**<MemoryINI 설명 문서>**

특정 INI 파일을 암호화해서 저장 & 읽기 기능을 수행하는 프로젝트로써 특이한 점은

암호화 로직을 외부에서 정해주도록 되어 있다. 정해주는 방식은 제공되어지는 인터페이스를

호출할 때 특정 암호화 로직을 수행해주는 함수의 포인터를 전달받도록 되어 있다. 해당 함수포인터가 NULL로 전달될 경우 암호화되지 않은 평문으로 저장이 된다.

내부적으로 INI 파일을 관리할 때 백터와 구조체를 이용해 관리를 수행한다.

구조체는 아래와 같다.

**typedef struct \_KEYDATA**

**{**

**TCHAR key[MAX\_LINE];**

**TCHAR value[MAX\_LINE];**

**} KEYDATA, \*PKEYDATA;**

**typedef struct \_INI\_SECTION**

**{**

**TCHAR name[MAX\_LINE];**

**vector<KEYDATA> keys;**

**} INI\_SECTION, \*PINI\_SECTION;**

KEYDATA 구조체는 INI 파일에서 섹션 하위에 있는 실제 값들을 가리키며, INI\_SECTION 구조체의 경우 최상위 구별자인 섹션의 이름과 KEYDATA의 구조체를 가지고 있게 된다.

실제 INI 파일에 적용하면 아래와 같다.

[TEST\_SECTION] 🡺 INI\_SECTION

testKey=value 🡺 KEYDATA

<프로젝트 구성>

1. **AKMemoryIni**

* MemoryIni의 실제 모듈 프로젝트이며, MemoryIni.h 파일과 MemoryIni.cpp 두개의 파일로 구성되어 있다.

1. **CryptIni**

* MemoryIni 모듈을 사용하는 Win32 콘솔용 샘플 프로그램의 프로젝트이며, 간단한 사용 샘플 예제를

볼 수 있다.

1. **CryptIni\_Test**

* 2번 프로젝트와 마찬가지의 샘플프로그램이며, Win32 콘솔이 아닌 MFC 기반의 샘플 프로그램이다.

<인터페이스 설명>

1. **Int SetData( LPTSTR ini\_data, DWORD ini\_size )**

* 특정 ini 파일(암호화되지 않은 ini 파일!!)을 한꺼번에 내부 버퍼에 저장시킬 때 사용되는 인터페이스다.

해당 인터페이스 호출시 ini\_data 버퍼에 있는 내용을 ini\_size 만큼 읽어서 내부 메모리 영역(위에서 설명한 구조체들의 백터)에 저장을 시키게 된다.

해당 함수 호출 후 저장되어 있는 버퍼의 내용을 실질적으로 물리적인 파일에 저장시 “SaveToFile” 이라는 인터페이스를 호출하면 파일로 저장이 되게 된다.

Ini\_data 버퍼는 실제 ini 파일 형식의 값들이 채워져 있어야 하며, ini\_size는 해당 버퍼의 사이즈를 넘겨주어야 한다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int GetData( LPTSTR ini\_data, DWORD ini\_size, PDWORD ini\_written )**

* 1번의 SetData와는 반대의 역할을 하는 인터페이스로써, 내부버퍼에 저장되어 있는 내용을 그대로 주어진 버퍼에 복사해서 출력해주는 인터페이스다.

GetData를 호출해주는 상황은 “SetData”, “LoadFromFile”, “putString”, “putInteger” 와 같이 내부 버퍼를 할당해주는 인터페이스를 호출 한 후 사용하여야 정상적인 값을 얻을 수 있다.

ini\_data는 값을 받아낼 버퍼의 포인터이며, ini\_size는 해당 버퍼의 크기, 마지막으로 ini\_written은 최종적으로 ini\_data 버퍼에 쓰기가 완료된 바이트의 크기를 반환해준다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int SaveToFile( LPCTSTR inifile, INI\_CRYPT encrypt )**

* 1번 인터페이스 설명시 언급한 것과 같이 내부 버퍼에 저장되어 있는 내용을 그대로 실제 물리적인 파일로 저장시켜주는 인터페이스다.

첫번째 인자인 inifile은 실제 저장시킬 ini 파일의 이름을 전달해주어야 하며, 두번째 인자인 encrypt는 함수포인터로써 저장시 해당 함수의 로직을 이용해 암호화 후 파일에 저장하게 된다.

해당 매개변수가 NULL로 전달될 경우 암호화되지 않은 평문이 그대로 ini 파일에 저장되게 된다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int LoadFromFile( LPCTSTR inifile, INI\_CRYPT decrypt )**

* 기본적인 구조는 “SaveToFile” 인터페이스와 비슷하지만 실제 물리적인 파일에서 해당 내용을 읽어들인다는 점이 다르다. “SaveToFile”과 마찬가지로 함수 포인터를 받게 되며 해당 포인터가 NULL 일 경우 마찬가지로 파일에서 읽은 내용을 단순히 버퍼에 읽어들이게 된다.

반대로 Decrypt 인자가 NULL이 아닐 경우 해당 함수를 호출하여 decrypt를 수행한 후 해당 데이터를 내부 버퍼에 저장하게 된다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int getString( LPCTSTR section, LPCTSTR key, LPTSTR pStr, DWORD slen )**

* 내부버퍼에 해당 Section, Key 값이 존재하면 스트링형식의 해당 Value 값을 읽어오는

인터페이스 함수이다.

결과값은 세번째 인자인 pStr로 반환이되며 slen은 pStr 인자가 가질 수 있는 최대 버퍼 크기를 전달해주면 된다. 해당 인터페이스 또한 마찬가지로 “SetData”, “LoadFromFile”, “putString”, “putInteger” 와 같이 내부 버퍼를 할당해주는 인터페이스를 호출 한 후 사용하여야 정상적인 값을 얻을 수 있다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int putString( LPCTSTR section, LPCTSTR key, LPCTSTR pStr )**

* 5번의 “GetString” 인터페이스의 반대 역할을 수행하는 인터페이스이며, 내부버퍼에 해당 SECTION과 KEY에 해당하는 스트링형식의 Value값을 저장하는 역할을 한다.

만약에 기존에 이미 동일한 SECTION, KEY가 존재할 경우 해당 Value값만을 업데이트해주게 된다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int getInteger( LPCTSTR section, LPCTSTR key, LONG \*pInt )**

* 5번 인터페이스와 같은 역할을 수행하지만 읽어오는 Value값은 LONG형의 값이다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.

1. **Int putInteger( LPCTSTR section, LPCTSTR key, LONG value )**

* 6번 인터페이스와 같은 역할을 수행하지만 읽어오는 Value값은 LONG형의 값이다.

리턴값은 0일 경우 성공이며, -1일 경우 실패를 반환한다.