**전화번호부 프로그램**

- 최종 보고서 –



|  |  |
| --- | --- |
| 강 의 명 | 소프트웨어공학 |
| 교 수 님 | 권세진 교수님 |
| 과 제 물 | 4주차(~9/27) |
| 학 과 | 소프트웨어미디어융합전공 |
| 학 번 | 201923220 |
| 성 명 | 김효정 |
| 제 출 일 자 | 2020. 9. 26 |

**목차**

**Ⅰ. 프로그램 계획 및 요구 분석**

-1) 전화번호부 프로그램 설명 1

-2) 전화번호부 프로그램 기능 정의 및 설명 2



**Ⅱ. 프로그램 설계**

-1) 전화번호부 프로그램 도면화 3

-2) 구조 설명 4-7

**Ⅲ. 프로그램 구현**

-1) 주요 함수 및 기능 설명 7-13

**Ⅳ. 프로그램 테스트 결과**

-1) 기능별 구현 결과 13-16

-2) 프로그램 분석 16-18

**>> 들어가기에 앞서**

이번 첫 프로젝트는 전화번호부 프로그램을 만들어 데이터를 저장하고 검색, 수정 및 삭제의 기능을 구현하는 프로그램입니다. 이번 프로그램을 계획하고 정의하는 과정에서 정보를 처리하는 다양한 표현 방법들을 생각하게 되었습니다. 이런 표현 방법들에 대한 기능을 정의하였고 구체적으로 내용을 설명하였습니다. 다음으로 설계 과정에서는 ‘flow chart’를 이용하여 전반적인 프로그램 실행 모습을 표현하였습니다. 순서도를 통해 큰 흐름과 각 기능이 수행되는 흐름을 설명하였습니다.

**Ⅰ. 프로그램 계획 및 요구 분석**

**Ⅰ- 1) 전화번호부 프로그램 설명**

일반적인 전화번호부 프로그램은 이름, 전화번호, 주소, 나이, 이메일, 그룹, 메모 등의 한 사람에 대한 정보를 입력 받아 이름(ㄱ, ㄴ, ㄷ, …) 순으로 정리를 해 놓은 시스템을 말합니다. 전화번호부 프로그램에서 실행되는 기능은 새로운 인물에 대한 정보의 추가, 인물에 대한 정보 삭제 및 검색, 정보 수정 및 전화번호부에 저장되어 있는 사람들의 정보가 출력되는 것이 주 기능입니다.

이번 전화번호부 프로그램은 한 사람에 대한 정보로 이름, 전화번호, 나이, 메모를 입력 받도록 설정하여 하나의 연락처를 이룹니다. 입력된 정보는 전화번호부 리스트에 추가가 됩니다. 이렇게 추가된 정보에 대한 검색 및 삭제의 기능이 구현되며 정보에 접근성을 높이기 위해서 ‘이름’을 기준으로 하여 정보를 검색하고 삭제하여 사용자가 실행한 전화번호부를 만들게 됩니다. 출력은 이용자가 보다 보기 편하게 하기 위하여 이름순에 따라 정보가 정렬되어 나옵니다.

**Ⅰ- 2) 전화번호부 프로그램 기능 정의 및 설명**

전화번호부 프로그램의 기능은 크게 5가지로 구분됩니다. 구분되는 기능으로는 정보의 구성 및 추가, 정보 검색, 정보 삭제, 전화번호부 출력, 프로그램 종료로 나뉩니다.

**① 정보의 구성**

기본적으로 이 전화번호부에서 갖는 정보는 phonebook 클래스를 통해 개인 객체를 생성하며 개인 연락처를 담는 공간입니다. 연락처를 형성하는 이름과 전화번호는 string으로 설정하여 접근 용이성을 높입니다. 나이는 int형으로 설정하고, 메모 또한 string으로 하여 정보의 입, 출력, 접근성을 높입니다. 더불어 **AddbySorted()** 함수를 통해 정보를 입력 받고 입력된 정보가 전화번호부에 추가됩니다. 동시에 **AddbySorted()**에 정보가 입력됨과 동시에 이름을 비교하게 하여 이름 순으로 정렬이 됩니다.

|  |  |  |
| --- | --- | --- |
| **구 성** | **변 수 타 입** | **비 고** |
| 이 름 (name) | String | 공백 미포함 |
| 전 화 번 호 (num) | String | 기호 ‘-‘ 포함 |
| 나 이 (age) | Int | 정수형 숫자만 입력 |
| 메 모 (memo) | String | 공백 미포함 |

**② 정보 검색**

검색에서는 찾으려는 정보의 이름을 입력하면 그 사람에 대한 이름, 전화번호, 나이, 메모의정보가 출력됩니다. 검색은 **SearchData()**를 통해 이루어집니다. 이름을 입력 받으면 리스트를 탐색하여 해당하는 이름에 접근합니다. 리스트의 정보에 접근하여 그에 대응하는 이름, 전화번호, 나이, 메모가 출력됩니다. 이미 연결되어 있는 정보 리스트에서 새로운 탐색 헤드를 만들어 검색이 됩니다..

**③ 정보 삭제**

삭제에서도 마찬가지로 이름을 기준으로 작동이 됩니다. 지우려고 하는 정보의 이름을 입력하면 그 사람에 대한 이름, 전화번호, 나이, 메모에 대한 내용이 지워집니다. 지우려는 이름을 입력 받아 **Delete()**를 호출하면 리스트에 저장된 동일한 이름을 찾게 실행됩니다. 이름 탐색을 통해 해당하는 객체의 정보를 찾으면 그 정보에 대한 모든 내용이 지워집니다. “연락처가 삭제되었습니다.”라는 삭제 완료 문구가 출력되어 삭제가 되었는지 확인이 됩니다..

**④ 전화번호부 출력**

이렇게 정리된 전화번호부는 **PrintList()**를 통해 출력됩니다. 이미 이름순으로 정렬이 되었기 때문에 리스트에 순차적으로 접근하여 출력됩니다. 기존 리스트에 접근하기 위해 새로운 임시 헤드로 정보에 접근이 됩니다. 전화번호부의 출력은 한 사람의 정보인 이름 전화번호 나이 메모가 한 줄에 하나씩 출력됩니다.

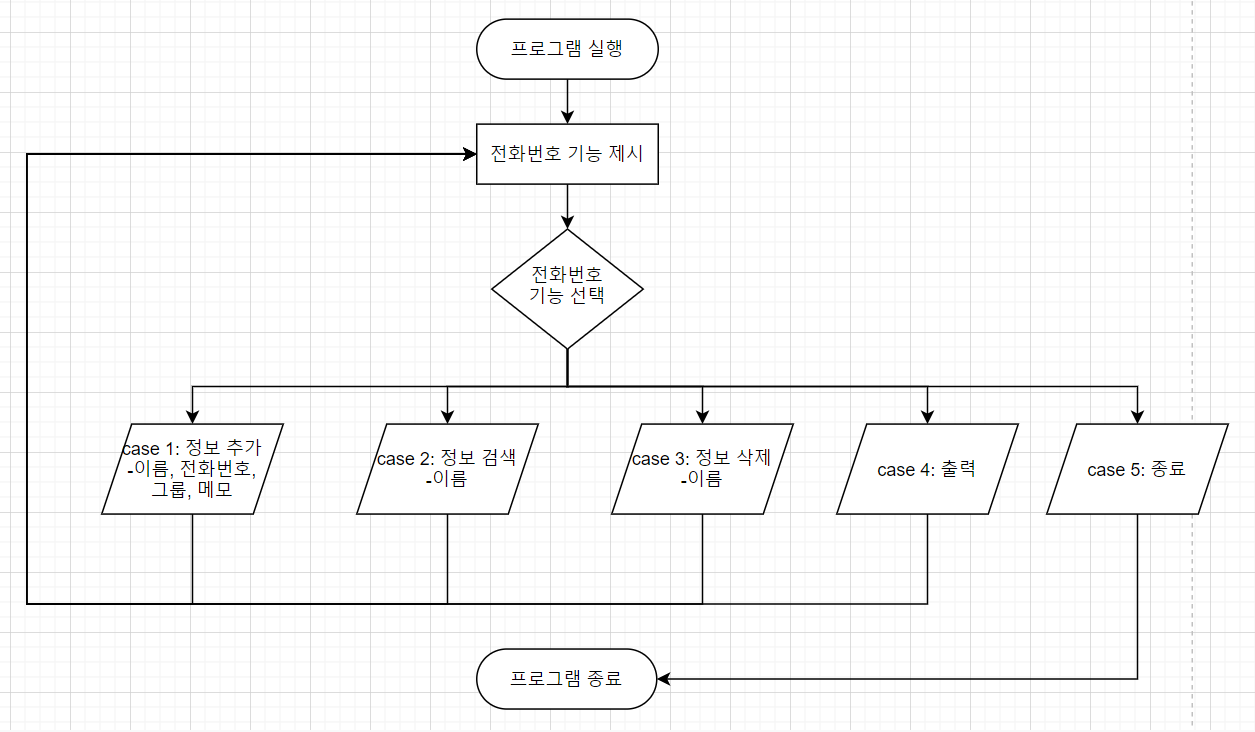
**⑤ 프로그램 종료**

전화번호부 프로그램은 정보의 추가, 검색, 삭제, 출력을 반복하여 작동됩니다. 반복문을 통해 기능을 여러 번 수행이 됩니다. 프로그램 작동이 멈추는 종료 기능을 추가하여 사용자가 종료를 원할 시 ‘종료하시겠습니까?’를 통해 프로그램이 종료됩니다.

**Ⅱ. 프로그램 설계**

**Ⅱ - 1) 전화번호부 프로그램 도면화**

전반적인 전화번호부 프로그램의 실행 과정은 아래의 순서도(flow chart)와 같이 작동이 됩니다.





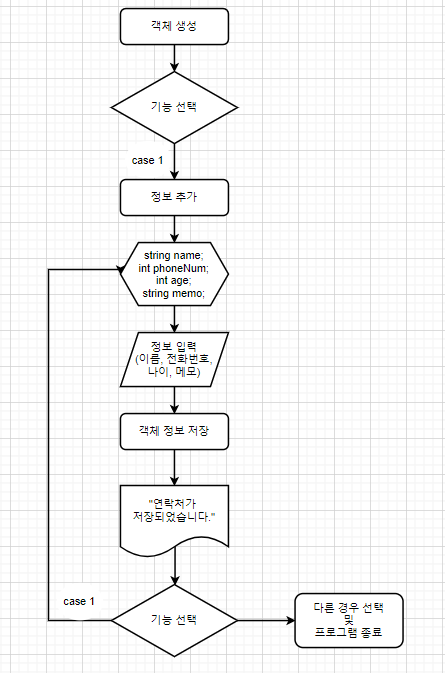
나이

이 전화번호부 프로그램은 프로그램이 실행되면 전화번호부 기능이 제시가 됩니다. “연락처 추가는 1번, 연락처 검색은 2번, 연락처 삭제는 3번, 출력은 4번, 종료는 4번을 누르세요”와 같이 기능 선택의 문구가 제시됩니다. 사용자가 해당하는 기능을 선택하면 선택한 번호의 기능이 작동하게 됩니다. 위 그림과 같이 첫 번째 경우는 정보 추가가, 두 번째 경우는 연락처 검색이, 세 번째 경우는 연락처 삭제, 네 번째 경우는 전화번호부 출력, 그리고 종료가 되는 경우가 선택에 따라 실행됩니다. 이 실행은 반복되어 작동하며, 종료 기능을 선택할 시 프로그램이 종료가 됩니다. 따라서 전반적인 흐름은 경우에 따라 기능이 작동하는 것으로 구성하였습니다.

**Ⅱ - 2) 구조 설명**

이 전화번호부 프로그램의 구조를 세부적으로 나누어 보면 다음 순서도들과 같은 기능을 가집니다. 세부적인 기능에는 전화번호부 기능 제시 및 기능 선택 부분, 연락처 추가 부분, 연락처 검색 부분, 연락처 삭제 부분, 전화번호부 출력 부분, 종료의 6개 기능이 구현됩니다. 아래 순서도는 프로그램이 실행되고 난 후를 가정으로 하여 프로그램 실행 및 기능 제시에 대한 과정은 **①** 순서도에서만 보여집니다.

**① 전화번호부 기능 제시 및 기능 선택, ② 연락처 추가**



**AddbySorted()**

**객체 정보 – phonebook**

**Private – node의 head public – 함수 및 변수**

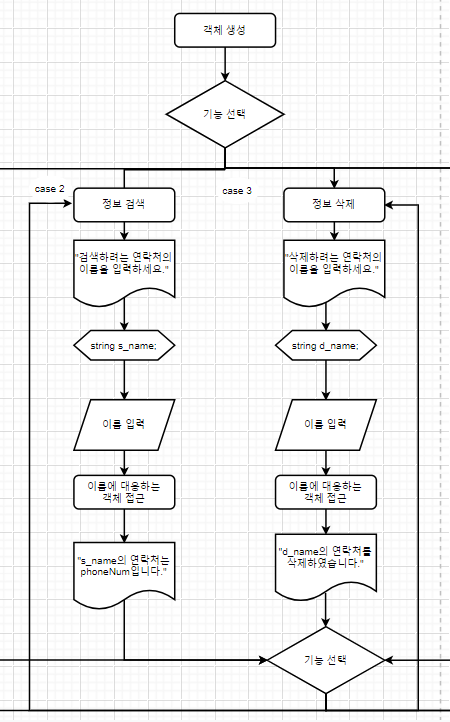
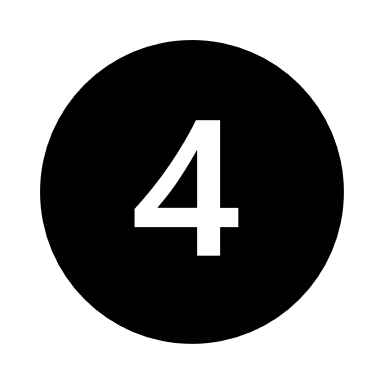
**끝(공통)**

**main.cpp시작**

프로그램이 실행되면 phoneBook클래스의 객체가 생성됩니다. 객체의 정보는 public으로 하여 접근의 용이성이 높아집니다. 그 외의 정보(변수) 및 기능(리스트 연결 class와 정보 처리 기능 함수들)도 public으로 설정하여 호환이 편하게 이루어집니다. 특히 template 형식으로 접근성이 높아집니다. 노드는 private으로 하여 쉽게 변형되지 않게 구성됩니다.

정보를 추가하는 번호를 입력하면 AddbySorted()가 실행되며 “연락처를 추가합니다”라는 문구가 출력됩니다. 순서에 따라 정보를 입력하면 객체에 정보가 저장이 되며 추가됩니다.

**③ 연락처 검색, ④연락처 삭제**



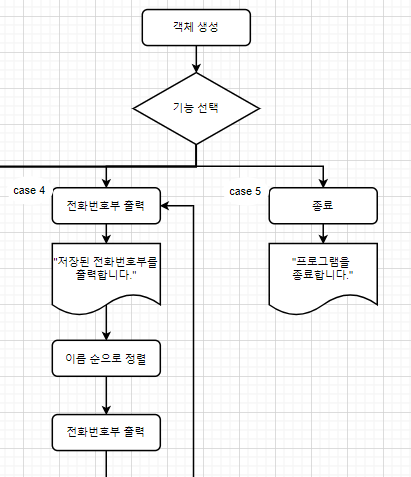
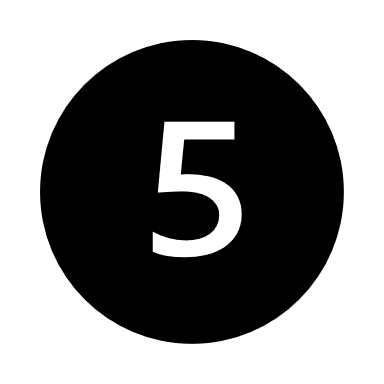
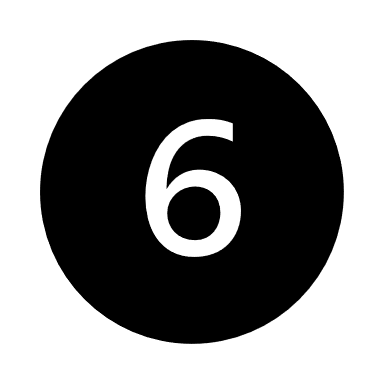
**Delete()**

**SearchData()**

연락처를 검색하는 기능의 번호를 누르면 SearchData()가 구현되며 입력 받은 이름에 대응하는 객체를 탐색합니다. 이중연결리스트로 저장된 정보에서 입력 받은 이름과 같은 이름을 찾아 정보를 불러 출력이 됩니다.

연락처를 삭제하는 기능도 마찬가지로 해당 번호를 입력하면 Delete()가 입력 받은 이름을 받고, 이에 대응하는 객체의 정보로 접근하여 삭제하게 됩니다. 삭제 후에 삭제 완료문을 출력하여 삭제가 되었음을 알립니다. 이때, 삭제된 데이터 뒤에 있는 정보들을 다시 연결하여 연결리스트 형식을 유지하게 합니다.

**⑤ 전화번호부 출력, ⑥프로그램 종료**



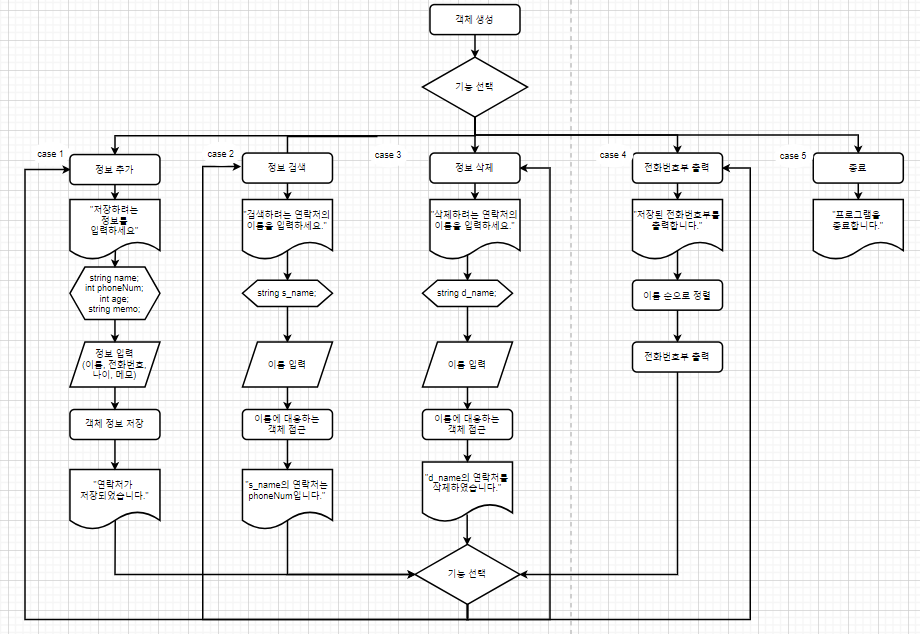
**PrintList()**

전화번호부를 출력하려면 해당 기능의 번호를 입력하여 전화번호부가 출력이 됩니다. 추가하면서 이름순으로 정렬이 되었기 때문에 바로 PrintList()를 통해 정렬된 내용이 출력됩니다. 처음 저장된 부분의 리스트부터 하나씩 넘겨 객체에 들어있는 모든 정보(이름, 전화번호, 나이, 메모)가 출력됩니다. 필요에 따라 printInfo()를 만들어 출력을 시각성이 좋게 담당하는 기능을 따로 만듭니다.

이러한 기능은 재생산성이 있어 계속 반복이 됩니다. while(true)와 같은 반복문을 통해프로그램이 지속적으로 실행되게 합니다. 그러다 사용자가 프로그램을 종료하기를 원하면, 즉 프로그램의 실행을 멈추려고 한다면 해당 번호를 입력하여 반복을 멈추어 프로그램이 종료됩니다.

따라서 이 전화번호부 프로그램은 정보의 추가, 검색, 삭제, 출력이 반복되게 수행되도록 설계된 프로그램입니다. 이렇게 무한히 실행되는 프로그램에 종료 기능을 넣어 반복문이 종료되는 순간 프로그램의 실행이 종료됩니다.

아래의 그림은 전반적인 순서도의 흐름에서 기능을 더욱 세분화하고 구체적인 설계를 표현한 순서도입니다. 기능 선택 시 출력되는 문구나 변수들을 간단하게 나타낸 흐름도입니다.



**구체적인 기능 순서도**

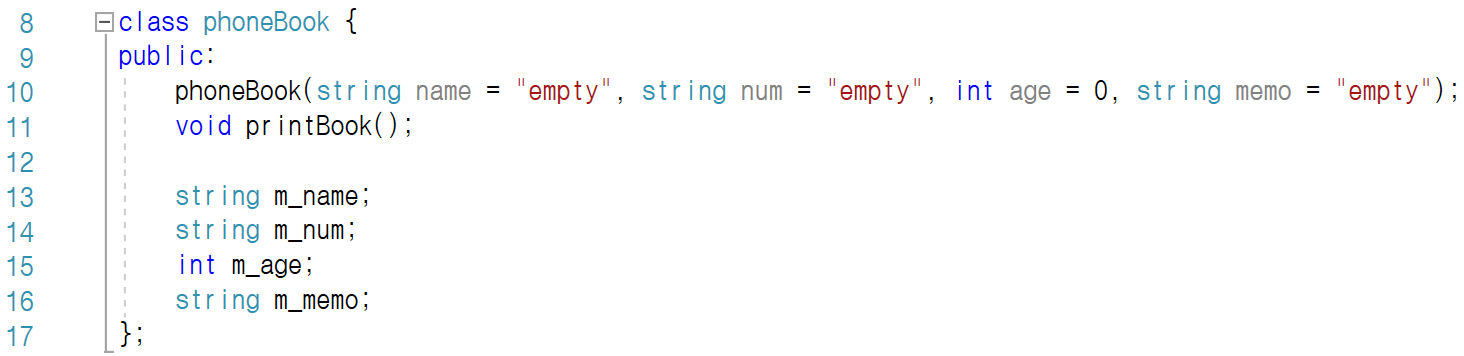
**Ⅲ. 프로그램 구현**

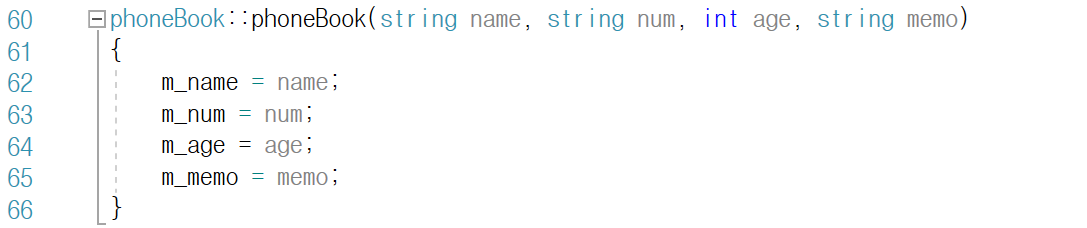
**Ⅲ - 1) 주요 함수 및 기능 설명**

앞서 설계 단계에서 나눈 6가지 주요 기능의 함수 및 기능에 대한 설명입니다. 6가지 기능 설명 전에는 연락처 구성 객체와 리스트 구현에 대한 설명을, 여섯 가지 기능의 설명 뒤엔 main()부분의 흐름이 설명됩니다. 이 프로그램은 하나의 cpp파일로 구현됩니다.

**◈ 연락처 객체 구성**

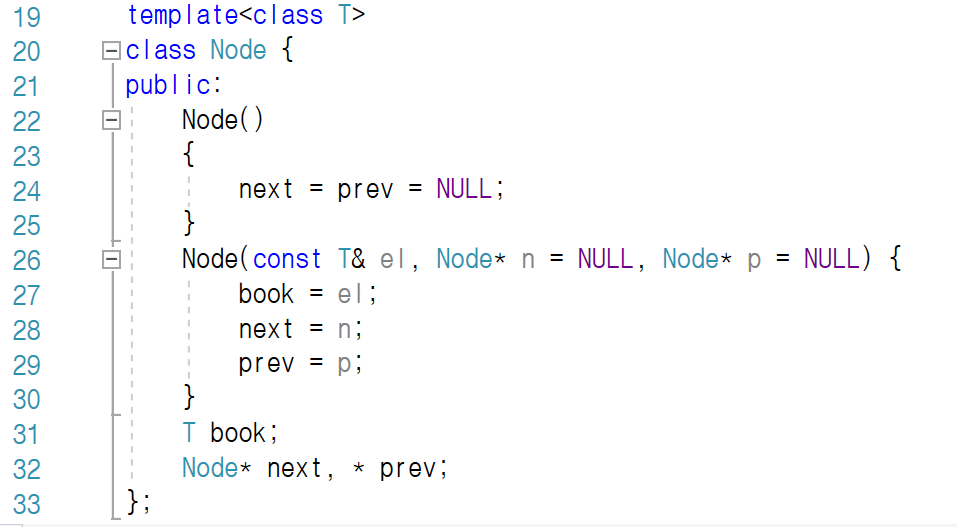
연락처는 phonebook 클래스로 하여 string 이름, string 전화번호, int 나이, string 메모를 구성합니다. 클래스 변수도 설정하여 phoneBook에 접근이 가능하게 됩니다.



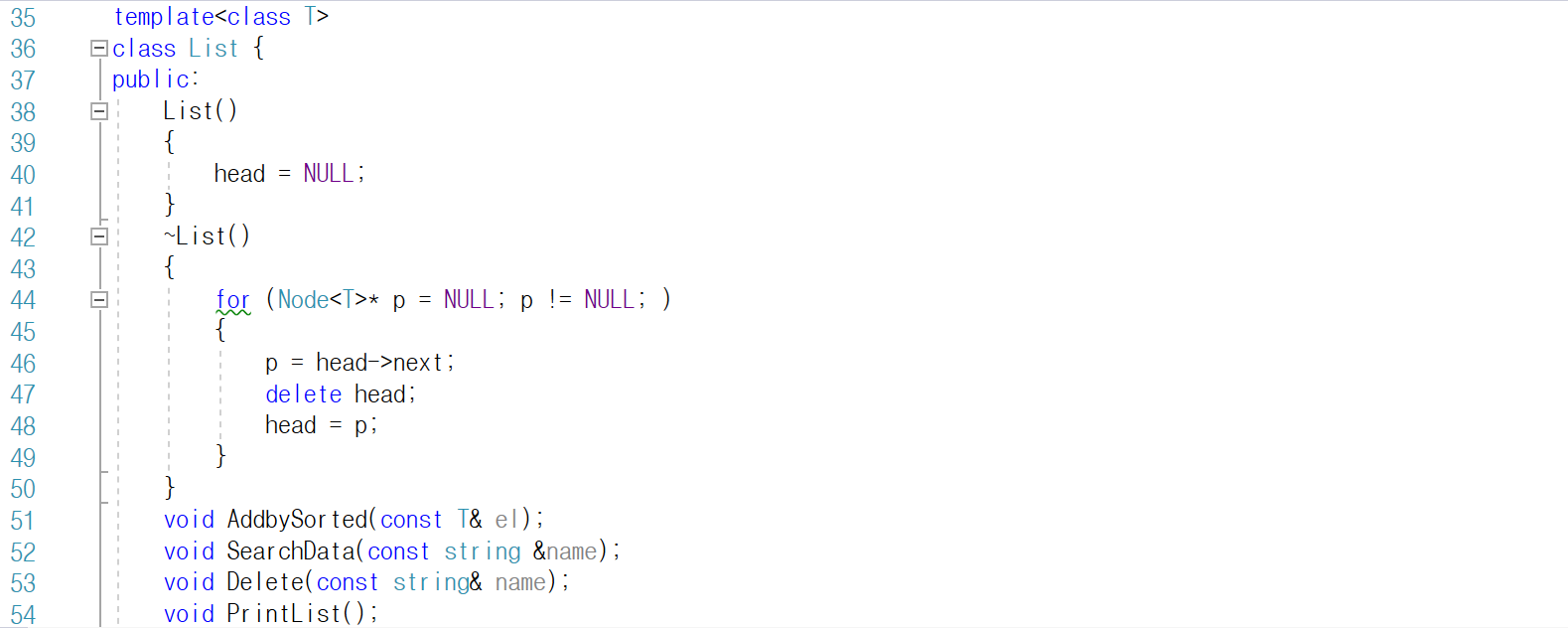


**◈ 연결리스트 구현**

이름을 기반으로 한 전화번호, 나이, 메모에 대한 정보를 하나의 연락처로 구성하기 위해 phonebook 클래스를 사용하였으며. 이 연락처(데이터)를 포인터로 연결시켜서 데이터의 삽입과 삭제가 보다 자유롭게 됩니다. 배열과 달리 미리 공간을 형성하지 않아 메모리가 보다 효율적으로 작동됩니다.



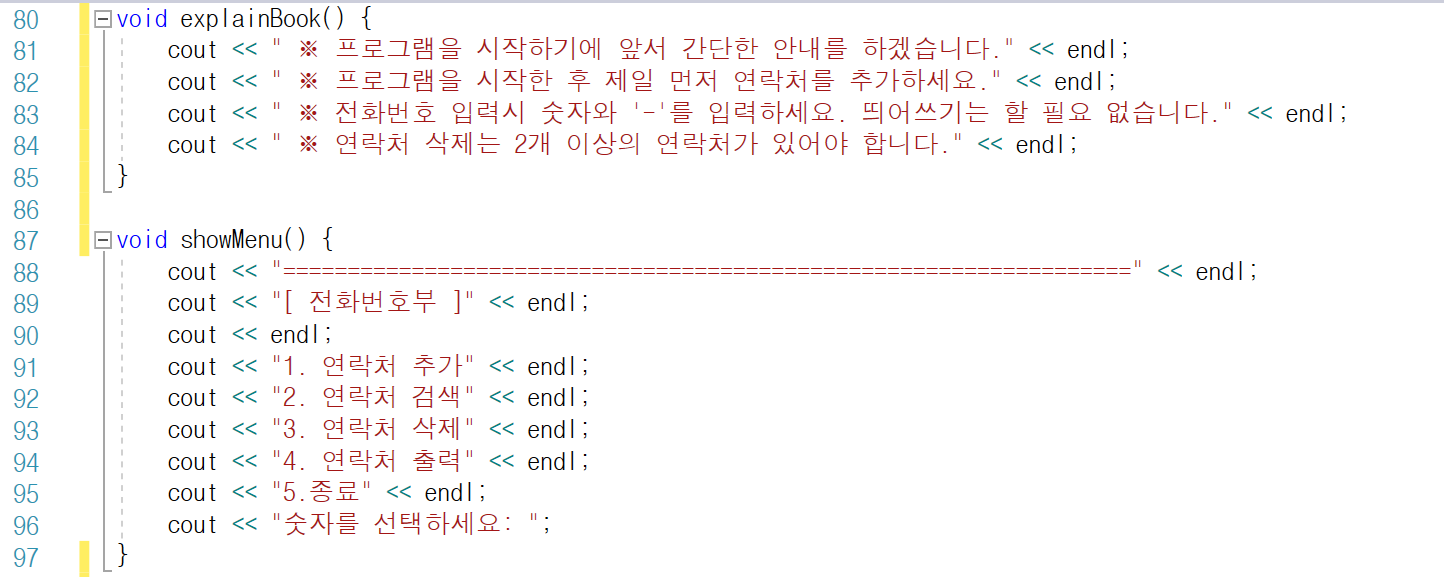
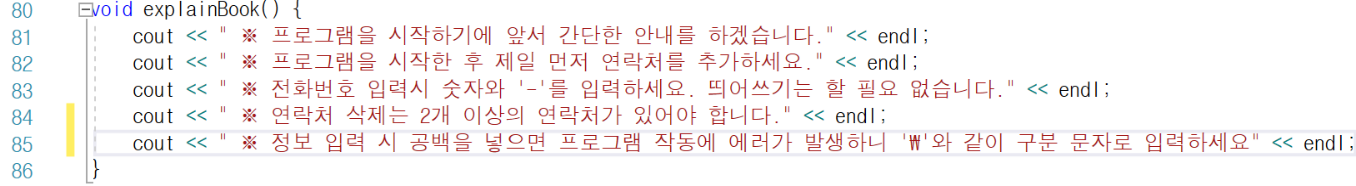
우선, node의 \* next와 \* prev 포인터를 통해 연결리스트를 설계하되, 자료형에 따라 유동적인 동작을 하도록 template 클래스로 설정하며, 데이터(T book)를 받아들입니다. 접근의 용이성을 위해 public으로 선언하여 구현이 됩니다.





그 후 List 클래스를 통해 포인터로 데이터를 연결하여 리스트가 구성됩니다. 주요 기능을 담당하는 함수를 public으로 하여 데이터 접근이 쉽게 됩니다. Node의 head 포인터는 private으로 하여 접근을 제한하여 정보가 보호됩니다.

**① 전화번호부 기능 제시 및 기능 선택**



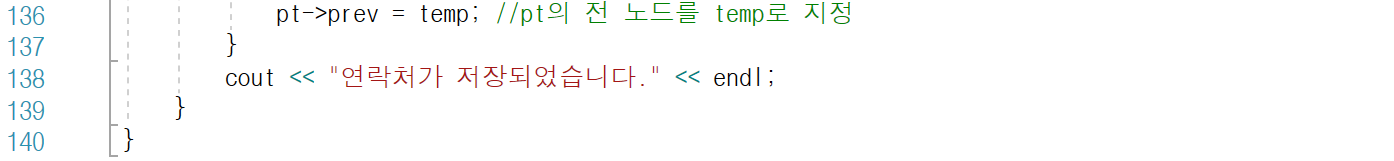
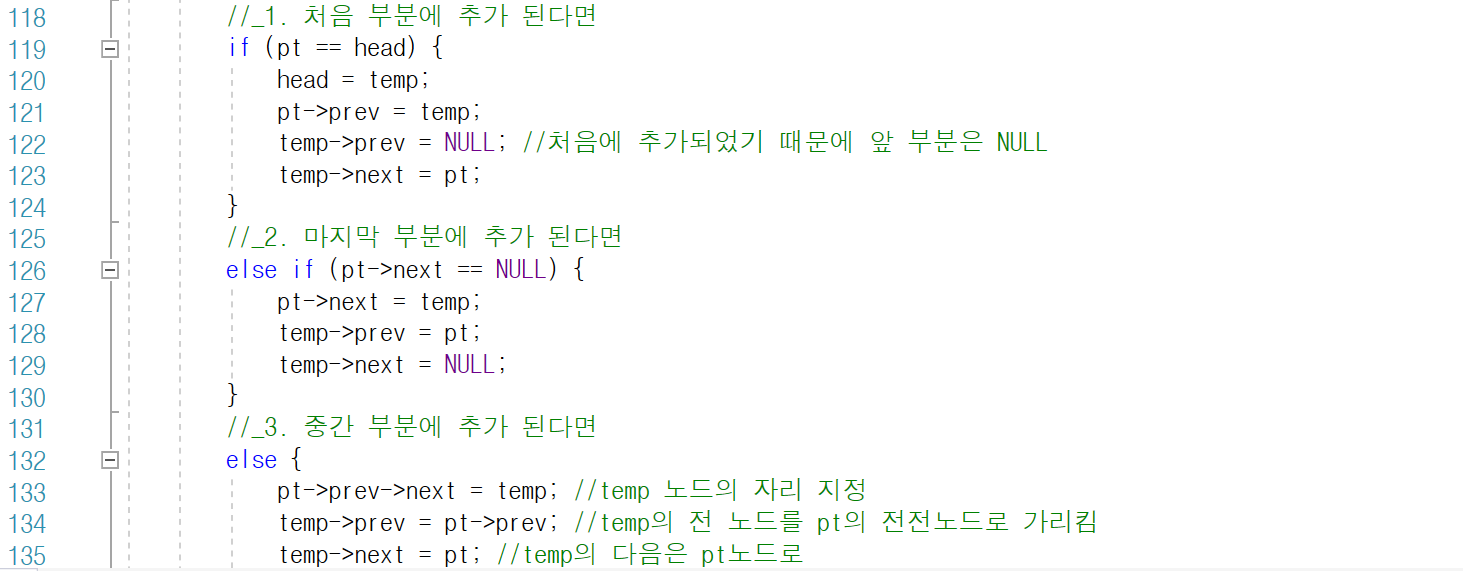
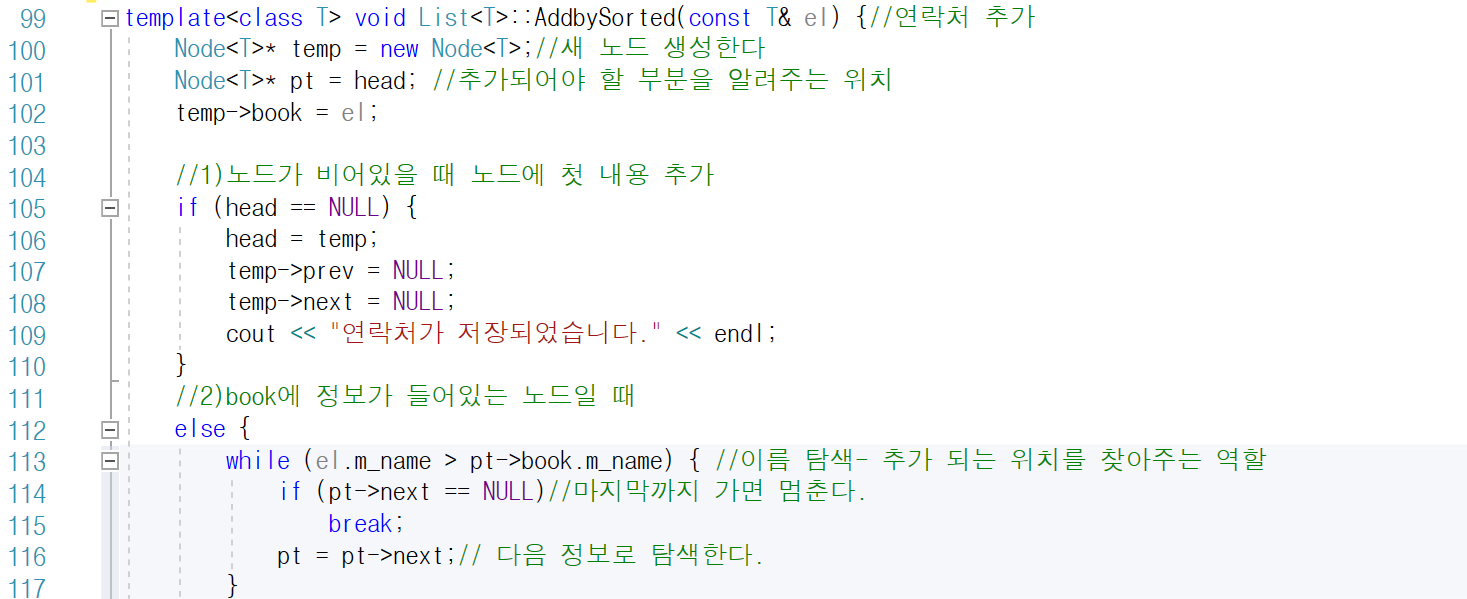
프로그램이 시작되면 간단한 안내 문구가 출력되고 기능 선택이 나타납니다.

**② 연락처 추가**

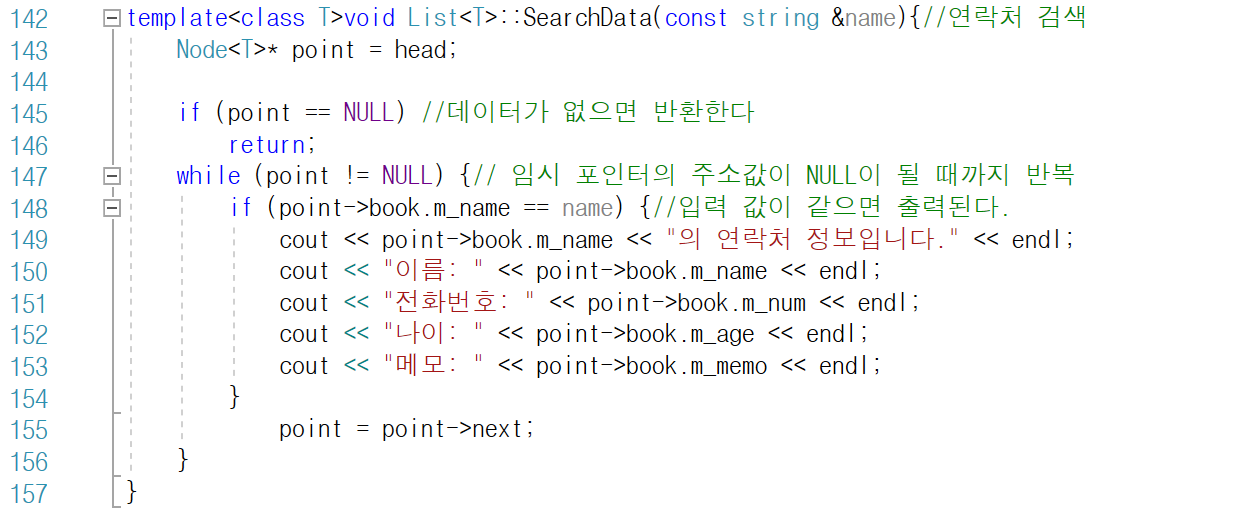
임시 새 노드를 생성하여 정보가 추가되는 위치에 따라 삽입하고 리스트를 이어주는 작업이 구현됩니다. 특히 추가할 때 저장될 정보와 기존 정보의 이름을 비교하면서(영어->한국어 순으로) 위치를 탐색하고 연락처를 추가하여 이름 순으로 정렬이 됩니다.

연락처가 빈 리스트에 처음 추가된다면 head가 추가되는 리스트로 바뀝니다. 그 후 연락처 정보가 저장되어 있는 리스트의 경우로 구현됩니다. 저장되는 정보의 이름을 저장된 리스트 정보의 이름과 비교하면서 리스트를 추가해줍니다. 처음 부분에 정보가 추가된다면 그 리스트를 head로 바꾸고 추가되는 리스트와 기존 첫 리스트를 포인터로 연결시킵니다. 중간 리스트에 추가된다면 추가되는 전 포인터로 추가되는 리스트 자리를 지정하고 추가되는 후 리스트의 포인터로 새 리스트를 연결하여 리스트가 구현됩니다. 마지막 자리에 추가된다면 리스트를 연결한 후 다음 포인터를 NULL값으로 지정하여 리스트가 구현됩니다.

특히 정보가 2개 이상이 들어가 리스트를 구현하고 있을 때는 입력 받은 이름값과 리스트 내의 이름을 비교하여 순차적으로 리스트가 구성됩니다.

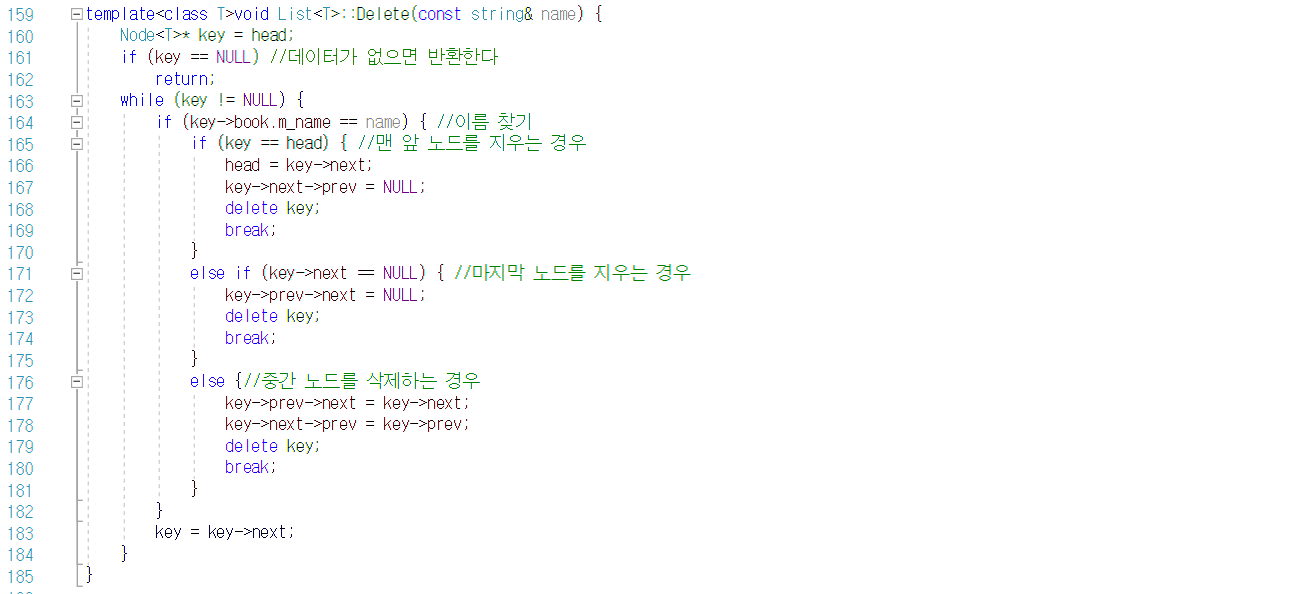


**③ 연락처 검색**



검색은 SearchData()를 통해 구현됩니다. 검색할 이름의 주솟값(const string &name)을 받아 리스트 내의 이름과 동일한지 확인합니다. 리스트의 첫 부분부터 탐색이 되며 검색할 이름 값과 리스트 내의 이름이 동일하면 그 이름에 대한 정보가 출력됩니다. 포인터를 다음 리스트의 정보로 넘겨 탐색이 됩니다.

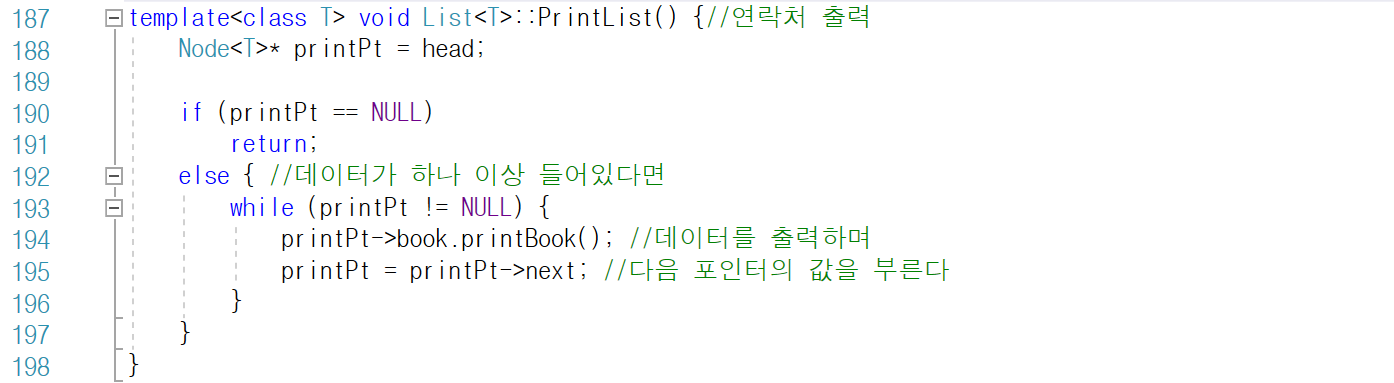
**④연락처 삭제**



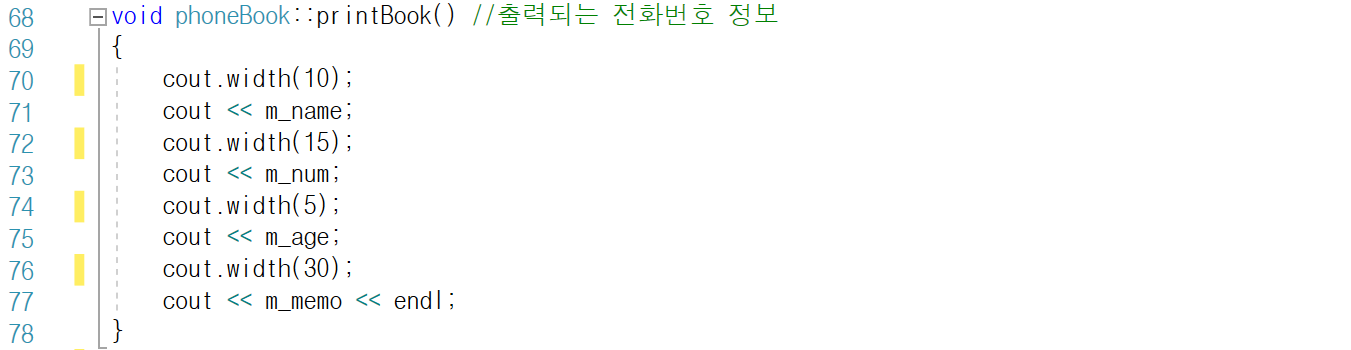
연락처 삭제는 Delete() 함수를 통해 삭제가 됩니다. 입력 받은 이름의 주솟값(const string &name)을 받아 리스트를 탐색하게 됩니다. 리스트 내에서 하나씩 순차적으로 탐색하다가 같은 이름의 데이터를 발견하면 delete key를 통해 해당 리스트가 삭제됩니다. 삭제를 할 때에도 포인터를 통해 순차적으로 리스트를 탐색하게 되며, 포인터를 다음으로 넘겨줌으로 탐색이 이루어집니다.

삭제가 되는 위치에 따라 리스트를 이어주는 작업이 다르게 이루어집니다. 첫 리스트를 삭제한다면 두 번째 리스트를 head로, prev포인터는 NULL로 하여 첫 리스트가 됩니다. 중간 리스트가 삭제된다면 삭제되는 리스트의 전 리스트의 포인터와 삭제되는 다음 리스트의 포인터로 서로 연결하게 됩니다. 마지막 리스트가 삭제된다면 삭제되는 전 리스트의 next 포인터를 NULL로 하여 리스트가 끊기지 않게 연결됩니다.

**⑤ 전화번호부 출력**



전화번호부 출력은 PrintList()를 통해 출력됩니다. 임시 새 노드의 포인터가 리스트를 순차적으로 지나가면서 printBook()을 통해 한 줄 씩 출력합니다.

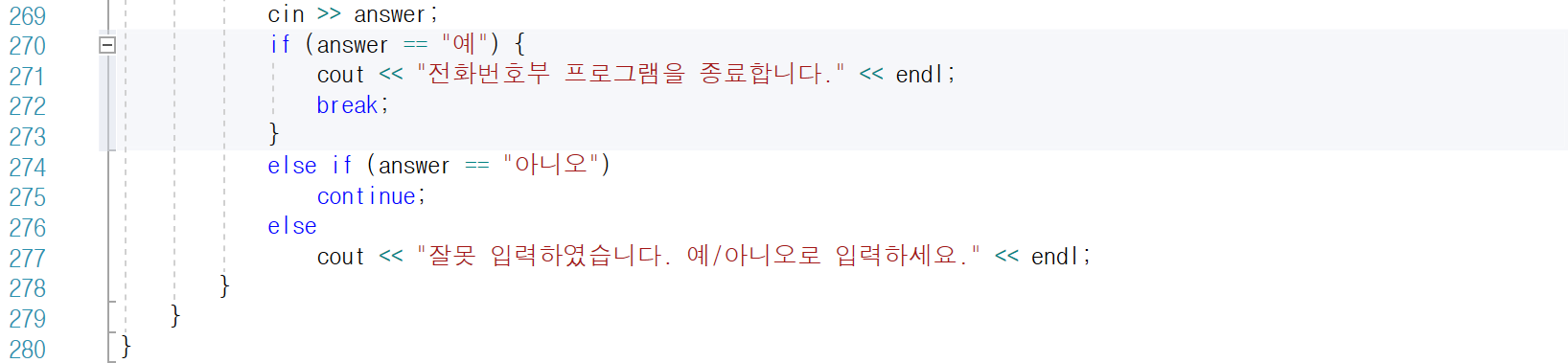
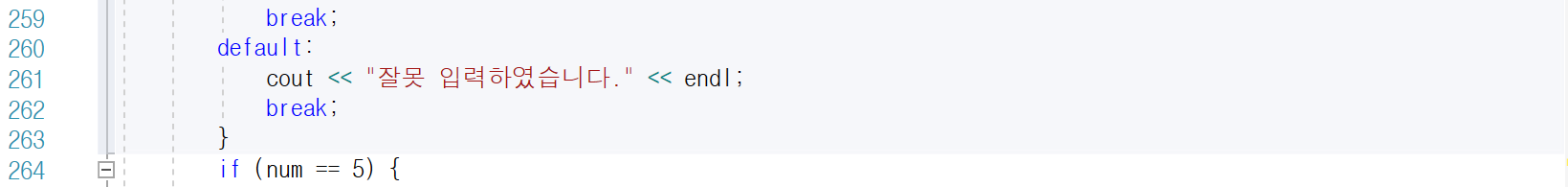
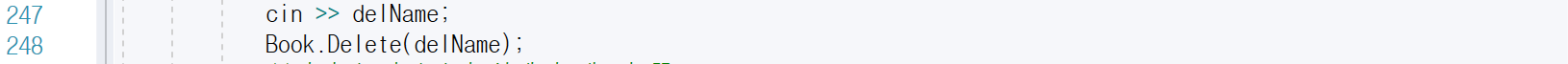
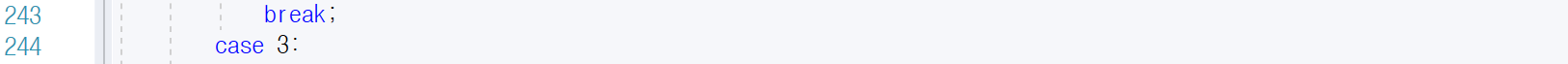
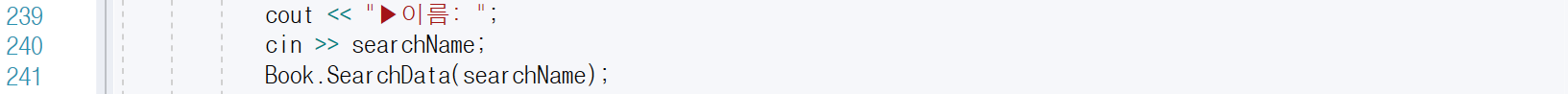
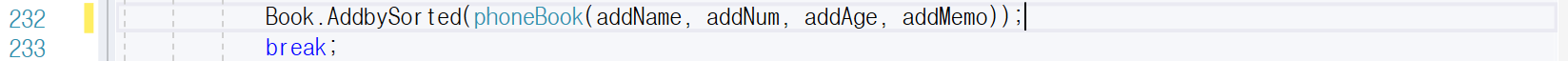
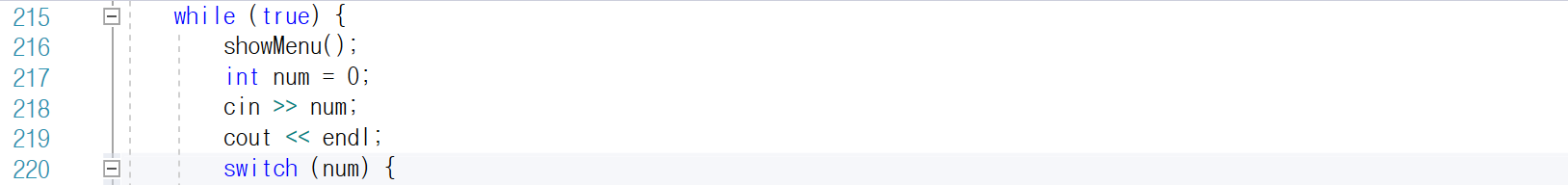
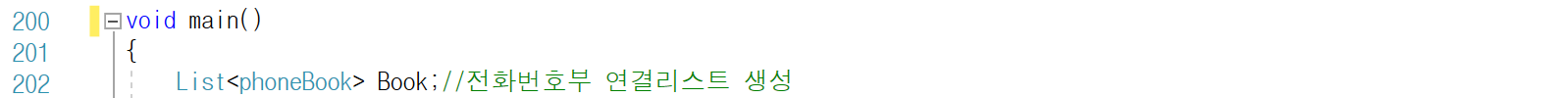


printBook()은 리스트로 연결된 연락처들 중 하나의 연락처에 들어있는 이름. 전화번호, 나이, 메모를 출력하는 함수입니다. width(int num)을 통해 출력 모양을 맞춰줍니다. phoneBook 클래스의 정보를 가져오는 역할을 합니다.

**◈ main() 함수**

메인 함수에서는 while(true)문과 switch-case문을 통하여 각 기능이 반복적으로 수행됩니다. while문을 벗어나는 순간 종료가 됩니다. switch-case문 내에서는 쓰이는 지역변수를 선언하고, 각 함수를 불러옵니다. phoneBook 클래스를 사용하여 구성한 연결리스트인 List 클래스로 전반적인 데이터를 넣을 리스트 공간을 만듭니다. 이를 Book으로 통용하여 리스트 내에 정보를 추가하고 검색하고 삭제되도록 합니다.

다음 사진은 main()함수에서 주요 기능을 하는 부분을 간략히 해 놓은 것입니다. 앞서 말한 Book 연결리스트를 생성하고 반복문 내에서 각 기능을 수행하는 모습과 그 기능이 쓰이는 모습을 보여주는 그림입니다.

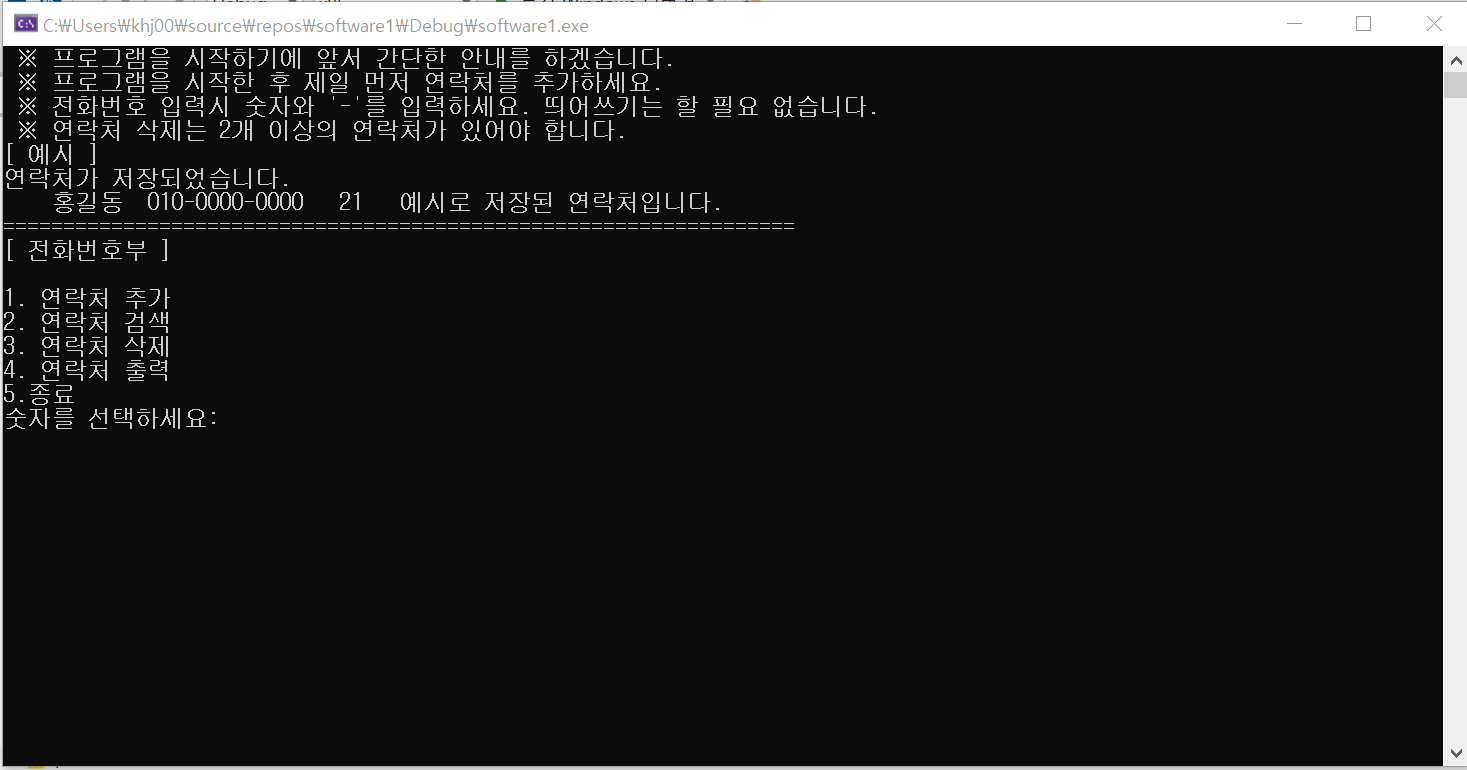


전체적으로 구현된 코드는 software1 프로젝트명으로 소스.cpp입니다. 주요 기능 설명을 위해 일부분을 첨부한 것으로, 전체 구현 파일은 따로 첨부합니다.

**Ⅳ. 프로그램 테스트 결과**

**Ⅳ - 1) 기능 별 구현 결과**

아래의 사진들은 실제 프로그램을 돌린 결과를 첨부한 것으로 각 기능이 구현되는 방식이 나타납니다.

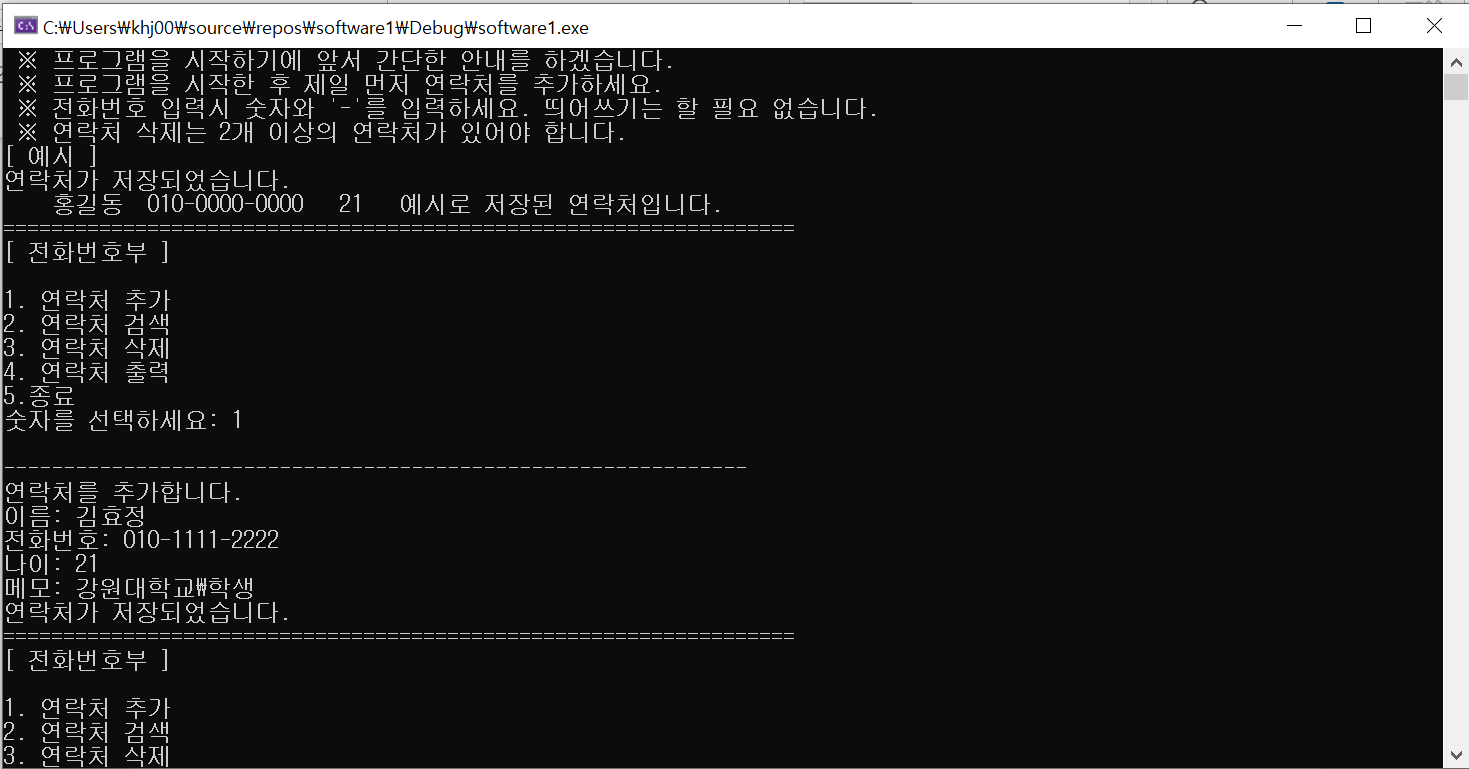


안내 문구 뒤에는

입력 예시를

알려줍니다.

**[프로그램 시작 시 나오는 안내 문구 및 기능 제시]**



1을 입력하면

이름:

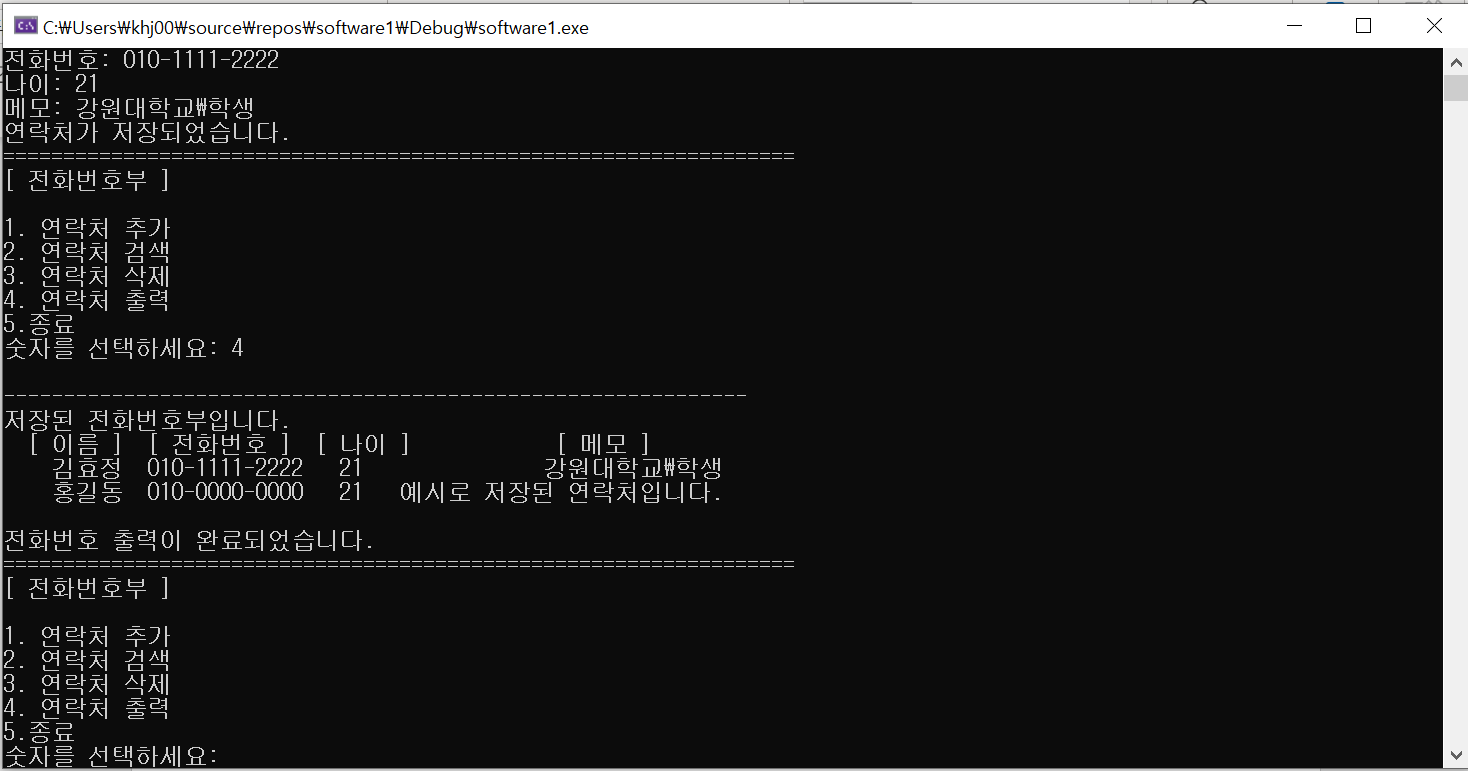
전화번호:

나이:

메모:

입력 말이 출력됩니다.

**[ 1 연락처 추가 기능 선택 및 출력 결과**



4를 입력하면

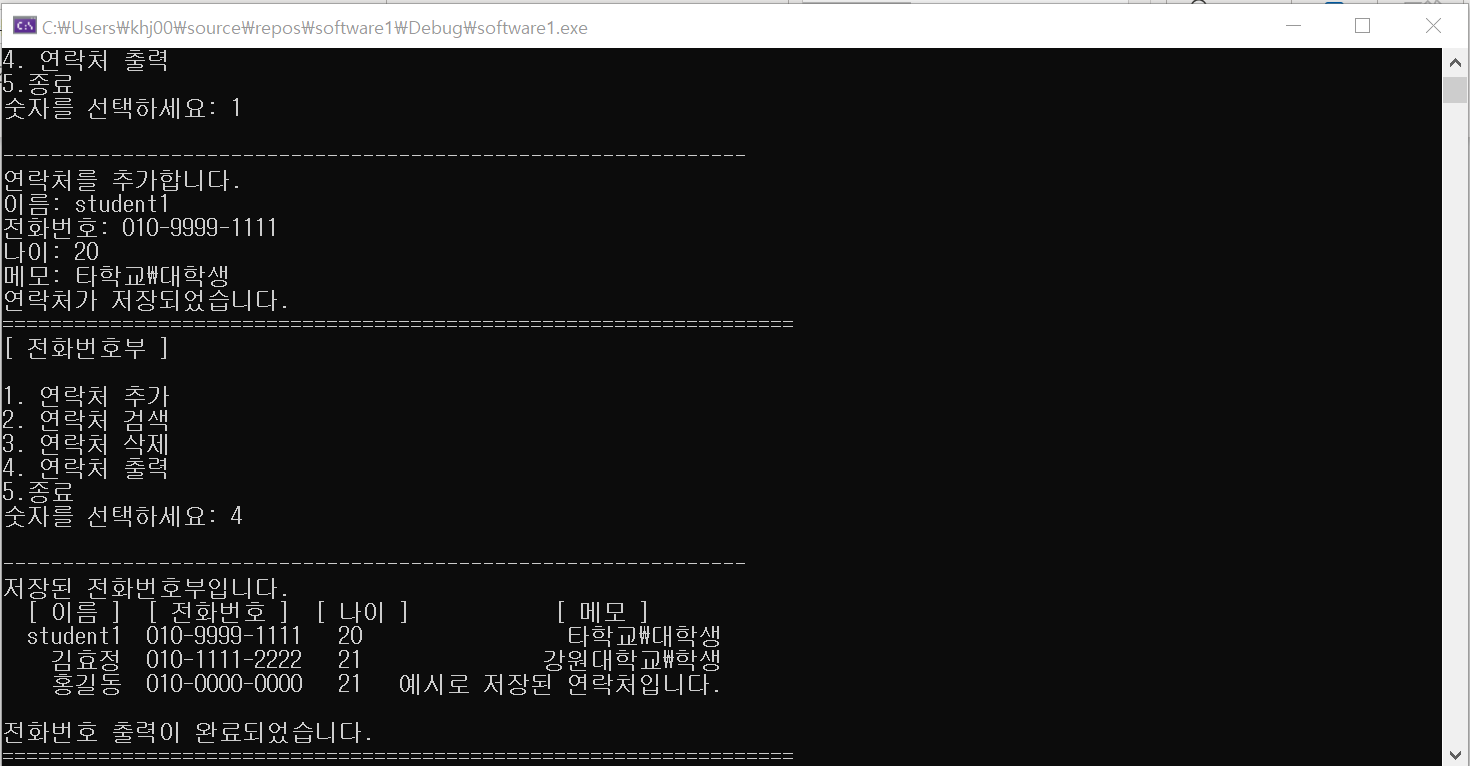
이름, 전화번호,

나이, 메모 순으로

전화번호가

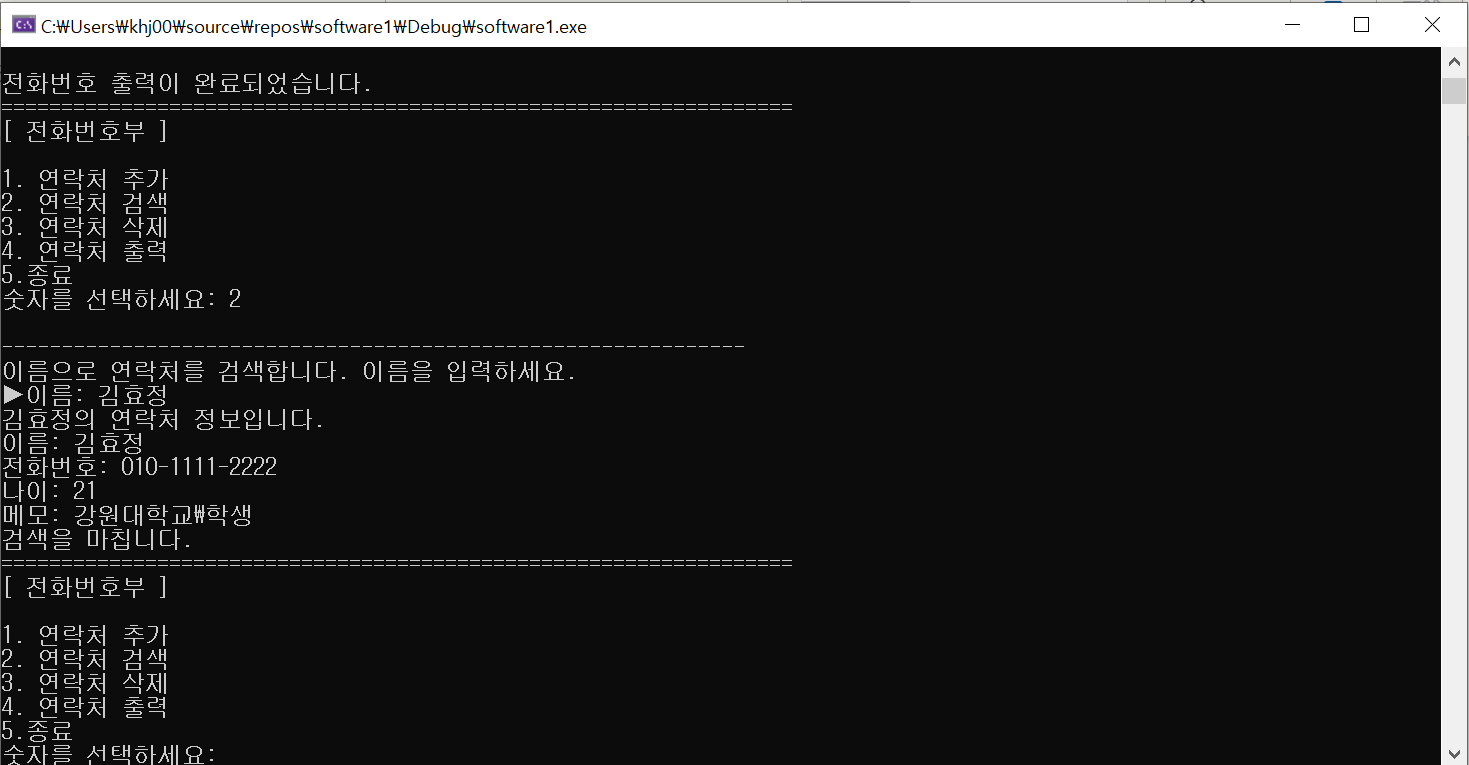
출력됩니다.

**[4 연락처 출력 기능 선택 및 출력 결과]**



추가된 연락처(student1)가 추가되고

순서대로 출력됩니다.

**[1, 4 한 번 씩 더 수행한 결과]**

2를 입력하고

검색을 하고 싶은

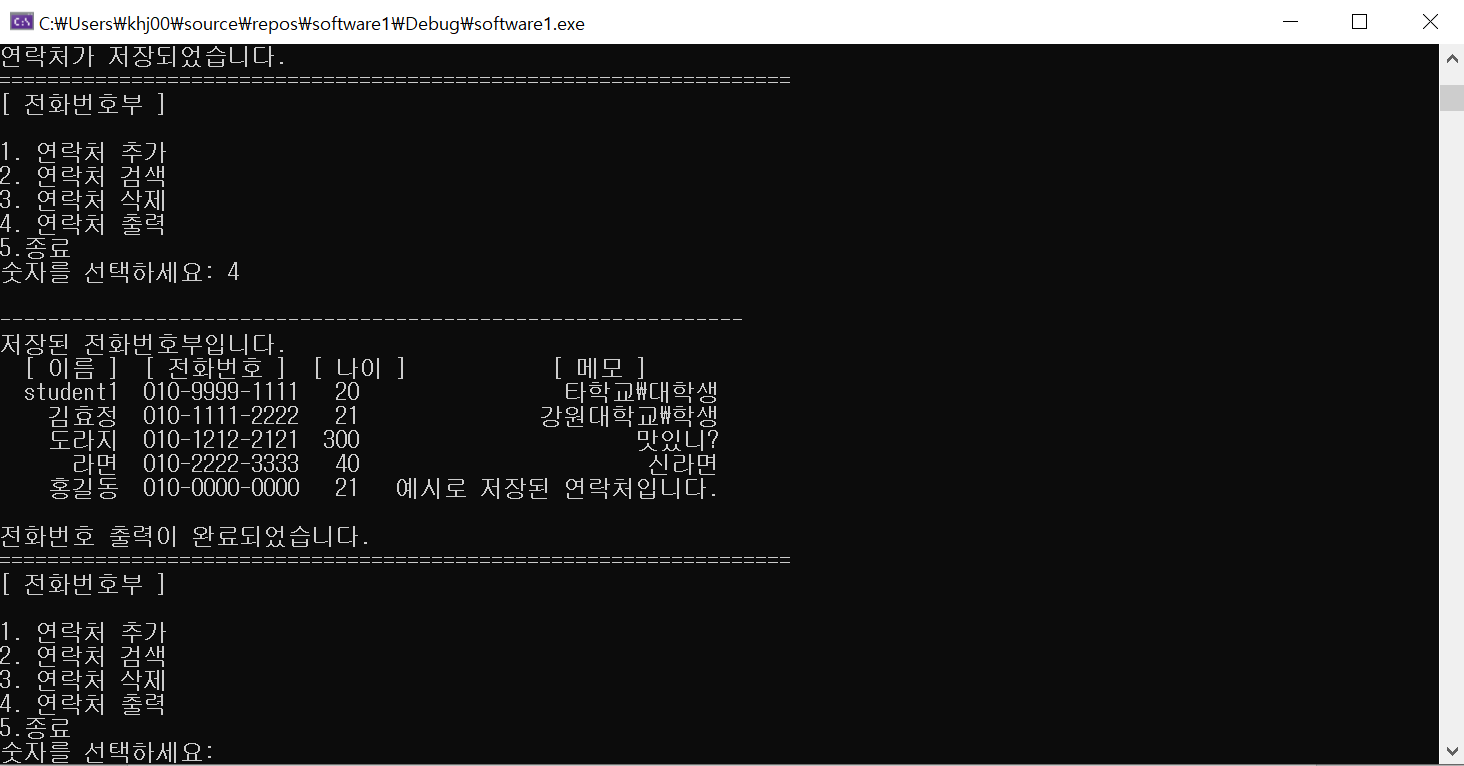
연락처의 이름을

검색하면 동일한 연락처

정보가 출력됩니다.

->김효정 검색한 결과

**[2 연락처 검색 기능 선택 및 출력 결과]**



다음과 같이 이름

순으로 정렬이 됩니다.

->라면, 도라지 추가

->세 번째 입력한 파프리카는

마지막에 저장되어 삭제하고

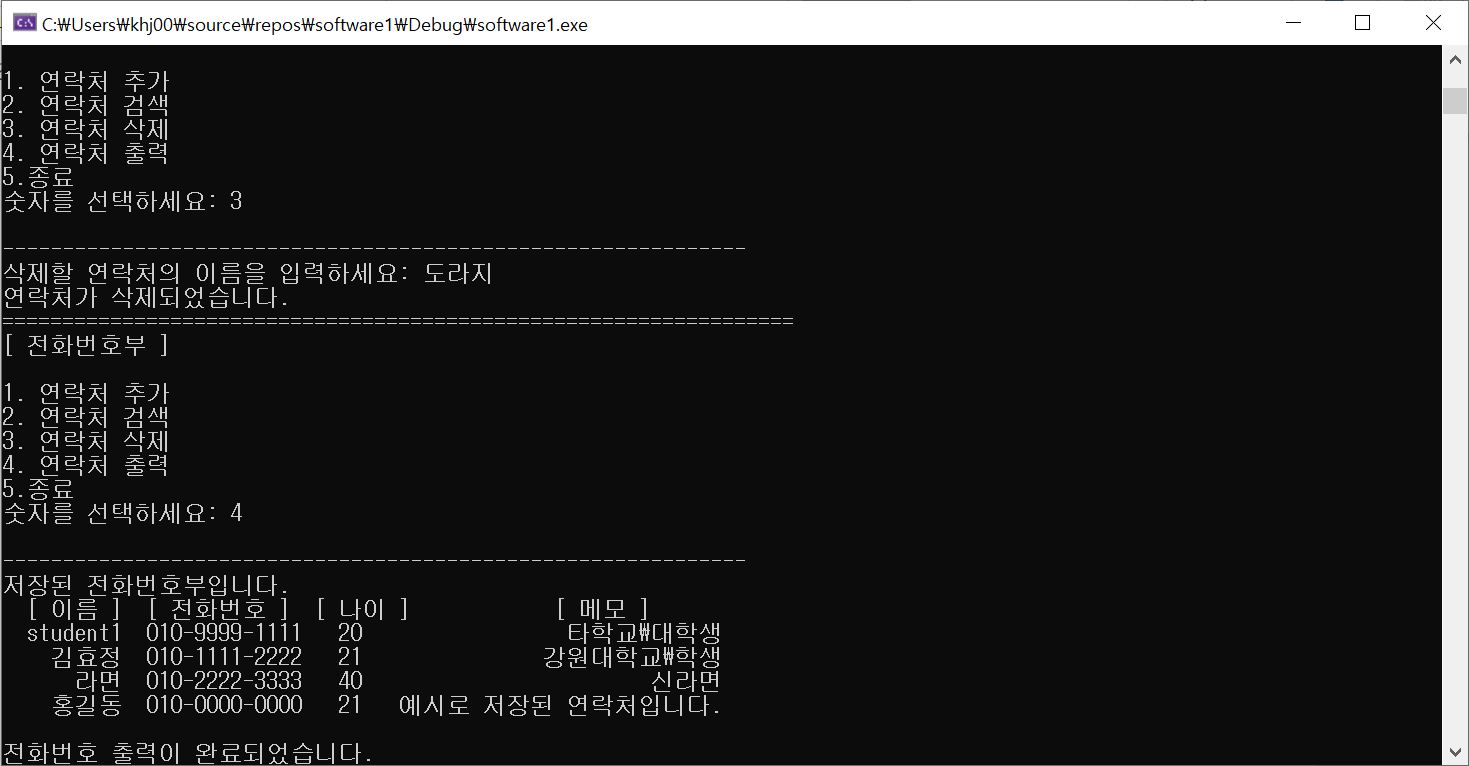
4, 5번 째 정보 입력한 결과입니다.

세 번째에 입력되는 정보는

정렬이 되지 않고

맨 마지막에 저장됩니다.

**[추가와 출력 반복한 결과]**



3을 입력하고

삭제 하고 싶은

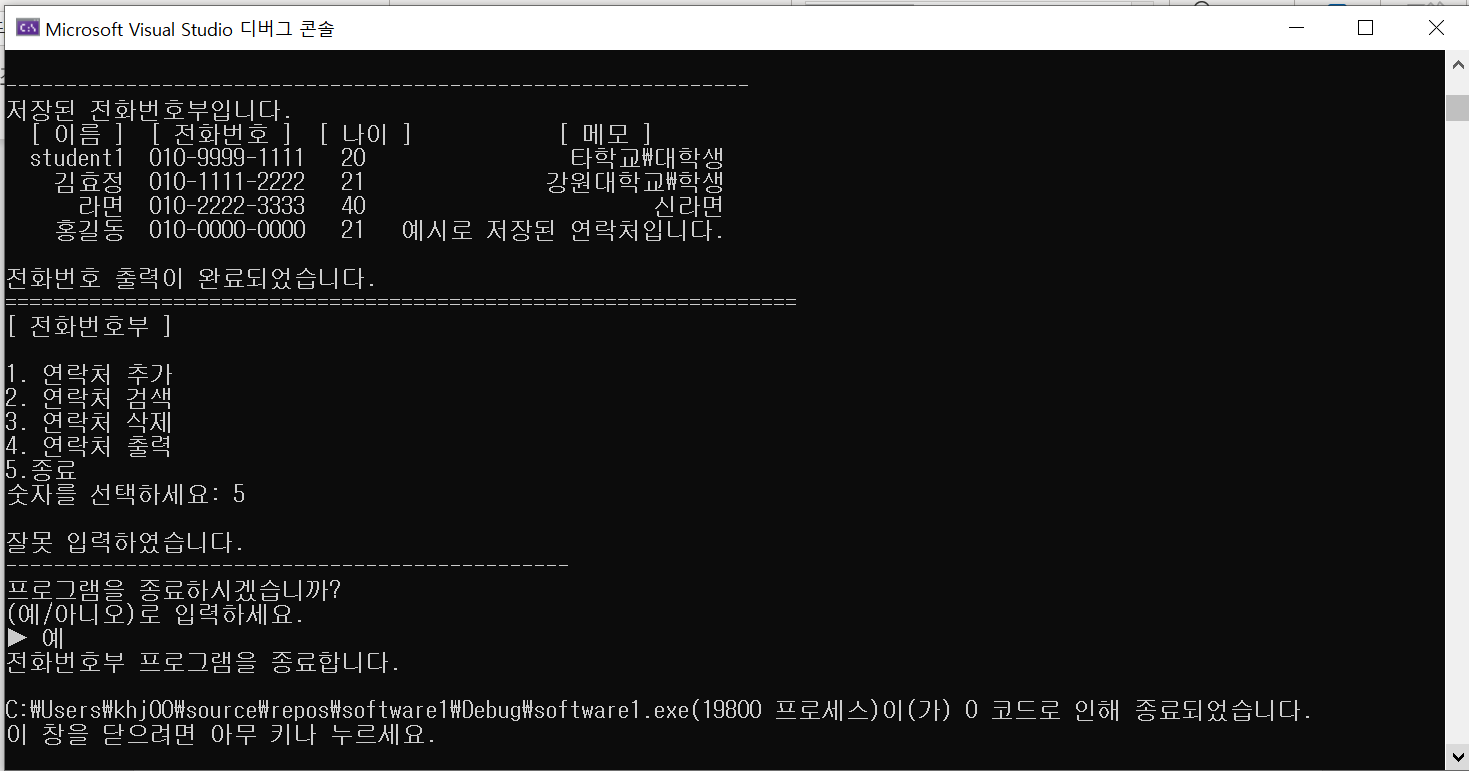
연락처의 이름을 입력하면 연락처가 삭제됩니다.

확인을 하려면

4출력 기능을 통해

확인합니다.

**[3 연락처 삭제 수행 및 출력 결과]**



5를 입력하면 종료를

안내하는 문구가

출력됩니다.

예를 입력하면 종료,

아니오를 입력하면

메뉴가 다시 나옵니다.

**[5 종료 수행 및 결과 – 프로그램 종료]**

**Ⅳ - 2) 프로그램 분석**

전화번호부 프로그램을 실제 코드로 구현하면서 생각하지 못했던 기능이 생기거나 프로그램 실행과정에 문제가 발생하여 구현에 포기한 기능에 대한 설명입니다.

**◈ 전반적인 분석**

우선 연락처 정보에 대한 구조체를 만들 때, 클래스로 형성할 지, 구조체(struct)를 이용할 지, template, typedef 등 여러 구조를 고민하다가 string으로 받고 자료형에 구애 받지 않도록 하기 위해 class로 정보 더미를 만들고 template으로 연결리스트와 호환되게 만든 프로그램입니다. 처음에는 전화번호를 int형으로 받으려고 구현하였으나 출력 과정에서 ‘-‘ 기호를 출력하기 위해서는 int를 문자열로 바꿔주어야 하는 등의 일련의 과정을 거쳐야 했고, 과정을 줄이기 위해 전화 번호 또한 string으로 선언하여 보다 간편한 출력이 되게 만든 프로그램입니다. 또한 각 정보의 입력 글자에 제한을 두어 설계 하였으나 string으로 사용하게 되면서 제한이 사라졌고 보다 편하게 입력을 받게 됩니다.

데이터(연락처 정보) 구축 과정에서 배열은 미리 공간을 할당하고 시작하기에 여러 개의 많은 데이터를 넣는 전화번호부 프로그램에서는 효율성이 떨어진다고 생각하게 되어 보다 효율적이고 선형적으로 데이터를 보관할 수 있는 자료구조 중 연결리스트로 구현한 프로그램입니다.

이런 연결리스트의 구조로 인해서 연락처 구조(이름-전화번호-나이-메모)의 추가, 검색, 삭제, 출력이 용이한 프로그램이 만들어 집니다. template으로 인해 연락처 구조 변수에 대한 접근이 용이하게 되었으며, 연결리스트의 선형적 자료 구조로 인해 정보에 대한 접근이 쉽게 됩니다. 추가, 검색, 삭제, 출력의 기능을 구성할 때, 포인터로 연결하는 부분이 복잡하였고 특히 연락처 클래스의 변수들과 연결리스트의 호환을 하는 것이 쉽지 않았으나 기본적인 작동이 구현되게 만든 프로그램입니다.

또한 string을 이용하여 정보를 저장하고 접근하기에 제약이 보다 적어 검색, 출력이 용이한 프로그램이 됩니다.

**◈ 문제점 분석**

연결리스트와 string의 사용은 최종적으로 출력된 파일이 텍스트 파일로도 저장이 되게끔 하는 파일 입출력 구현에 실패하게 된 원인이 됩니다. 대게 문자열을 이용하여 파일 입출력을 하기에 string을 통한 간단한 입출력을 겨우 했으나 정보가 각 함수마다 새 노드로 연결된 연결리스트의 구조로 인해 정보를 파일에 입력하는 것을 성공하지 못했습니다.

또한 함수의 기능에 있어서 부족한 기능을 지닌 함수도 존재합니다. 연락처를 추가하는 AddbySorted()함수에서의 문제점은 두 번째로 입력하는 정보가 추가는 되지만 정렬되지 않는 것입니다. 그러나 두 번째로 저장한 연락처를 삭제하고 다시 입력하면 원래 순서로 정렬이 되고 그 외의 순서로 입력한 정보는 정렬되어 추가가 잘 됩니다. 다음으로는 검색을 하는 SearchData()함수에 대한 아쉬운 점입니다. 검색 함수에서 이미 저장된 이름과 같은 이름이 들어오면 ‘동일한 이름의 연락처가 존재합니다. 다시 입력해주세요’와 같이 중복 확인을 하고 싶었으나 구현에 어려움을 겪은 함수입니다. 연결리스트에 순차적으로 접근하다 보니 이미 입력된 이름을 찾아도 출력 부분에서 섞이게 되어 기본 검색 기능과 정보 출력 기능이 되는 함수로 마무리 했습니다. 마지막으로 삭제를 하는 Delete()함수에 대한 문제점입니다. 삭제에서는 하나의 연락처만이 저장되어 있는 리스트에서 저장된 연락처를 삭제하면 프로그램이 반복되지 않고 종료되는 문제입니다. 이를 해결하기 위해 삭제 함수 내에 위치한 “데이터가 없을 때, return한다.” 부분을 지워보고 해봤으나 안내 문구로 정보가 하나일 때는 삭제가 안된다는 것을 표현했습니다.

마지막으로는 정보를 입력할 때 공백을 입력할 수 없다는 문제입니다. 입력할 때 공백이 입력되면 프로그램이 무한으로 돌아가는 문제가 발생하나 이 부분도 마찬가지로 제한하지 않고 문구로 안내를 했습니다.

**>> 프로젝트를 마치며**

여러 기능이 복잡하게 실행되는 프로그램을 만들게 되면서 효율성과 접근성에 대해 생각을 많이 할 수 있게 된 프로젝트였습니다. 예전에는 배열과 같이 고정된 메모리의 크기를 사용했다면 지금은 보다 효율적으로 돌아갈 수 있는 방법을 모색하게 되고, 클래스나 구조체를 통해 접근의 용이성에 대해 생각할 수 있었습니다.

또한 설계에서 계획하고 구상했던 세부 기능이 실제 구현할 때 생각보다 다르기도 하여 많이 헤매기도 했습니다. 그러나 설계 단계에서 기본적으로 그려 놓은 흐름으로 인해서 목표했던 기능이 흔들리지 않을 수 있었습니다.