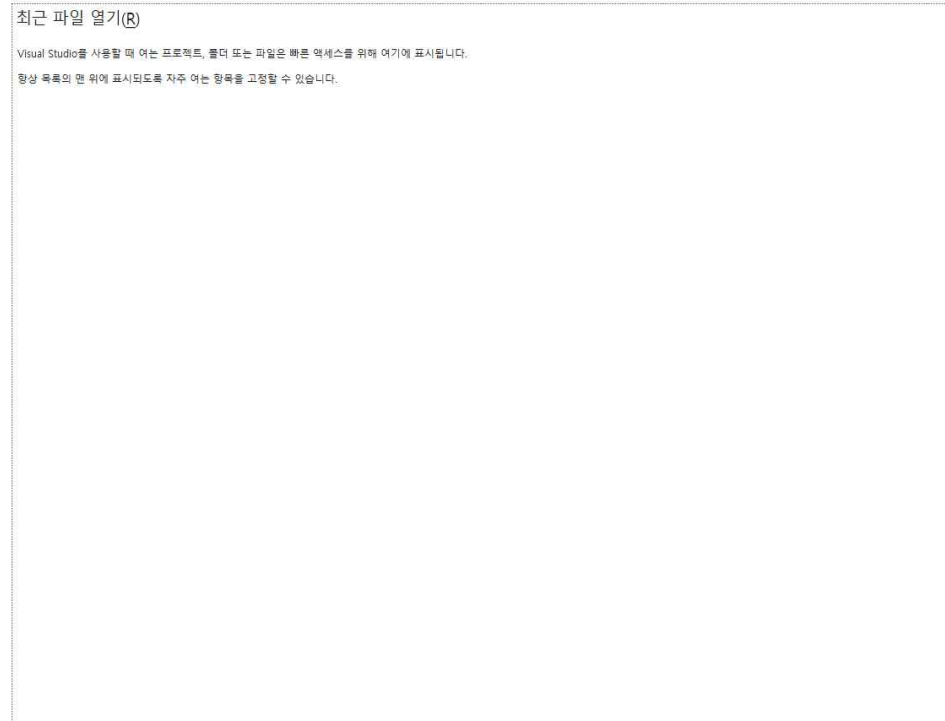


실습과제#1 : 프로젝트 생성 및 초기 코드 작성

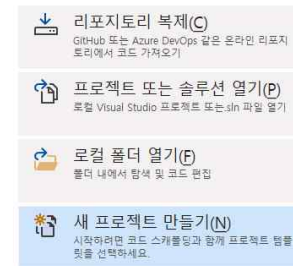
■ Visual Studio 실행 > 새 프로젝트 만들기

Visual Studio를 실행하고 새 프로젝트를 선택합니다.

원하는 작업을 선택하세요.



시작



■ 콘솔 앱 > 다음

새 프로젝트 만들기 창이 열리면 콘솔 앱을 선택한 후 다음을 클릭합니다.

새 프로젝트 만들기

최근 프로젝트 템플릿(R)

콘솔 앱

빈 프로젝트

C++

C++



다음(N)

■ 프로젝트 이름(StringCtrlSample1234) 만들기

프로젝트 이름에 StringCtrlSample+학번 숫자 4자리를 입력한 후 만들기를 클릭합니다.

새 프로젝트 구성

콘솔 앱 C++ Windows 콘솔

프로젝트 이름(N)

StringCtrlSample1234

위치(L)

D:\VC



솔루션 이름(M) ⓘ

StringCtrlSample1234

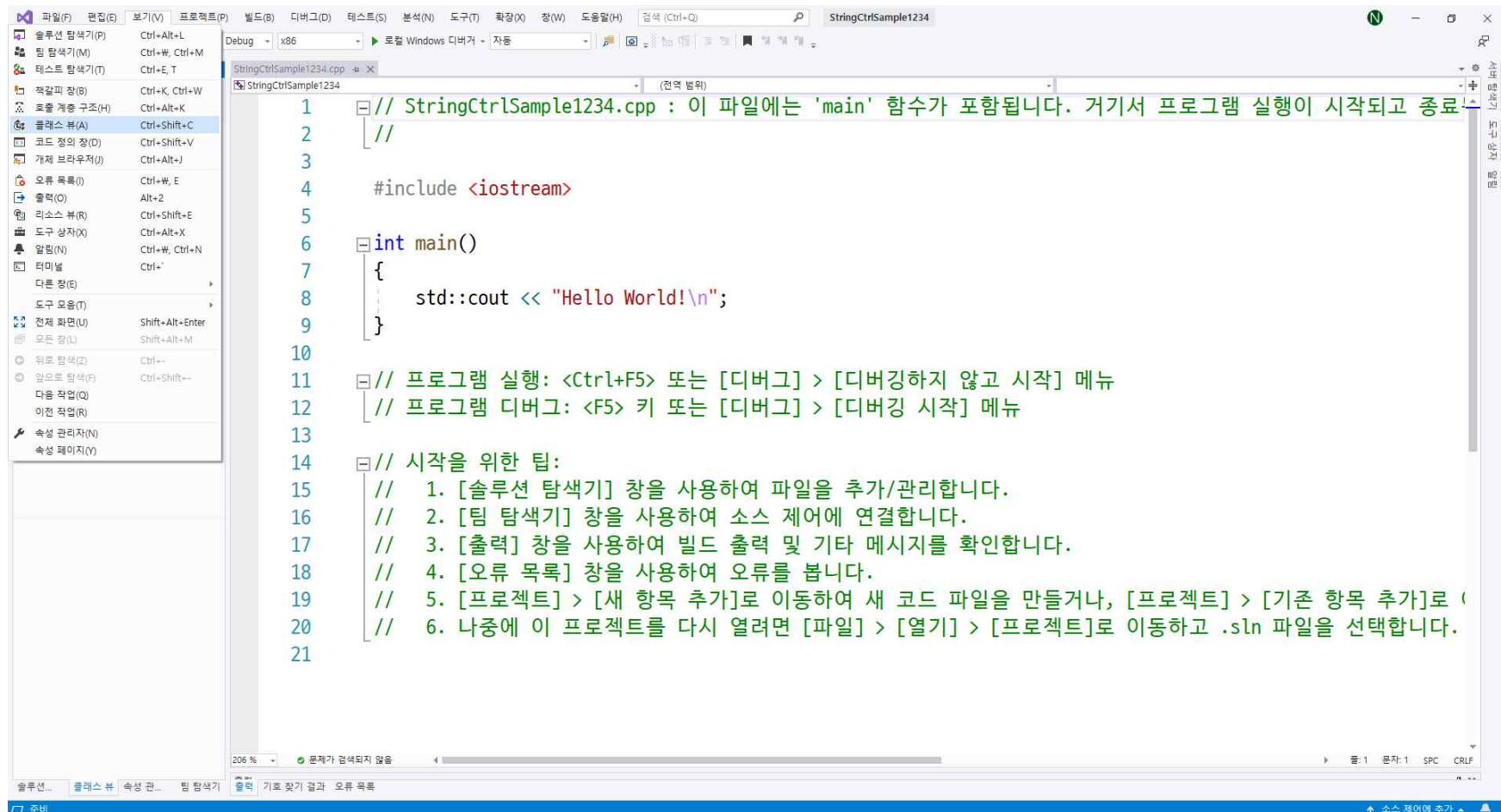
☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(B)

뒤로(B)

만들기(C)

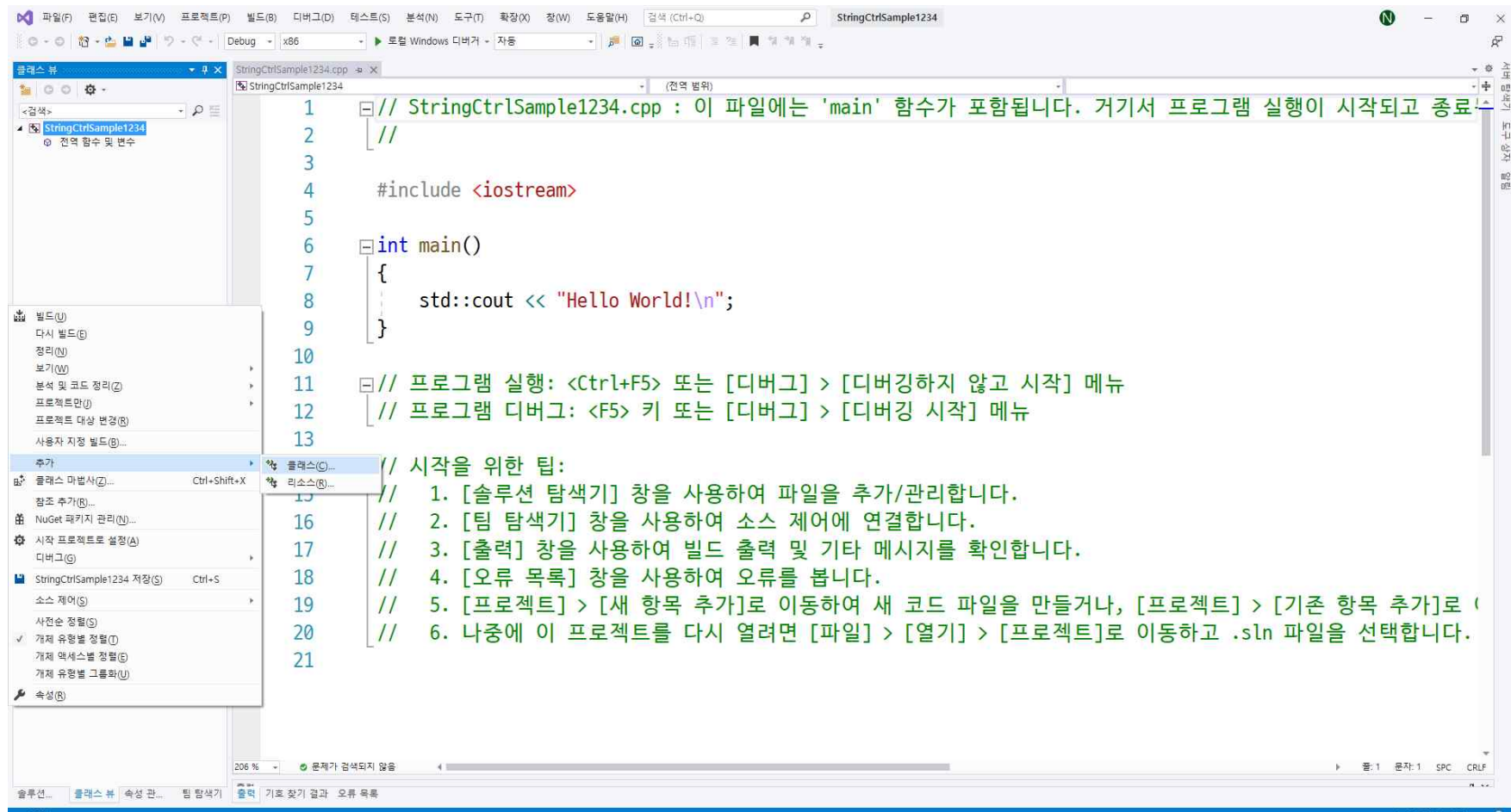
■ 클래스 뷰 선택

보기 메뉴에서 클래스 뷰 선택 (단축기 Ctrl+Shift+C)



■ 클래스 뷰에서 마우스 우클릭 > 추가 > 클래스

클래스 뷰에서 마우스 오른쪽 버튼을 클릭한 후 메뉴에서 추가 > 클래스를 선택합니다.



■ 클래스이름 (CMyString) > 확인

클래스 추가 창이 나타납니다. 클래스 이름에 'CMyString'이라고 입력하면 .h파일, .cpp파일 항목의 이름은 자동으로 결정됩니다. 입력이 끝나면 확인을 클릭합니다.

클래스 추가

클래스 이름(U) .h 파일(F) .cpp 파일(P)

CMyString CMyString.h CMyString.cpp

기본 클래스(B) 액세스(A)

public

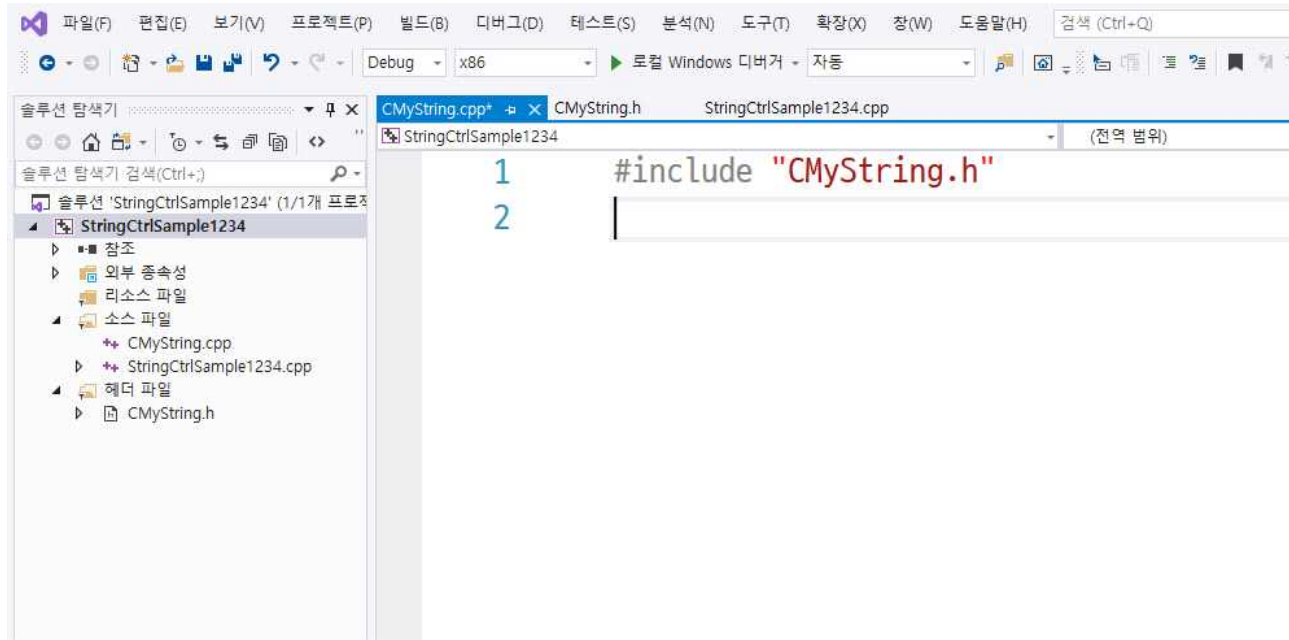
기타 옵션:

☐ 인라인(I)

☐ 관리됨(M)

확인 취소

CMyString.h, CMyString.cpp 파일과 소스코드가 생성된 것을 확인할 수 있습니다.



```
// CMyString.h
#pragma once
class CMyString
{
};
```

```
// CMyString.cpp
#include "CMyString.h"
```

■ 생성자 소멸자 선언

CMyString.h 파일을 아래와 같이 수정

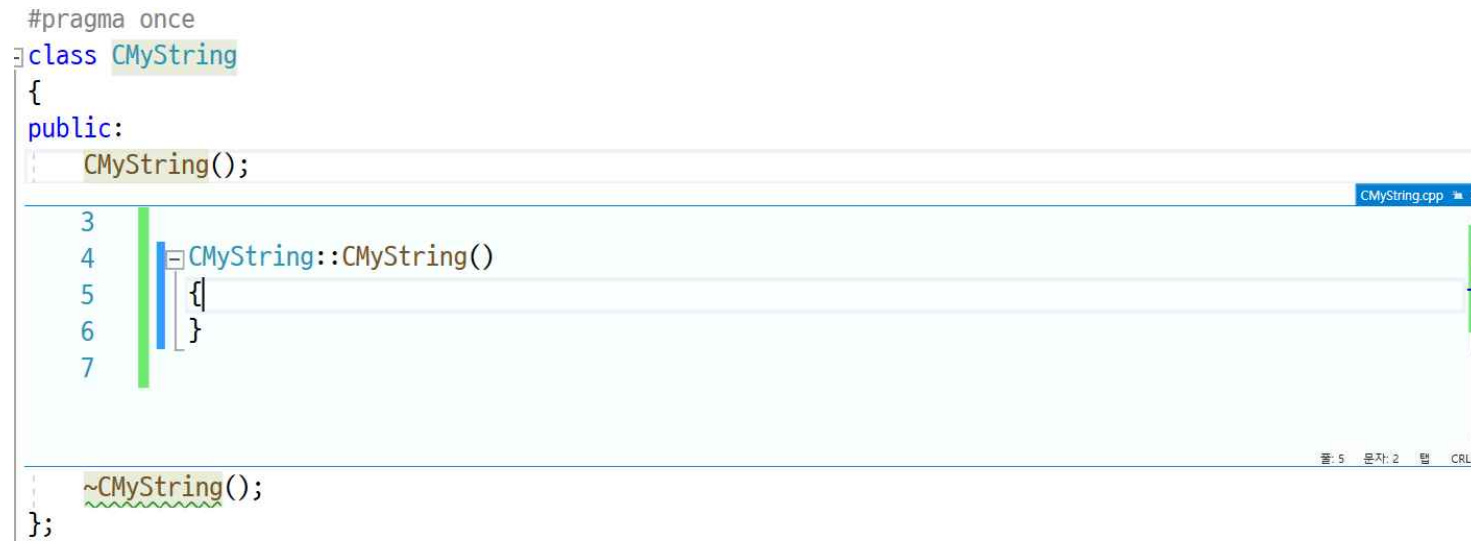
```
#pragma once  
class CMyString  
{  
public:  
    CMyString();  
    ~CMyString();  
};
```


■ 생성자 정의 추가

생성자 CMyString();를 클릭하고 Alt+Enter 눌러서 CMyString 에서 'CMyString.cpp'정의 파일 만들기 선택



아래와 같이 정의파일에 추가된 거 확인하고 저장!



■ 소멸자 정의 추가

소멸자 `~CMyString();`를 클릭하고 `Alt+Enter` 눌러서 `~CMyString` 에서 '`CMyString.cpp`'정의 파일 만들기 선택

```
#pragma once
class CMyString
{
public:
    CMyString();
    ~CMyString();
};
```

`~CMyString`에서 '`CMyString.cpp`' 정의 만들기(D)
'`~CMyString`' 시그니처를 클립보드로 복사(C)

아래와 같이 정의파일에 추가된 거 확인하고 저장!

```
#pragma once
class CMyString
{
public:
    CMyString();
    ~CMyString();
};
```

`CMyString.cpp`

```
7
8  CMyString::~CMyString()
9  {
10 }
11
```

줄: 8 문자: 1 탭: CRLF

■ CMyString.cpp 파일 확인

생성자와 소멸자가 정의된 것을 확인

```
_scalar.cpp  CMyString.cpp  CMyString.h

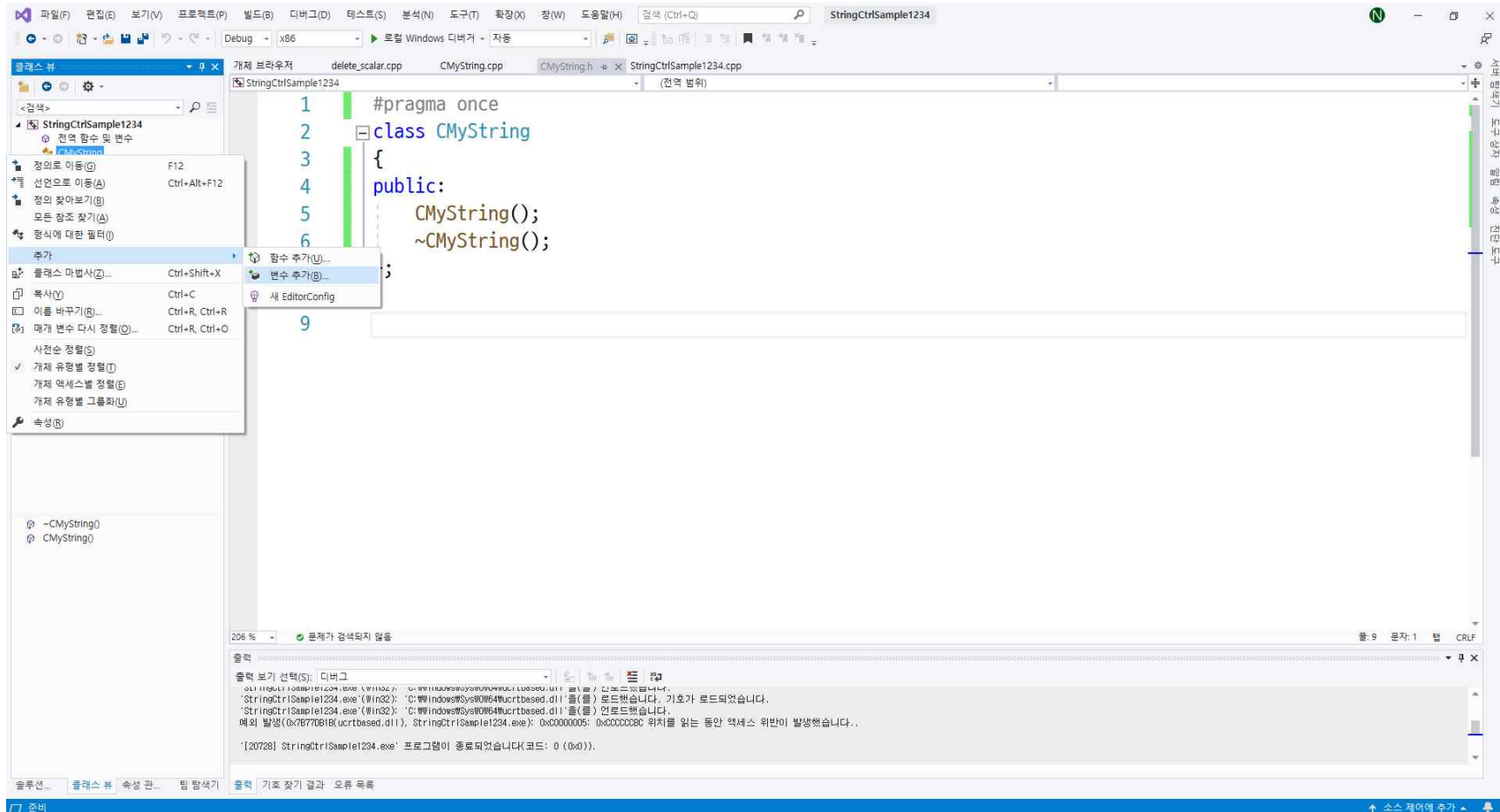
#include "CMyString.h"

CMyString::CMyString()
{
}

CMyString::~CMyString()
{
}
```

■ 멤버 변수 추가

클래스 뷰 창에 있는 StringCtrlSample 항목을 열어 CMyString 항목을 마우스 오른쪽 버튼으로 클릭한 후 메뉴에서 추가 > 변수 추가를 선택합니다.

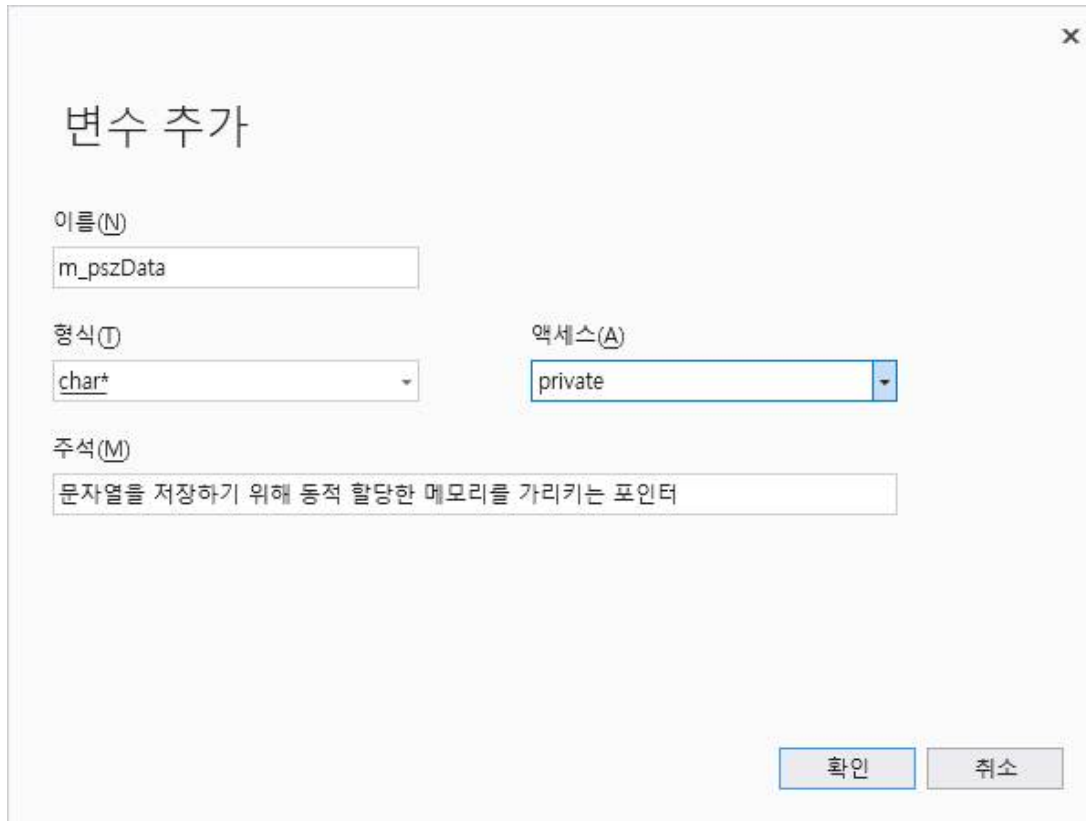


■ 멤버 변수 추가(char* m_pszData)

변수 추가 창이 나타나면 이름 항목은 'm_pszData'을, 형식 항목은 'char*'을 직접 입력합니다.

액세스 항목은 private를 선택합니다.

주석에는 '문자열을 저장하기 위해 동적 할당된 메모리를 가리키는 포인터'라고 입력한 후 확인을 클릭합니다.



변수 추가

이름(N)
m_pszData

형식(T)
char*

액세스(A)
private

주석(M)
문자열을 저장하기 위해 동적 할당된 메모리를 가리키는 포인터

확인 취소

CMyString.h파일에 소스코드가 추가된 것을 확인할 수 있습니다.

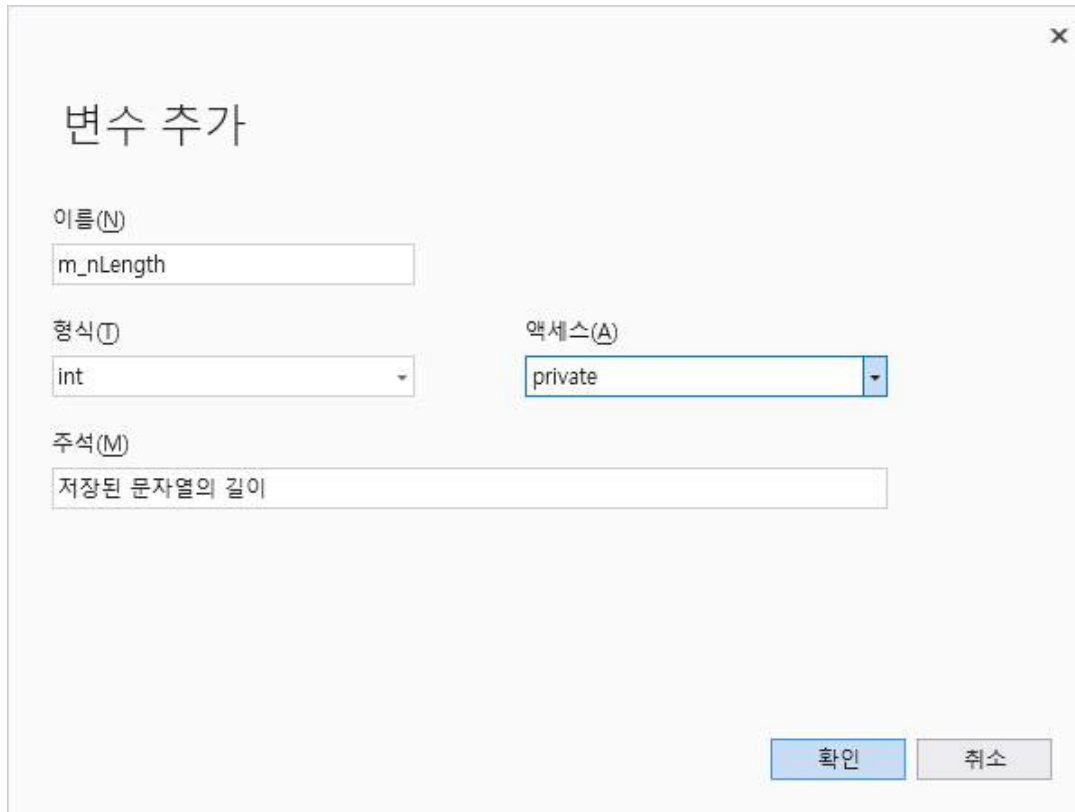
```
#pragma once
class CMyString
{
public:
    CMyString();
    ~CMyString();
private:
    // 문자열을 저장하기 위해 동적 할당한 메모리를 가리키는 포인터
    char* m_pszData;
};
```

■ 멤버 변수 추가(int m_nLength)

변수 추가 창이 나타나면 이름 항목은 'm_nLength'을, 형식 항목은 'int'을 직접 입력합니다.

액세스 항목은 private를 선택합니다.

주석에는 '저장된 문자열의 길이'라고 입력한 후 확인을 클릭합니다.



변수 추가

이름(N)
m_nLength

형식(T)
int

액세스(A)
private

주석(M)
저장된 문자열의 길이

확인 취소

CMyString.h파일에 소스코드가 추가된 것을 확인할 수 있습니다.

```
#pragma once
class CMyString
{
public:
    CMyString();
    ~CMyString();
private:
    // 문자열을 저장하기 위해 동적 할당된 메모리를 가리키는 포인터
    char* m_pszData;
    // 저장된 문자열의 길이
    int m_nLength;
};
```


■ 멤버 변수 초기화

CMyString.cpp 파일 생성자에 초기화 목록을 이용하여 멤버변수를 초기화 합니다.

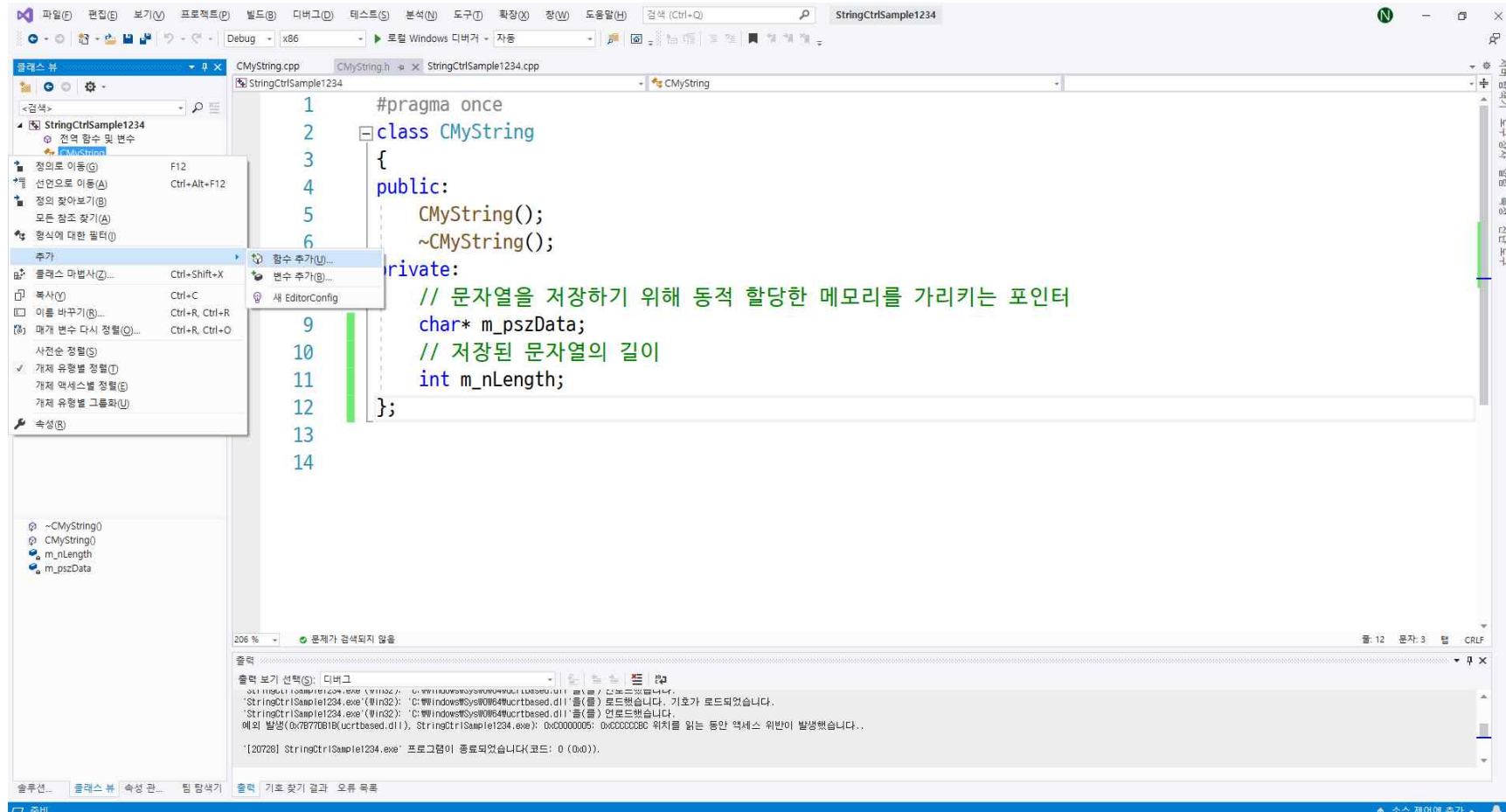
```
#include "CMyString.h"
```

```
≡ CMyString::CMyString()  
{  
    :m_pszData(nullptr)  
    , m_nLength(0)  
}  
}
```

```
≡ CMyString::~~CMyString()  
{  
}  
}
```

■ 멤버 함수 추가

클래스 뷰 창에 있는 StringCtrlSample 항목을 열어 CMyString 항목을 마우스 오른쪽 버튼으로 클릭한 후 메뉴에서 추가 > 함수 추가를 선택합니다.



- **멤버 함수 추가** : `int SetString(const char* pszParam);` // 매개변수로 받은 문자열을 멤버변수에 저장하는 함수
매개변수로 넘어온 문자열을 클래스 멤버변수인 `m_pszData`에 저장하는 함수

함수 추가

함수 이름(U) : SetString

반환 형식(Y) : int

액세스(A) : public

.cpp 파일(F) : CMyString.cpp

매개 변수(P) : const char* pszParam

주석(M)

기타 옵션:

- ☐ 인라인(I)
- ☐ 정적(S)
- ☐ 시각적 개체(V)
- ☐ 순수(P)

확인 취소

1. 함수이름 : SetString
2. 반환형식 : int
3. 액세스 : public
4. .cpp 파일 : CMyString.cpp
5. 매개변수 : 녹색 + 버튼 누르고
`const char* pszParam` 입력

확인버튼 클릭

CMyString.h, CMyString.cpp 파일에 소스코드가 추가된 것을 확인할 수 있습니다.

```
#pragma once
class CMyString
{
public:
    CMyString();
    ~CMyString();
private:
    // 문자열을 저장하기 위해 동적 할당된 메모리를 가리키는 포인터
    char* m_pszData;
    // 저장된 문자열의 길이
    int m_nLength;
public:
    int SetString(const char* pszParam);
};
```

```
#include "CMyString.h"
```

```
CMyString::CMyString()
    :m_pszData(nullptr)
    , m_nLength(0)
{
}
```

```
CMyString::~CMyString()
{
}
```

```
int CMyString::SetString(const char* pszParam)
{
    // TODO: 여기에 구현 코드 추가.
    return 0;
}
```

- **멤버 함수 추가** : `const char* GetString();` // 멤버변수에 저장된 문자열을 가져오는 함수
문자열이 저장된 포인터 멤버변수 `m_pszData`를 반환하는 함수

함수 추가

함수 이름(U) : GetString

반환 형식(Y) : const char*

액세스(A) : public

.cpp 파일(F) : CMyString.cpp

매개 변수(P)

주석(M)

기타 옵션:

- ☐ 인라인(I)
- ☐ 정적(S)
- ☐ 시각적 개체(V)
- ☐ 순수(P)

확인 취소

1. 함수이름 : GetString
2. 반환형식 : const char*
3. 액세스 : public
4. .cpp 파일 : CMyString.cpp
5. 매개변수 : 없음

확인버튼 클릭

■ 멤버 함수 추가 : void Release();

멤버변수 m_pszData에 할당된 메모리를 해제하고 NULL로 초기화



함수 추가

함수 이름(U) : Release

반환 형식(Y) : void

액세스(A) : public

.cpp 파일(F) : CMyString.cpp

매개 변수(P)

주석(M)

기타 옵션:

- ☐ 인라인(I)
- ☐ 정적(S)
- ☐ 시각적 개체(V)
- ☐ 순수(P)

확인 취소

1. 함수이름 : Release
2. 반환형식 : void
3. 액세스 : public
4. .cpp 파일 : CMyString.cpp
5. 매개변수 : 없음

확인버튼 클릭

CMyString.h, CMyString.cpp 파일에 소스코드가 추가된 것을 확인할 수 있습니다.

```
#pragma once
class CMyString
{
public:
    CMyString();
    ~CMyString();
private:
    // 문자열을 저장하기 위해 동적 할당된 메모리를 가리키는 포인터
    char* m_pszData;
    // 저장된 문자열의 길이
    int m_nLength;
public:
    int SetString(const char* pszParam);
    const char* GetString();
    void Release();
};
```

```
#include "CMyString.h"

CMyString::CMyString()
    :m_pszData(nullptr)
    , m_nLength(0)
{
}

CMyString::~CMyString()
{
}


int CMyString::SetString(const char* pszParam)
{
    // TODO: 여기에 구현 코드 추가.
    return 0;
}

const char* CMyString::GetString()
{
    // TODO: 여기에 구현 코드 추가.
    return nullptr;
}

void CMyString::Release()
{
    // TODO: 여기에 구현 코드 추가.
}
```

■ StringCtrlSample1234.cpp 파일 메인 함수 작성 (사용자 코드 작성)

```
#include "CMyString.h" // CMyString 클래스 사용을 위해 헤더파일 추가
using namespace std; // cout 객체 사용을 위해 std namespace 사용
// 메인 함수 내용은 아래 확인
```



```
CMyString.cpp  CMyString.h  StringCtrlSample1234.cpp  (전역 범위)
StringCtrlSample1234
1  // StringCtrlSample1234.cpp : 이 파일에는 'main' 함수가 포함됩니다.
2  // 거기서 프로그램 실행이 시작되고 종료됩니다.
3
4  #include <iostream>
5  #include "CMyString.h"
6  using namespace std;
7
8  int main()
9  {
10     CMyString strData;
11     strData.SetString("Hello");
12     cout << strData.GetString() << endl;
13     return 0;
14 }
15
```

컴파일하고 하고 실행해보면 커서만 깜박거릴 것임
strData.GetString() 함수가 NULL을 반환하기 때문입니다.

아래와 같은 원하는 실행결과를 위해 함수를 정의해봅시다!!!

 C:\Windows\system32\cmd.exe

Hello

계속하려면 아무 키나 누르십시오 . . .

■ SetString() 메서드를 정의 해보세요!

```
int CMyString::SetString(const char* pszParam)
{
    // TODO: 여기에 구현 코드 추가.

    return 0;
}
```

주의사항

1. 매개변수로 전달된 문자열의 길이를 측정하고 m_nLength에 저장합니다.
2. 매개변수로 전달된 문자열이 저장될 수 있는 메모리를 동적 할당합니다.(new 연산자를 이용할 것)
3. 동적 할당한 메모리에 문자열을 저장(m_pszData)합니다.
4. 매개변수가 NULL이거나 문자열의 길이가 0 인 경우를 고려해야 합니다.
5. 여기서 동적할당한 메모리는 언제 어디서 해제하는지 생각하고 대응합니다.
6. 사용자가 다음예와 같이 이 함수를 2회 호출하는 경우를 생각하고 대응합니다.

```
strData.SetString("Hello");
strData.SetString("World");
```

■ GetString() 메서드 정의!

멤버 변수 m_pszData을 반환

```
const char* CMyString::GetString()  
{  
    return m_pszData;  
}
```

■ Release() 메서드를 정의 해보세요!

```
void CString::Release()  
{  
    // TODO: 여기에 구현 코드 추가.  
}
```

주의사항

1. m_pszData라는 멤버 변수가 가리키는 메모리를 해제합니다.(delete 연산자를 이용할 것)
2. 사용자 코드에서 접근이 허용된 메서드들은 무엇이든 호출할 수 있으며 그 순서는 임의로 달라질수 있습니다. 가령 Release() 메서드를 호출한 직후 SetString() 함수를 호출할 수 있는 것이죠. 이같은 상황을 고려하면서 코드를 작성합니다.

처음이라 내용이 복잡하게 느껴질 수 있습니다. 너무 복잡하다 하더라도 일단 도전해보기 바랍니다.!!!