

## ■ 기초수준 객체화

기존 예제에서 등장했던 함수들을 세 개의 클래스에 나눠서 넣는 것입니다. 세 개 클래스의 구성과 기능은 다음과 같습니다.

클래스 이름	설명
CUserData	USERDATA 구조체와 동일한 의미를 갖는 클래스
CMyList	CUserData 클래스의 인스턴스를 단일 연결리스트로 관리하는 클래스
CUserInterface	CMyList 클래스를 참조를 멤버로 가지면서 사용자 인터페이스를 구현한 클래스

다음 단계에 따라 절차지향 프로그램의 구조를 객체지향 프로그램의 구조로 바꿔보겠습니다.

### 1. AddressBook\_OOP 라는 새 프로젝트를 생성하고

## 새 프로젝트 구성

빈 프로젝트

C++

Windows

콘솔

프로젝트 이름(N)

AddressBook\_OOP

위치(L)

D:\VC

...

솔루션 이름(M) ⓘ

AddressBook\_OOP

☒ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

### 2. CUserData 클래스를 프로젝트에 추가합니다.

클래스 추가

클래스 이름(L)

CUserData

.h 파일(F)

CUserData.h

...

.cpp 파일(P)

CUserData.cpp

...

기본 클래스(B)

액세스(A)

public

기타 옵션:

☐ 인라인(I)

☐ 관리됨(M)

확인

취소

### 3. 사용자의 데이터를 저장할 CUserData 클래스 선언 ( CUserData.h 파일 )

```
#pragma once
class CUserData
{
    friend class CMyList;
public:
    CUserData();
    ~CUserData();

    const char* GetName() const { return szName; }
    const char* GetPhone() const { return szPhone; }
    const CUserData* GetNext() const { return pNext; }

    static int GetUserDataCounter() { return nUserDataCounter; }

protected:
    char szName[32];    // 이름
    char szPhone[32];  // 전화번호

    CUserData* pNext;

    static int nUserDataCounter;
};
```

### 4. 사용자의 데이터를 저장할 CUserData 클래스 정의 ( CUserData.cpp 파일 메소드 내용은 작성하지 않습니다.)

```
#include "CUserData.h"
#include <cstring>

CUserData::CUserData()
{
}

CUserData::~CUserData()
{
}
```

5. CMyList 클래스를 프로젝트에 추가합니다.



클래스 추가

클래스 이름(L) : CMyList    .h 파일(F) : CMyList.h    .cpp 파일(P) : CMyList.cpp

기본 클래스(B) :    액세스(A) : public

기타 옵션:

☐ 인라인(I)    ☐ 관리됨(M)

확인    취소

6. 주소록 리스트를 관리하는 CMyList 클래스 선언 ( CUserData.h 파일 )

```
#pragma once
#include "CUserData.h"

class CMyList
{
public:
    CMyList();
    ~CMyList();

protected:
    void ReleaseList();
    CUserData* pHead;
    CUserData* pTail;

public:
    CUserData* FindNode(const char* pszName);
    int AddNewNode(const char* pszName, const char* pszPhone);

    void PrintAll();

    int RemoveNode(const char* pszName);
};
```

7. 주소록 리스트를 관리하는 CMyList 클래스 정의( CMyList.cpp 파일 메소드 내용은 작성하지 않습니다.)

```
#include "CMyList.h"

CMyList::CMyList()
{
}

CMyList::~CMyList()
{
}

void CMyList::ReleaseList()
{
}

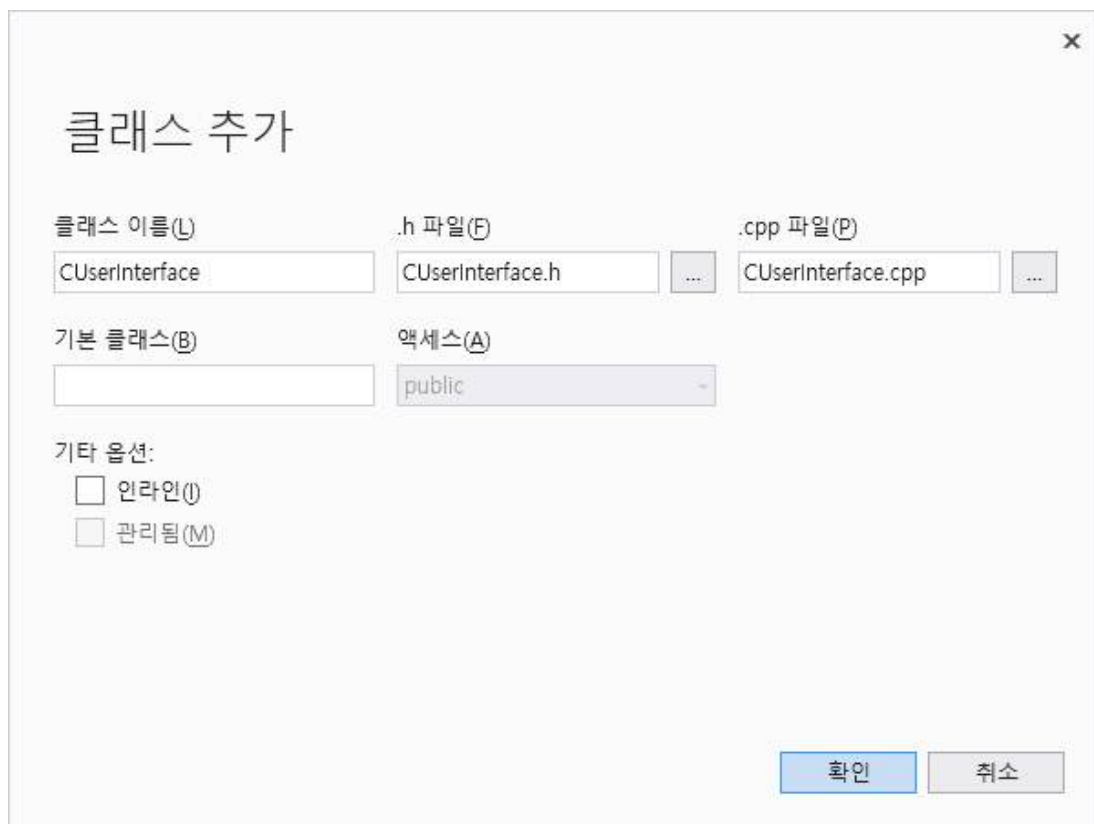
CUserData* CMyList::FindNode(const char* pszName)
{
    return nullptr;
}

int CMyList::AddNewNode(const char* pszName, const char* pszPhone)
{
    return 0;
}

void CMyList::PrintAll()
{
}

int CMyList::RemoveNode(const char* pszName)
{
    return 0;
}
```

## 8. CUserInterface 클래스 추가



클래스 추가

클래스 이름(L) : CUserInterface

.h 파일(F) : CUserInterface.h ...

.cpp 파일(P) : CUserInterface.cpp ...

기본 클래스(B) :

액세스(A) : public

기타 옵션:

☐ 인라인(I)

☐ 관리됨(M)

확인 취소

## 9. 사용자 인터페이스를 관리하는 CUserInterface 클래스 선언 ( CUserInterface.h 파일 )

```
#pragma once
#include "CMyList.h"

class CMyList;

class CUserInterface
{
public:
    CUserInterface(CMyList &rList);
    ~CUserInterface();
    void Add();
protected:
    CMyList& m_List;
public:
    void Search();
    void Remove();
    int PrintUI();
    int Run();
};
```

10. 사용자 인터페이스를 관리하는 CUserInterface 클래스 정의 ( CUserInterface.cpp 파일, 메소드 내용은 작성하지 않습니다.)

```
#include "CUserInterface.h"

CUserInterface::CUserInterface(CMyList& rList)
{
    : m_List(rList)
}

CUserInterface::~CUserInterface()
{
}

void CUserInterface::Add()
{
}

void CUserInterface::Search()
{
}

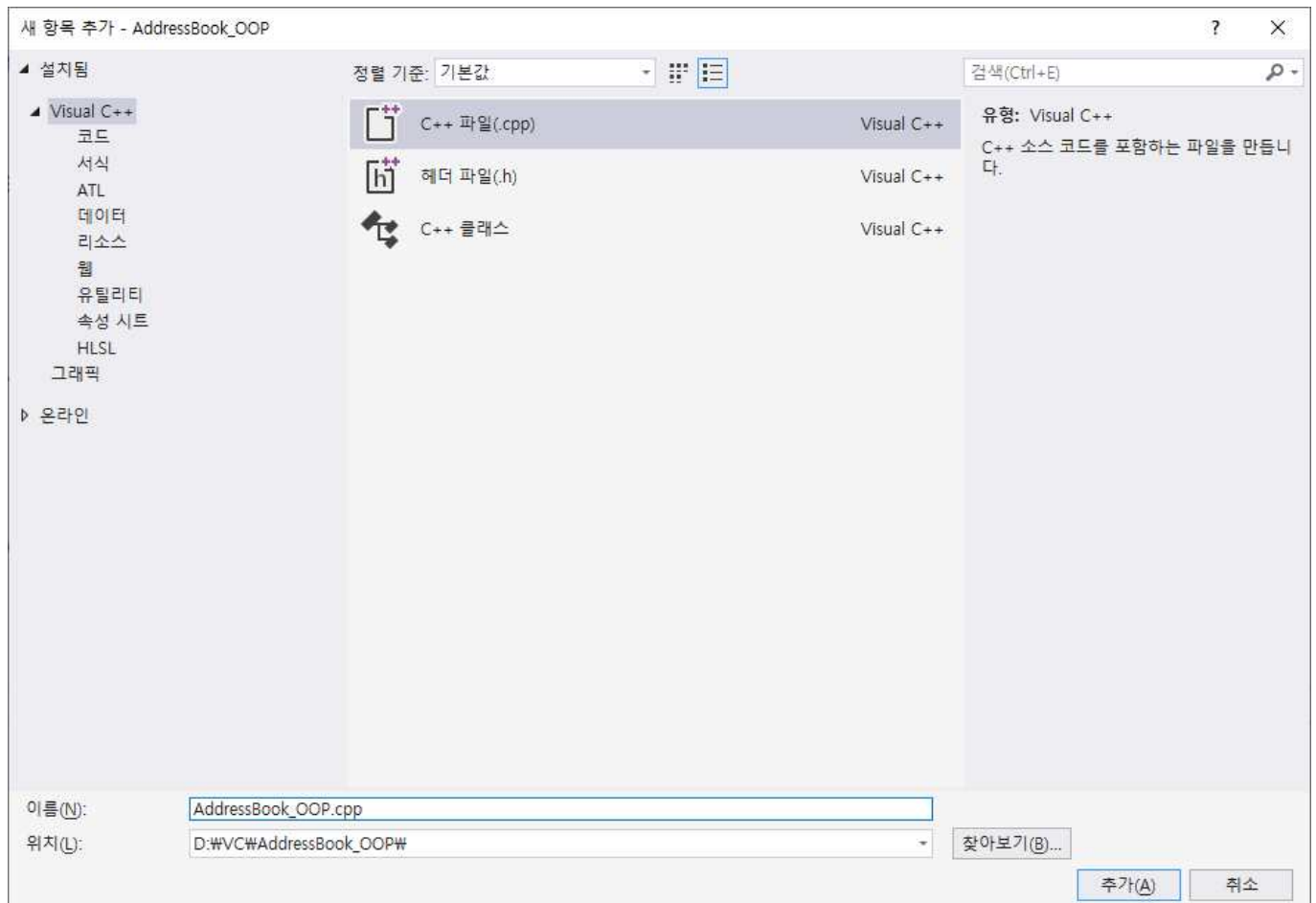
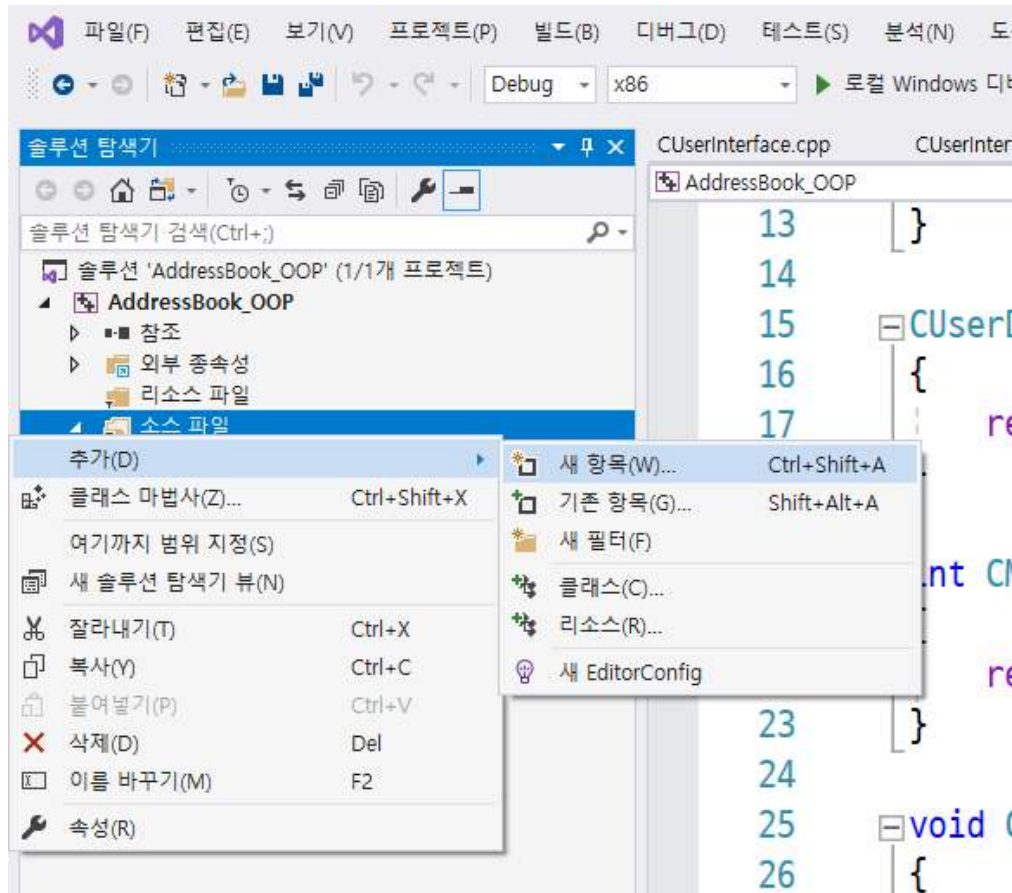
void CUserInterface::Remove()
{
}

int CUserInterface::PrintUI()
{
    return 0;
}

int CUserInterface::Run()
{
    return 0;
}
```



## 11. AddressBook\_OOP.cpp 파일 추가 ( 소스파일 우클릭-추가-새항목 )



12. 실행 구현부분 AddressBook\_00P.cpp 파일 작성

```
#include "CMyList.h"
#include "CUserInterface.h"

int main()
{
    CMyList DB;
    CUserInterface UI(DB);
    UI.Run();

    return 0;
}
```

13. CUserInterface 클래스의 Run 메서드 안에 이전 소스 main() 함수에 보이는 이벤트 반복문 코드를 추가합니다.

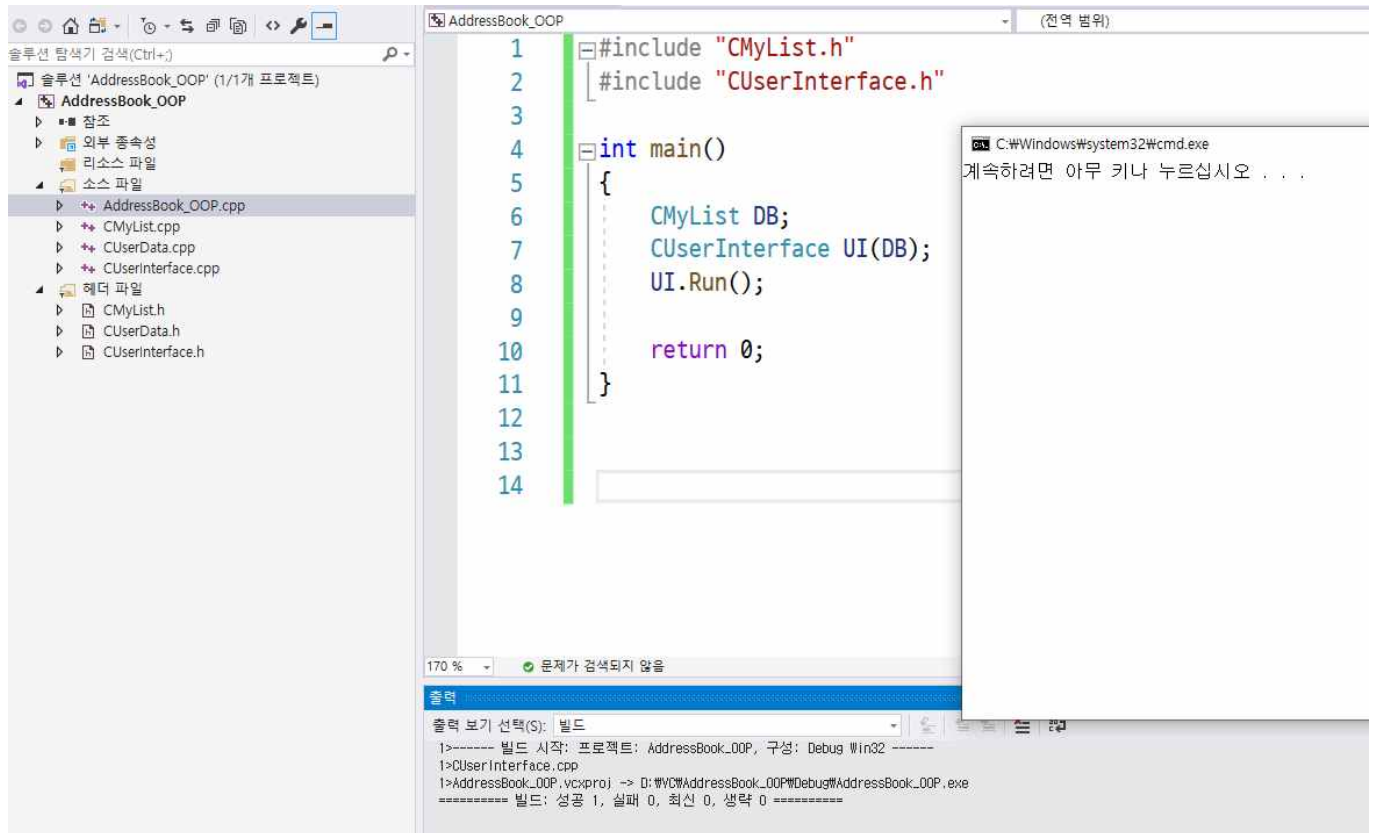
```
int CUserInterface::Run()
{
    int nMenu = 0;

    // 메인 이벤트 반복문
    while ( (nMenu = PrintUI()) != 0 )
    {
        switch (nMenu)
        {
            case 1:
                Add();           // Add
                break;
            case 2:
                Search();        // Search
                break;
            case 3:
                m_List.PrintAll(); // PrintAll
                break;
            case 4:
                Remove();        // Remove
                break;
        }
    }

    return nMenu;
}
```



이제 프로젝트를 빌드한 후 실행하고 결과를 확인합니다.



이제 클래스별로 메서드들을 모두 완성해가는 방식으로 구현하겠습니다.

14. CUserData 클래스의 메서드를 정의합니다.

```
#include "CUserData.h"
#include <cstring>

// nUserDataCounter 정적 변수 0으로 초기화

// 생성자 함수 정의
CUserData::CUserData()
{
    // szName, szPhone 메모리 초기화 ( memset 함수 이용 )
    // pNext 포인터 변수 초기화 ( nullptr )
    // 데이터 생성으로 nUserDataCounter 증가 ( ++ )
}

// 소멸자 함수 정의
CUserData::~CUserData()
{
    // 데이터 삭제로 nUserDataCounter 감소 ( -- )
}
```

15. CMyList 클래스의 메서드들을 모두 정의합니다.

```
#pragma warning(disable:4996)
#include "CMyList.h"
#include <cstdlib>
#include <cstring>
#include <iostream>
using namespace std;

CMyList::CMyList()
{
}

CMyList::~CMyList()
{
    ReleaseList();
}
```

```

void CMyList::ReleaseList()
{
    // 아래 함수를 참고하여 작성
}

/////////////////////////////////////////////////////////////////
// 리스트의 모든 데이터를 삭제하는 함수
//void ReleaseList()
//{
//    USERDATA* pTmp = pHead;
//    USERDATA* pDelete = nullptr;
//
//    while (pTmp != nullptr)
//    {
//        pDelete = pTmp;
//        pTmp = pTmp->pNext;
//
//        delete pDelete;
//    }
//    pHead = nullptr;
//}

CUserData* CMyList::FindNode(const char* pszName)
{
    // 아래 함수를 참고하여 작성
    return nullptr;
}

/////////////////////////////////////////////////////////////////
// 리스트에서 이름으로 특정 노드를 검색하는 함수
//USERDATA* FindNode(char* pszName)
//{
//    USERDATA* pTmp = pHead;
//    while (pTmp != NULL)
//    {
//        if (strcmp(pTmp->szName, pszName) == 0)
//            return pTmp;
//
//        // 다음 노드로 이동
//        pTmp = pTmp->pNext;
//    }
//    // 일치하는 데이터를 찾지 못한 경우
//    return NULL;
//}

```

AddNewNode 함수 내부에서 sprintf() 대신에 문자열 복사 strcpy() 함수로 사용

```
int CMyList::AddNewNode(const char* pszName, const char* pszPhone)
{
    // 아래 함수를 참고하여 작성
    return 0;
}

////////////////////////////////////
//int AddNewNode(char* pszName, char* pszPhone)
//{
//    USERDATA* pNewUser = NULL;
//
//    // 같은 이름이 이미 존재하는지 확인한다.
//    if (FindNode(pszName) != NULL)        return 0;
//
//    // 메모리를 확보한다.
//    pNewUser = new USERDATA;
//    memset(pNewUser, 0, sizeof(USERDATA));
//
//    // 메모리에 값을 저장한다.
//    sprintf(pNewUser->szName, "%s", pszName);
//    sprintf(pNewUser->szPhone, "%s", pszPhone);
//    pNewUser->pNext = nullptr;
//
//    // 새로운 노드가 리스트 마지막에 삽입된다.
//    if (pTail == nullptr) { // 리스트가 비었을때, 만든노드가 처음 노드라면
//        pHead = pNewUser;
//    }
//    else {
//        pTail->pNext = pNewUser;
//    }
//
//    pTail = pNewUser;
//
//    return 1;
//}
```

```

void CMyList::PrintAll()
{
    // 아래 함수를 참고하여 작성
}
/////////////////////////////////////////////////////////////////
// 리스트에 들어있는 모든 데이터를 화면에 출력하는 함수
//void PrintAll()
//{
//    USERDATA* pTmp = pHead;
//
//    while (pTmp != nullptr)
//    {
//        cout << "[" << pTmp << "]" " << pTmp->szName << "\t"
//            << pTmp->szPhone << " [" << pTmp->pNext << "]\n";
//        pTmp = pTmp->pNext;
//    }
//
//    cin.get();
//}

```



```

int CMyList::RemoveNode(const char* pszName)
{
    // 아래 함수를 참고하여 작성
    return 0;
}

/////////////////////////////////////////////////////////////////
// 특정 노드를 검색하고 삭제하는 함수
//int RemoveNode(char* pszName)
//{
//    USERDATA* pPrevNode = nullptr;
//    USERDATA* pDelete = pHead;
//
//    while (pDelete != nullptr)
//    {
//        if (strcmp(pDelete->szName, pszName) == 0)
//        {
//            if (pPrevNode == nullptr) { // 삭제 노드가 처음 노드일때!
//                pHead = pDelete->pNext;
//            }
//            else {
//                if (pDelete == pTail) // 삭제 노드가 마지막 노드일때!
//                    pTail = pPrevNode;
//                pPrevNode->pNext = pDelete->pNext;
//            }
//            delete pDelete;
//            return 1;
//        }
//
//        pPrevNode = pDelete;
//        pDelete = pDelete->pNext;
//    }
//    return 0;
//}

```



16. CUserInterface 클래스의 메서드들을 모두 정의합니다.

```
#include "CUserInterface.h"
#include <iostream>
using namespace std;

CUserInterface::CUserInterface(CMyList& rList)
    : m_List(rList)
{
}
```

```
CUserInterface::~~CUserInterface()
{
}
```

AddNewNode 함수는 m\_List 객체 멤버 함수 호출

```
void CUserInterface::Add()
{
    // 아래 함수를 참고하여 작성
}

////////////////////////////////////
// 이름을 입력받아 리스트에 추가하는 함수
//void Add()
//{
//    char szName[32] = { 0 };
//    char szPhone[32] = { 0 };
//
//    cout << "Input name : ";
//    fflush(stdin);
//    cin >> szName;
//
//    cout << "Input phone number : ";
//    fflush(stdin);
//    cin >> szPhone;
//
//    // 실제로 리스트에 추가한다.
//    AddNewNode(szName, szPhone);
//}
```

RemoveNode 함수는 m\_List 객체 멤버 함수 호출

```
void CUserInterface::Remove()
{
    // 아래 함수를 참고하여 작성
}

/////////////////////////////////////////////////////////////////
// 이름을 입력받아 자료를 검색하고 삭제하는 함수
//void Remove()
//{
//    char szName[32] = { 0 };
//    USERDATA* pNode = NULL;
//
//    cout << "Input name : ";
//    fflush(stdin);
//    cin >> szName;
//
//    RemoveNode(szName);
//}

int CUserInterface::PrintUI()
{
    // 아래 함수를 참고하여 작성
    return 0;
}

/////////////////////////////////////////////////////////////////
// 메뉴를 출력하는 User Interface(UI) 함수
//int PrintUI()
//{
//    int nInput = 0;
//
//    cout << "\n[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit\n:";
//
//    // 사용자가 선택한 메뉴의 값을 반환한다.
//    cin >> nInput;
//    return nInput;
//}
```

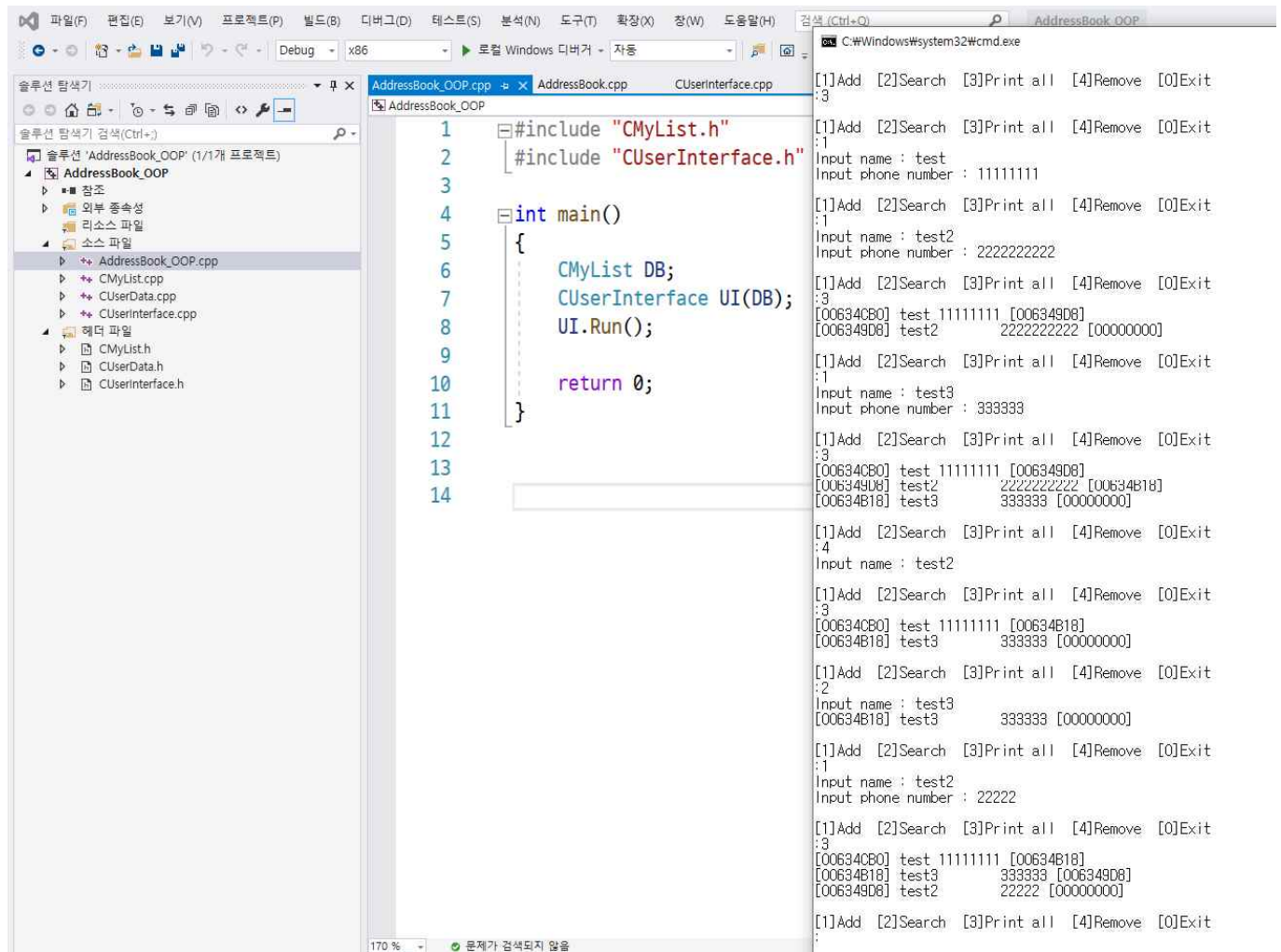
FindNode 함수는 m\_List 객체 멤버 함수 호출

pNode 객체 멤버변수 szName, szPhone, pNext는 외부 클래스인 CUserInterface에서 접근 불가능  
CUserData 객체에 만들어 놓은 Get 함수를 이용

```
void CUserInterface::Search()
{
    // 아래 함수를 참고하여 작성
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// 특정 노드를 검색하는 함수
//void Search()
//{
//    char szName[32] = { 0 };
//    USERDATA* pNode = NULL;
//
//    cout << "Input name : ";
//    fflush(stdin);
//    cin >> szName;
//
//    pNode = FindNode(szName);
//    if (pNode != NULL)
//    {
//        cout << "[" << pNode << "]" " << pNode->szName << "\t"
//            << pNode->szPhone << " [" << pNode->pNext << "]\n";
//    }
//    else
//    {
//        cout << "ERROR: 데이터를 찾을 수 없습니다.";
//    }
//
//    cin.get();
//}
```

이제 프로젝트를 빌드한 후 실행하고 결과를 확인합니다.



```
1 #include "CMYList.h"
2 #include "CUserInterface.h"
3
4 int main()
5 {
6     CMYList DB;
7     CUserInterface UI(DB);
8     UI.Run();
9
10    return 0;
11 }
12
13
14
```

```
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:3
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:1
Input name : test
Input phone number : 11111111
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:1
Input name : test2
Input phone number : 2222222222
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:3
[00634CB0] test 11111111 [006349D8]
[006349D8] test2 2222222222 [00000000]
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:1
Input name : test3
Input phone number : 333333
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:3
[00634CB0] test 11111111 [006349D8]
[006349D8] test2 2222222222 [00634B18]
[00634B18] test3 333333 [00000000]
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:4
Input name : test2
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:3
[00634CB0] test 11111111 [00634B18]
[00634B18] test3 333333 [00000000]
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:2
Input name : test3
[00634B18] test3 333333 [00000000]
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:1
Input name : test2
Input phone number : 2222
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:3
[00634CB0] test 11111111 [00634B18]
[00634B18] test3 333333 [006349D8]
[006349D8] test2 22222 [00000000]
[1]Add [2]Search [3]Print all [4]Remove [0]Exit
:
```

파일 저장, 읽기 함수를 제외하고 동일하게 작동합니다.