■ 주소록 예제

지금부터 살펴볼 예제는 전형적인 단일연결리스트 기반의 주소록 프로그램입니다.
리스트를 이루는 각 요소는 USERDATA 구조체 안에 구현했으며 여러 전역함수들로 프로그램 기능이 만들어집니다. 그런데 프로그램을 분석해보면 다음 두 종류의 함수로 나누어진다는 것을 알 수 있습니다.

✓ 리스트를 직접 다루는 함수
✓ 화면 및 사용자 인터페이스 함수

그리고 이 함수들은 모두 USERDATA 구조체를 중심으로 한 데 묶입니다. 어떤 함수는 정보를 추가하고 또 어떤 함수는 정보를 읽어서 출력합니다.

자 이제 여러분은 AddressBook이라는 새 프로젝트를 생성하고 다음 소개하는 예제를 작성해서 만들고 빌드한 후 실행해봅니다.

```
#pragma warning(disable:4996)
// AddressBook.cpp
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;

////////////////////////////////////////////////////////////////////////
// 주소록이 저장될 데이터 파일
#define DATA_FILE_NAME  "Address.dat"

////////////////////////////////////////////////////////////////////////
struct USERDATA
{
    char    szName[32];     // 이름
    char    szPhone[32];    // 전화번호
    USERDATA* pNext;
};

// Head, Tail 포인터 선언 및 정의
USERDATA* pHead = nullptr;
USERDATA* pTail = nullptr;

////////////////////////////////////////////////////////////////////////
// 함수 선언
USERDATA* FindNode(char* pszName);
int AddNewNode(char* pszName, char* pszPhone);
void Add();
void Search();
void PrintAll();
int RemoveNode(char* pszName);
void Remove();
int PrintUI();
int LoadList(const char* pszFileName);
int SaveList(const char* pszFileName);
void ReleaseList();
```

```
////////////////////////////////////////////////////////////////////////
int main()
{
    int nMenu = 0;
    LoadList(DATA_FILE_NAME);

    // 메인 이벤트 반복문
    while ((nMenu = PrintUI()) != 0) {
        switch (nMenu)
        {
        case 1:     //Add
            Add();
            break;
        case 2:     //Search
            Search();
            break;
        case 3:     //Print all
            PrintAll();
            break;
        case 4:     //Remove
            Remove();
            break;
        }
    }
    // 종료전에 파일로 저장하고 메모리를 해제한다.
    SaveList(DATA_FILE_NAME);
    ReleaseList();
    return 0;
}
// 함수 정의
////////////////////////////////////////////////////////////////////////
// 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;
}
```

```
////////////////////////////////////////////////////////////////////////
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;
}

////////////////////////////////////////////////////////////////////////
// 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);
}
```

```
////////////////////////////////////////////////////////////////////////
// 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();
}

////////////////////////////////////////////////////////////////////////
// 리스트에 들어있는 모든 데이터를 화면에 출력하는 함수
void PrintAll()
{
    USERDATA *pTmp = pHead;
    while(pTmp != NULL)
    {
        cout << "[" << pTmp << "] " << pTmp->szName << "\t"
            << pTmp->szPhone << " [" << pTmp->pNext << "]\n";
        pTmp = pTmp->pNext;
    }

    cin.get();
}
```

```
////////////////////////////////////////////////////////////////////////
// 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;
}
////////////////////////////////////////////////////////////////////////
// 이름을 입력받아 자료를 검색하고 삭제하는 함수
void Remove()
{
    char szName[32] = {0};
    USERDATA *pNode = NULL;

    cout << "Input name : ";
    fflush(stdin);
    cin >> szName;

    RemoveNode(szName);
}
```

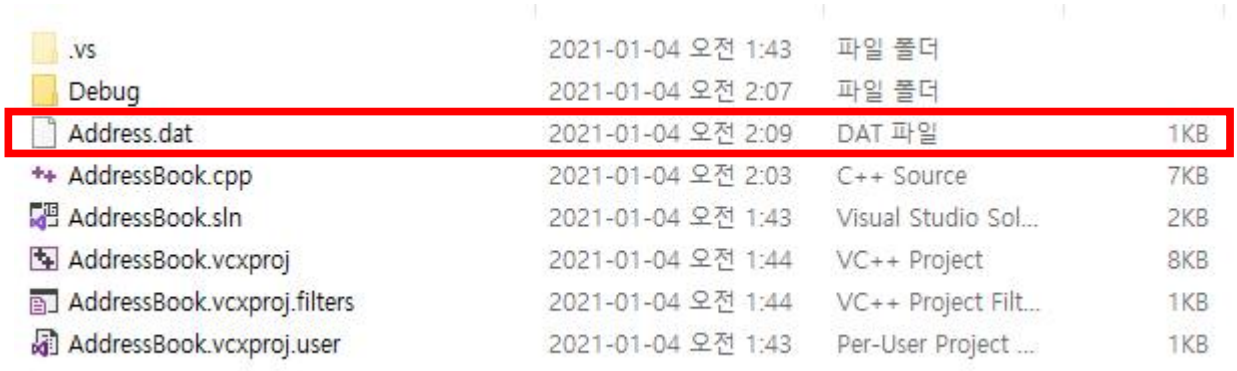
```
////////////////////////////////////////////////////////////////////////
// 메뉴를 출력하는 User Interface(UI) 함수
int PrintUI()
{
    int nInput = 0;

    cout << "\n[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit\n:";

    // 사용자가 선택한 메뉴의 값을 반환한다.
    cin >> nInput;
    return nInput;
}

////////////////////////////////////////////////////////////////////////
// 데이터 파일에서 노드들을 읽어와 리스트를 완성하는 함수
int LoadList(const char* pszFileName)
{
    FILE* fp = NULL;
    USERDATA user = { 0 };

    fp = fopen(pszFileName, "rb");
    if (fp == NULL)
        return 0;

    ReleaseList();

    while (fread(&user, sizeof(USERDATA), 1, fp))
    {
        AddNewNode(user.szName, user.szPhone);
    }

    fclose(fp);
    return 0;
}
```

```
////////////////////////////////////////////////////////////////////////
// 리스트 형태로 존재하는 정보를 파일에 저장하는 함수
int SaveList(const char* pszFileName)
{
    FILE* fp = NULL;
    USERDATA* pTmp = pHead;

    fp = fopen(pszFileName, "wb");
    if (fp == NULL) {
        cout << "ERROR: 리스트 파일을 쓰기모드로 열지 못했습니다.";
        cin.get();
        return 0;
    }

    while (pTmp != NULL)
    {
        if (fwrite(pTmp, sizeof(USERDATA), 1, fp) != 1)
        {
            cout << "ERROR: " << pTmp->szName
                << "에 대한 정보를 저장하는데 실패했습니다.\n";
        }
        pTmp = pTmp->pNext;
    }

    fclose(fp);
    return 1;
}
////////////////////////////////////////////////////////////////////////
// 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;
}
```

// 실행화면

```
[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:1
Input name : test1
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
[01264BD8] test1        11111111 [01264900]
[01264900] test2        2222222222 [00000000]

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:1
Input name : test3
Input phone number : 333333333333

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:3
[01264BD8] test1        11111111 [01264900]
[01264900] test2        2222222222 [01264A40]
[01264A40] test3        333333333333 [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
[01264BD8] test1        11111111 [01264A40]
[01264A40] test3        333333333333 [00000000]

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:1
Input name : test4
Input phone number : 44444444444

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:3
[01264BD8] test1        11111111 [01264A40]
[01264A40] test3        333333333333 [01264900]
[01264900] test4        44444444444 [00000000]

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:2
Input name : test2
ERROR: 데이터를 찾을 수 없습니다.
[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:2
Input name : test4
[01264900] test4        44444444444 [00000000]

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:0
계속하려면 아무 키나 누르십시오 . . .
```

// Address.dat 파일 생성확인!



.vs	2021-01-04 오전 1:43	파일 폴더	
Debug	2021-01-04 오전 2:07	파일 폴더	
Address.dat	2021-01-04 오전 2:09	DAT 파일	1KB
AddressBook.cpp	2021-01-04 오전 2:03	C++ Source	7KB
AddressBook.sln	2021-01-04 오전 1:43	Visual Studio Sol...	2KB
AddressBook.vcxproj	2021-01-04 오전 1:44	VC++ Project	8KB
AddressBook.vcxproj.filters	2021-01-04 오전 1:44	VC++ Project Filt...	1KB
AddressBook.vcxproj.user	2021-01-04 오전 1:43	Per-User Project ...	1KB

// 다시 실행해서 저장된 내용 확인!

```
[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:3
[00908BD8] test1        11111111 [0090E688]
[0090E688] test3        333333333333 [00904BD8]
[00904BD8] test4        44444444444 [00000000]

[1]Add  [2]Search  [3]Print all  [4]Remove  [0]Exit
:
```