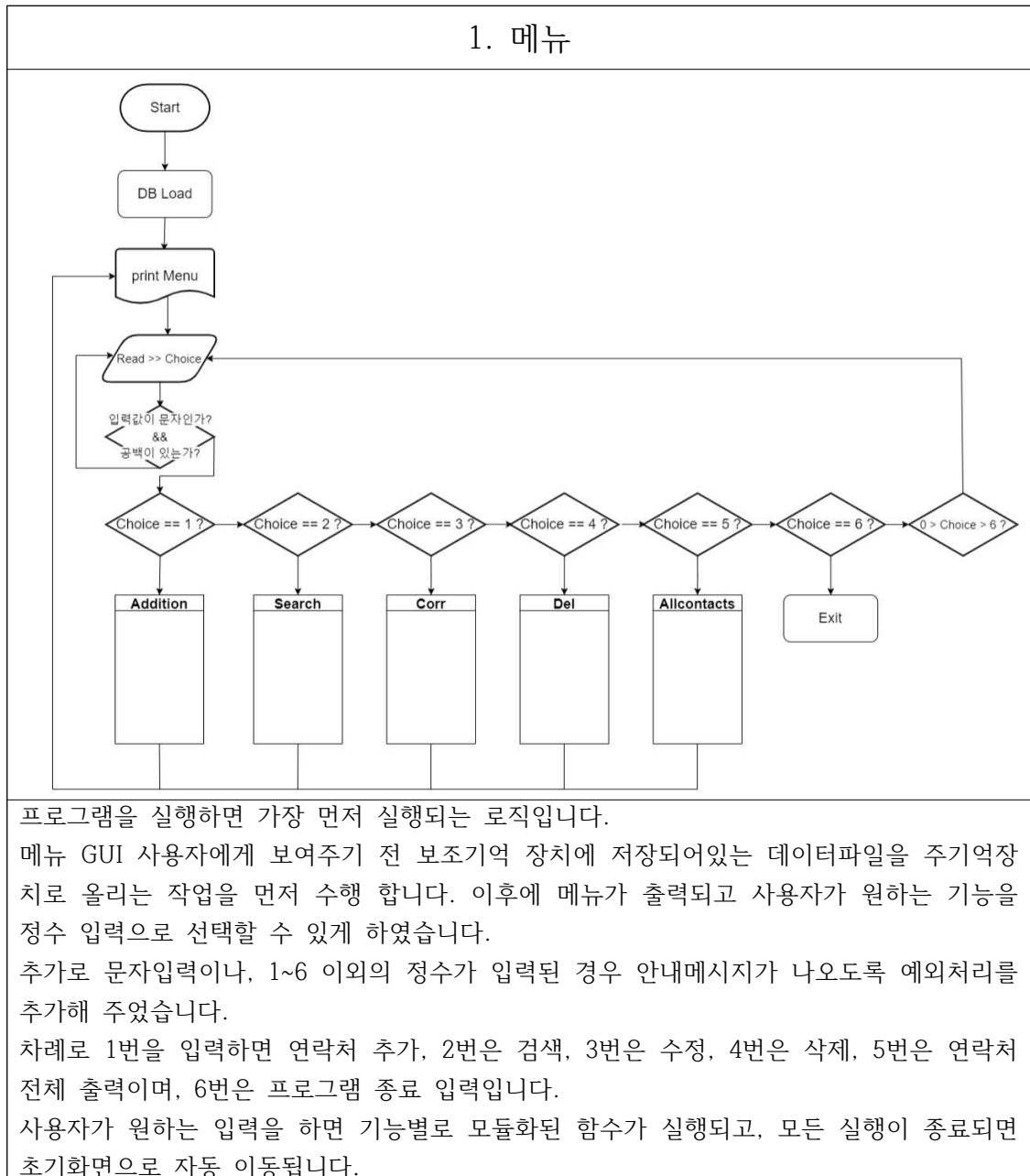
| 데이터 추가 |
|--|
| 사용자가 저장하고 싶은 연락처를 저장하는 기능입니다. 010으로 시작하는 번호와 02 또는 031, 033 등 의 지역번호 모두 저장할수 있어야 하며, 사용자가 중복되는 번호를 입력할시 중복사실을 알려주거나, 입력한 번호에 ‘-’를 자동으로 추가하여 사용자의 편의성을 높여주었습니다. |
| 데이터 검색 |
| 사용자가 저장한 연락처 중에서 검색할 수 있는 기능입니다. 예를 들어 ‘홍길동’이라는 사람의 연락처를 검색할 경우 ‘홍’또는 ‘길동’ 등 특정 단어만 입력해도 검색되게 하였습니다. |

| 데이터 수정 |
|---|
| <p>사용자가 저장한 연락처를 수정하는 기능입니다. 이름수정 기능과 전화번호 수정 기능을 선택할 수 있게 하여 사용자가 원하는 방향으로 수정할 수 있게 하였습니다.</p> |

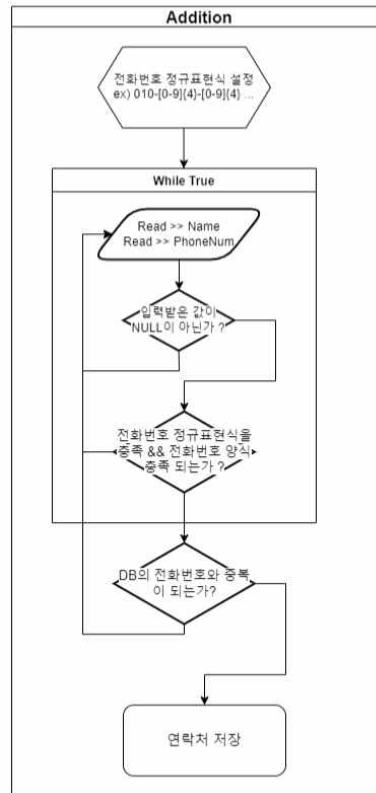
| 데이터 삭제 |
|--|
| <p>데이터 삭제는 데이터 검색과 달리 이름을 정확하게 적어야 작동하게 하였습니다. 뿐만 아니라 이름이 같을 경우가 있을 수 있기 때문에 삭제할 정보를 선택할 수 있는 메뉴를 추가해 주었습니다.</p> |

| 데이터 출력 |
|--|
| <p>연락처 데이터 출력에는 사용자의 편의성을 높이기 위하여 오름차순 정렬한 후 출력되도록 만들었습니다.</p> |

로직설명



2. 연락처 추가



사용자가 메뉴에서 ‘1’을 입력하면 실행되는 함수입니다.

먼저 사용자가 입력할 전화번호의 양식을 두 가지로 나누어 정규표현 식을 만들어 주었습니다. 사용자가 010-1234-1234를 입력할 경우 해당 입력값은 `phoneRegex`에 포함이 되고, 01012341234를 입력할 경우 phoneRegexNoDash 에 포함될 수 있게 하여 사용자의 입력자유도를 높여주었습니다.

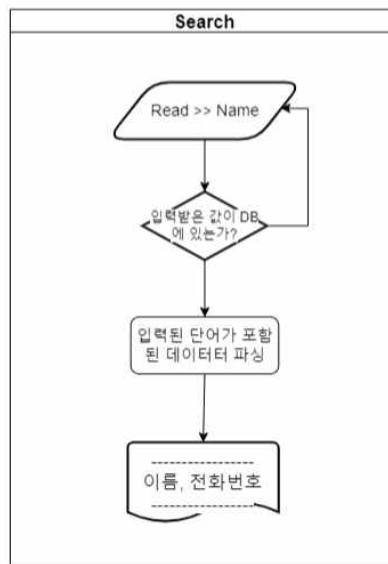
정규 표현 식의 정의를 내린 후 사용자로부터 연락처에 저장할 이름을 입력받습니다.

이때 이름은 영어, 한글, 숫자 모두 가능하도록 해주었습니다. 저의 경우 연락처를 저장할 때 동명이인이 있을 수 있기 때문에 상대방의 소속을 이름 앞에 기재하는 경우가 있습니다.(상대방이 강원대학교 AI 소프트웨어 2학년 홍길동일 경우: “강원대 AI 23 홍길동”) 이처럼 사용자마다 쓰임이 각기 다를 수 있기 때문에 이름의 경우에는 단순 공백을 제외한 모든 문자가 들어갈 수 있도록 해주었습니다.

전화번호는 입력되는 가짓수가 매우 많습니다. 휴대전화에서 송신되는 010으로 시작되는 전화번호뿐만 아니라 서울은 02로 시작되고, 서울을 제외한 모든 지역은 각기 다른 3자리의 지역번호를 가지고 있습니다. 뿐만 아니라 지역번호는 지역번호-3자리-4자리이거나 지역번호-4자리-4자리인 경우도 있기 때문에 모든 경우를 조건문으로 만들어 번호의 총개수가 9자리,10자리,11자리가 이외의 번호는 안내메시지가 나오게 하여 주었으며, ‘-’ 문자가 입력되지 않은 번호는 ‘-’문자를 자동으로 추가해 주는 기능을 추가해주었습니다.

만약 데이터 베이스에 있는 전화번호와 중복이 되면 안내메시지가 나오며 이름 중복은 동명이인이 있을 수 있기 때문에 추가하지 않았습니다.

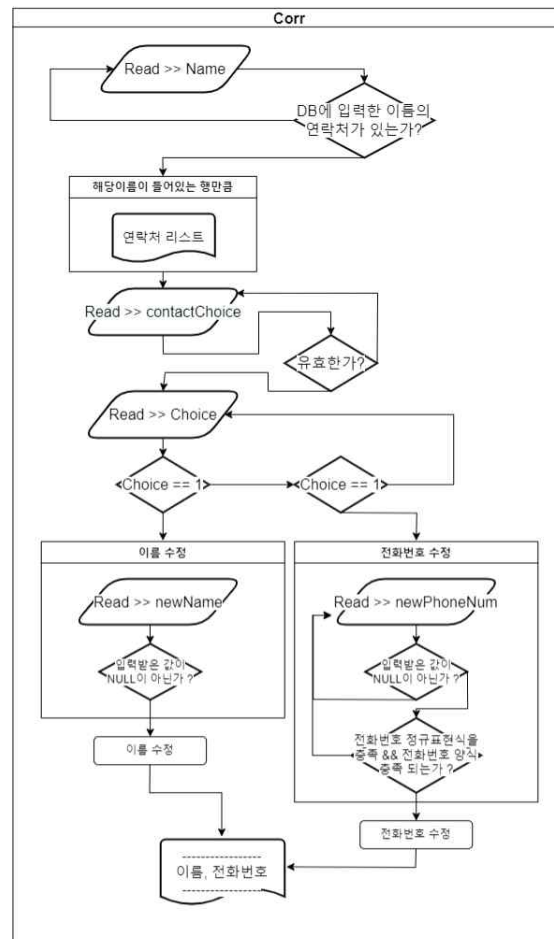
3. 연락처 검색



사용자가 메뉴에서 '2'를 입력하면 실행되는 함수입니다.

연락처 검색은 이름을 이용하여 검색되며, 전체이름중 특정부분만 입력해도 일치하는 항목이 출력되도록 만들었습니다.

4. 연락처 수정



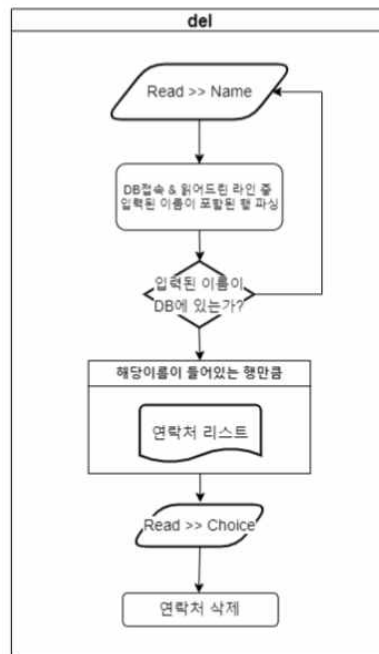
사용자가 메뉴에서 '3'을 입력하면 실행되는 함수입니다.

연락처 검색기능과 달리 수정에서는 사용자가 수정하고 싶은 대상의 이름을 정확하게 입력해야 수정할 수 있습니다. 만약 이름이 다를 경우 안내메시지가 출력됩니다.

같은 이름을 가진 연락처들이 있을 수 있기 때문에 검색되는 연락처의 리스트를 보여주고 사용자가 선택하여 1개의 연락처만 선택되게 하였습니다.

이후 전화번호 변경 또는 이름 변경 두 가지의 선택메시지가 나오며, 메뉴와 마찬가지로 1번을 입력하면 이름수정 기능이 활성화되고, 2번을 입력하면 전화번호 수정기능이 활성화 됩니다. 이름수정은 수정이름 값이 공백일 경우 안내 메시지가 출력되며, 아닐 경우 이름 수정 작업에 들어갑니다. 전화번호 수정은 연락처 추가 기능과 같은 로직을 가지고 있습니다.

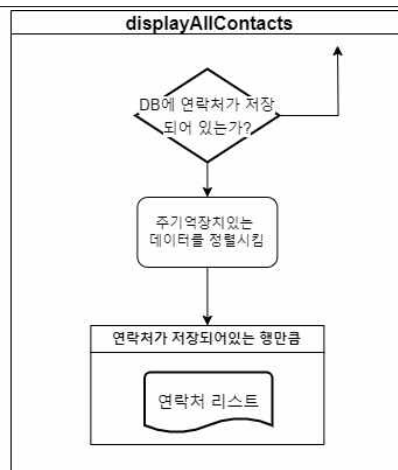
5. 연락처 삭제



사용자가 메뉴에서 '4'를 입력 하면 실행되는 함수입니다.

이 또한 연락처 검색기능과 달리 수정에서는 사용자가 수정하고 싶은 대상의 이름을 정확하게 입력해야 삭제할 수 있습니다. 만약 이름 전체가 같은 데이터가 있는 경우가 있을 수 있기 때문에 검색된 이름 리스트를 출력한 후 원하는 연락처의 번호를 입력하여 선택할 수 있게 하였습니다. 선택된 연락처는 삭제됩니다.

6. 연락처 전체 출력



사용자가 메뉴에서 '5'를 입력 하면 실행되는 함수입니다.

연락처 데이터에 데이터가 없으면 안내메시지가 출력되며, 데이터가 있으면 데이터들을 오름차순 정렬 후 출력됩니다.

구현 코드

데이터 저장 & 로드

file_io.cpp

```
#include "file_io.h"
#include <fstream>

void saveToTxt(const std::vector<std::pair<std::string, std::string>>& phonebook)
{
    std::ofstream file("phonebook_nonjson.txt");
    for (const auto& contact : phonebook) {
        file << contact.first << "," << contact.second << "\n";
    }
}

void loadFromTxt(std::vector<std::pair<std::string, std::string>>& phonebook)
{
    std::ifstream file("phonebook_nonjson.txt");
    if (file.is_open()) {
        std::string line;
        while (std::getline(file, line)) {
            size_t commaPos = line.find(",");
            if (commaPos != std::string::npos) {
                std::string name = line.substr(0, commaPos);
                std::string phoneNumber = line.substr(commaPos + 1);
                phonebook.push_back(std::make_pair(name, phoneNumber));
            }
        }
    }
}
```

file_io.h

```
#ifndef FILE_IO_H
#define FILE_IO_H

#include <vector>
#include <string>

void saveToTxt(const std::vector<std::pair<std::string, std::string>>& phonebook);
void loadFromTxt(std::vector<std::pair<std::string, std::string>>& phonebook);

#endif
```

file_io 모듈은 데이터 파일의 입출력을 담당하고 있는 함수를 가지고 있습니다. 헤더파일에서 함수를 정의한 후 소스파일에 기능소스를 담아두었습니다. 데이터를 저장할 때는 ','를 사용하여 이름과 전화번호를 구분해주었고, 사용자가 입력한 데이터를 보조기억 장치에 있는 데이터파일에 저장하는 역할을 합니다. 프로그램을 처음 실행할 때 자동적으로 실행되는 데이터 로드 함수의 경우 보조기억 장치에 저장되어있는 정보를 주기억장치로 올려주는 역할을 합니다. 이때 C++의 STL라이브러리의 벡터 방식을 이용하여 데이터를 주기억장치로 올리게 하여주었습니다. 동적 배열의 장점과 인덱스를 사용하여 요소에 빠르게 접근할 수 있다는 장점을 가지고 있기 때문에 벡터로 구현하였습니다.

전화번호 유효성 검사 & '-' 추가 코드

addition.cpp (일부분)

```
std::string name, phoneNumber;
std::regex phoneRegex("^(010-[0-9]{4}-[0-9]{4}|02-[0-9]{3,4}-[0-9]{4}|[0-6]{3}-[0-9]{3,4}-[0-9]{4})$");
std::regex phoneRegexNoDash("^(010[0-9]{8}|02[0-9]{7,8}|[0-6]{3}[0-9]{7,8})$");

// 전화번호 입력
while (true)
{
    std::cout << "전화번호 (해외 번호의 경우 '-'와 함께 입력): ";
    std::getline(std::cin, phoneNumber);

    if (phoneNumber.empty()) {
        std::cout << "전화번호가 비어 있습니다. 다시 입력해 주세요." << std::endl;
    }
    else if (std::regex_match(phoneNumber, phoneRegexNoDash)) { // - 추가
        if (phoneNumber.substr(0, 2) == "02" && phoneNumber.length() == 9) { // 서울지역번호 2자리-3자리-4자리
            phoneNumber = phoneNumber.substr(0, 2) + "-" + phoneNumber.substr(2, 3) + "-" + phoneNumber.substr(5, 4);
        }
        else if (phoneNumber.substr(0, 2) == "02" && phoneNumber.length() == 10) { // 서울지역번호 2자리-4자리-4자리
            phoneNumber = phoneNumber.substr(0, 2) + "-" + phoneNumber.substr(2, 4) + "-" + phoneNumber.substr(6, 4);
        }
        else if ((phoneNumber.substr(0, 3) == "031" || phoneNumber.substr(0, 3) == "032" || phoneNumber.substr(0, 3) == "042" ||
            phoneNumber.substr(0, 3) == "051" || phoneNumber.substr(0, 3) == "052" || phoneNumber.substr(0, 3) == "058" ||
            phoneNumber.substr(0, 3) == "062" || phoneNumber.substr(0, 3) == "064" || phoneNumber.substr(0, 3) == "038" ||
            phoneNumber.substr(0, 3) == "041" || phoneNumber.substr(0, 3) == "043" || phoneNumber.substr(0, 3) == "054" ||
            phoneNumber.substr(0, 3) == "055" || phoneNumber.substr(0, 3) == "061" || phoneNumber.substr(0, 3) == "063") &&
            phoneNumber.length() == 10) { // 지역번호 3자리-3자리-4자리
            phoneNumber = phoneNumber.substr(0, 3) + "-" + phoneNumber.substr(3, 3) + "-" + phoneNumber.substr(6, 4);
        }
        else if ((phoneNumber.substr(0, 3) == "031" || phoneNumber.substr(0, 3) == "032" || phoneNumber.substr(0, 3) == "042" ||
            phoneNumber.substr(0, 3) == "051" || phoneNumber.substr(0, 3) == "052" || phoneNumber.substr(0, 3) == "058" ||
            phoneNumber.substr(0, 3) == "062" || phoneNumber.substr(0, 3) == "064" || phoneNumber.substr(0, 3) == "038" ||
            phoneNumber.substr(0, 3) == "041" || phoneNumber.substr(0, 3) == "043" || phoneNumber.substr(0, 3) == "054" ||
            phoneNumber.substr(0, 3) == "055" || phoneNumber.substr(0, 3) == "061" || phoneNumber.substr(0, 3) == "063") &&
            phoneNumber.length() == 11) { // 지역번호 3자리-4자리-4자리
            phoneNumber = phoneNumber.substr(0, 3) + "-" + phoneNumber.substr(3, 4) + "-" + phoneNumber.substr(7, 4);
        }
        else if (phoneNumber.substr(0, 3) == "010" && phoneNumber.length() == 11) {
            phoneNumber = phoneNumber.substr(0, 3) + "-" + phoneNumber.substr(3, 4) + "-" + phoneNumber.substr(7, 4); // 휴대전화번호 010-4자리-4자리
        }
        else {
            std::cout << "잘못된 전화번호 형식입니다. 다시 입력해 주세요." << std::endl;
            continue;
        }
        break;
    }
    else if (std::regex_match(phoneNumber, phoneRegex)) { // - 있는 경우 패스
        break;
    }
    else {
        std::cout << "잘못된 전화번호 형식입니다. 전화번호는 010-xxxx-xxxx 또는 010-xxxx-xxxx 이거나 지역전화의 경우 (지역번호)-(x)xxx-xxxx, (지역번호)(x)xxx-xxxx 형식입니다." << std::endl;
    }
}
```

addition.h

```
#ifndef ADDITION_H
#define ADDITION_H

#include <vector>
#include <string>

void addition(std::vector<std::pair<std::string, std::string>>& phonebook);

#endif // ADDITION_H
```

사용자가 전화번호를 입력했을 때 유효한 번호인지 판별하는 코드입니다.

정규표현 식을 사용하여 사용자가 입력한 값이 '-'이 입력된 문자인지 아닌지 판별해줍니다. 만약 사용자가 '-'문자를 입력하지 않고 전화번호를 입력했을 시 전화번호의 첫 번째 값(ex: 010 or 031 or 02 or etc..)과 전체문자의 수를 이용하여 입력값이 해당하는 조건을 찾아서 '-'문자를 삽입시켜 줍니다.

사용자가 '-'문자와 함께 전화번호를 입력했을시 입력한 자릿수만 확인한 후 데이터 저장에 들어갑니다.

만약 사용자가 전화번호를 입력하지 않거나, 중복되는 전화번호를 입력하거나, 전화번호 조건에 맞지 않는 번호를 입력할새 안내메시지와 함께 반환됩니다.

데이터 수정 (요소 접근 코드)

correction.cpp (일부분)

```
std::vector<int> indices;
for (size_t i = 0; i < phonebook.size(); ++i) {
    if (phonebook[i].first == name) {
        indices.push_back(static_cast<int>(i)); // 수정된 부분
    }
}

if (indices.empty()) {
    std::cout << "연락처를 찾을 수 없습니다." << std::endl;
    return;
}

std::cout << "=== 검색된 연락처 ===" << std::endl;
for (size_t i = 0; i < indices.size(); ++i) {
    std::cout << i + 1 << ", 이름: " << phonebook[indices[i]].first << ", 전화번호: " << phonebook[indices[i]].second << std::endl;
}

int contactChoice;
std::cout << "수정할 연락처의 번호를 입력하세요: ";
std::cin >> contactChoice;

if (contactChoice < 1 || contactChoice > static_cast<int>(indices.size())) {
    std::cout << "잘못된 선택입니다. 다시 시도하세요." << std::endl;
    return;
}

int index = indices[contactChoice - 1];

int choice;
std::cout << "=== 수정하고 싶은 항목을 선택해 주세요 ===" << std::endl;
std::cout << "1. 이름 수정" << std::endl;
std::cout << "2. 전화번호 수정" << std::endl;
std::cout << "숫자를 입력하여 선택: ";
std::cin >> choice;

switch (choice) {
case 1:
    modifyName(phonebook, index); // 수정된 부분
    break;
case 2:
    modifyPhoneNumber(phonebook, index); // 수정된 부분
    break;
default:
    std::cout << "잘못된 선택입니다. 다시 시도하세요." << std::endl;
    break;
}

// 이름을 수정하는 함수
void modifyName(std::vector<std::pair<std::string, std::string>>& phonebook, int index) {
    std::string newName;
    std::cout << "새 이름을 입력하세요: ";
    std::cin.ignore();
    std::getline(std::cin, newName);

    phonebook[index].first = newName;
    std::cout << "이름이 수정되었습니다." << std::endl;
}
```

correction.h

```
#ifndef CORR_H
#define CORR_H

#include <vector>
#include <string>


void corr(std::vector<std::pair<std::string, std::string>>& phonebook);
void modifyName(std::vector<std::pair<std::string, std::string>>& phonebook, int index);
void modifyPhoneNumber(std::vector<std::pair<std::string, std::string>>& phonebook, int index);

#endif
```

사용자가 데이터를 수정할 때 실행되는 코드입니다. 벡터의 장점인 간편한 요소 접근방식을 이용하여 사용자가 선택한 데이터에 접근하고 이를 수정하는 방식을 가지고 있습니다.

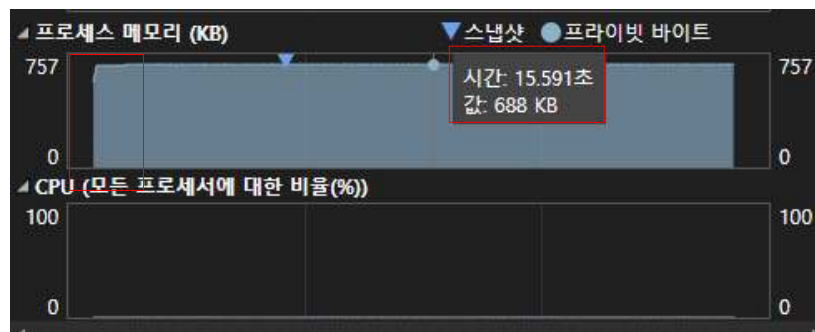
테스트 결과

| | |
|--|---|
| <p>메뉴 UI</p> <pre> =====메뉴 선택===== 1. 연락처 추가 2. 연락처 검색 3. 연락처 수정 4. 연락처 삭제 5. 연락처 전체 보기 6. 프로그램 종료 ===== 번호를 입력하세요: </pre> | <p>> 1번 - 연락처 추가</p> <pre> 이름: 김세현 전화번호(해외번호의 경우 '-'와 함께 입력): 01042387086 추가 완료! 메뉴로 이동하려면 아무 키나 누르세요... </pre> <p>> enter 입력 후 다시 메뉴로 진입</p> |
| <p>> 2번 - 연락처 검색</p> <pre> 검색할 이름을 입력하세요: 김 ----- 이름: 김세현, 전화번호: 010-2222-2222 ----- 이름: 김세현, 전화번호: 010-4238-7086 ----- 메뉴로 이동하려면 아무 키나 누르세요... </pre> <p>기존에 입력되었는 값과 같이 출력된걸 볼 수 있다. 뿐만 아니라 '-'를 입력하지 않아도 기입되어있다.</p> <p>> enter 입력 후 다시 메뉴로 진입</p> | <p>> 3번 - 연락처 수정</p> <pre> 수정할 연락처의 이름: 김세현 ==== 검색된 연락처 ==== 1. 이름: 김세현, 전화번호: 010-2222-2222 2. 이름: 김세현, 전화번호: 010-4238-7086 수정할 연락처의 번호를 입력하세요: </pre> <p>> 수정하고 싶은 연락처 선택</p> |
| <p>>1번 연락처 선택</p> <pre> 수정할 연락처의 번호를 입력하세요: 1 ==== 수정하고 싶은 항목을 선택해 주세요 ==== 1. 이름 수정 2. 전화번호 수정 숫자를 입력하여 선택: </pre> <p>> 수정하고 싶은 요소 선택</p> | <p>> 2번 - 전화번호 수정 선택</p> <pre> ==== 수정하고 싶은 항목을 선택해 주세요 ==== 1. 이름 수정 2. 전화번호 수정 숫자를 입력하여 선택: 2 새 전화번호를 입력하세요: 0315777086 전화번호가 수정되었습니다. 메뉴로 이동하려면 아무 키나 누르세요... </pre> <p>> 경기도 지역번호로 수정</p> <p>> enter 입력 후 다시 메뉴로 진입</p> |

| | |
|---|---|
| <p>> 5번 - 연락처 전체 출력</p> <pre>===== 번호를 입력하세요 : 5 ===== 이름 : 김세현 , 전화번호 : 031-577-7086 ===== 이름 : 김세현 , 전화번호 : 010-4238-7086 ===== 이름 : 서울2 , 전화번호 : 02-138-8289 ===== 메뉴로 이동하려면 아무 키나 누르세요...</pre> <p>정렬된후 출력된 모습을 볼수있으며, 수정하였던 전화번호와 추가한 전화번호 모두 볼 수 있다.</p> | <p>> 4번 - 연락처 삭제</p> <pre>삭제할 연락처의 이름 : 김세현 ===== 1. 이름 : 김세현 , 전화번호 : 031-577-7086 2. 이름 : 김세현 , 전화번호 : 010-4238-7086 ===== 삭제할 연락처의 번호를 입력하세요 : 2 삭제 완료. 메뉴로 이동하려면 아무 키나 누르세요...</pre> <p>수정때와 마찬가지로 삭제할 데이터를 선택할수 있다. > enter 입력 후 다시 메뉴로 진입</p> |
| <p>> 5번 - 연락처 전체 출력</p> <pre>3. 연락처 수정 4. 연락처 삭제 5. 연락처 전체 보기 6. 프로그램 종료 ===== 번호를 입력하세요 : 5 ===== 이름 : 김세현 , 전화번호 : 031-577-7086 ===== 이름 : 서울2 , 전화번호 : 02-138-8289 ===== 메뉴로 이동하려면 아무 키나 누르세요...</pre> <p>삭제가 완료된걸 볼 수 있다.</p> | <p>> 6번 - 종료</p> <pre>=====메뉴 선택===== 1. 연락처 추가 2. 연락처 검색 3. 연락처 수정 4. 연락처 삭제 5. 연락처 전체 보기 6. 프로그램 종료 ===== 번호를 입력하세요 : 6 프로그램을 종료합니다. D:\과제 프로젝트\전화번호부(최종)\x64\Debu . 디버깅이 중지될 때 콘솔을 자동으로 닫으려 하도록 설정합니다. 이 창을 닫으려면 아무 키나 누르세요... </pre> |
|  <p>(프로그램 종료후 데이터 파일)</p> | <pre>5. 연락처 전체 보기 6. 프로그램 종료 ===== 번호를 입력하세요 : 5 ===== 이름 : 김세현 , 전화번호 : 031-577-7086 ===== 이름 : 서울2 , 전화번호 : 02-138-8289 ===== 메뉴로 이동하려면 아무 키나 누르세요...</pre> <p>(프로그램을 다시 실행한후 전체조회 결과) 데이터가 유실되거나 변형되지 않을거 볼 수 있다.</p> |

마치며 & 결과분석

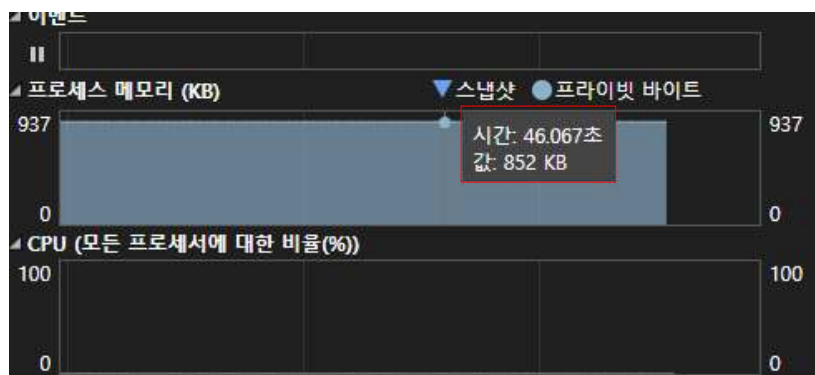
전화번호부인 만큼 하드웨어 자원을 최소한으로 사용해야 한다고
생각했습니다. 그래서 릴리즈 후 램 사용량을 측정 해보았습니다.



데이터파일에 임의의 연락처 111개를 저장한후 측정한 값입니다.
데이터 로드 후 급속하게 자원사용량이 증가되지만 688Kb 정도
사용하는걸로 나왔습니다. 이후 연락처를 1004개 추가 후 다시
측정해보았습니다.

서울,01-126-8181
경기3,031-577-2676

줄 1004, 열 17 | 15,105/15,105자



측정 결과 852Kb로 다소 높게 나왔지만 데이터의 양을 생각하면
적당하다고 생각하였습니다.

다만 사용자 편의성은 다소 아쉬움이 들었습니다.

프로젝트 초기에는 커맨드 창에서 작동되는 프로그램이 아닌 실제
서비스에서도 사용할 수 있는 GUI를 구현할 계획이었지만, 예상보다
너무 프로젝트가 커질 거 같아 포기한 점이 아쉬움 중 하나입니다.
다음 과제에서는 GUI를 구현해 보거나, 조금 더 신경 써서 작업할
생각입니다.

읽어주셔서 감사합니다.