MFC 기초 클래스

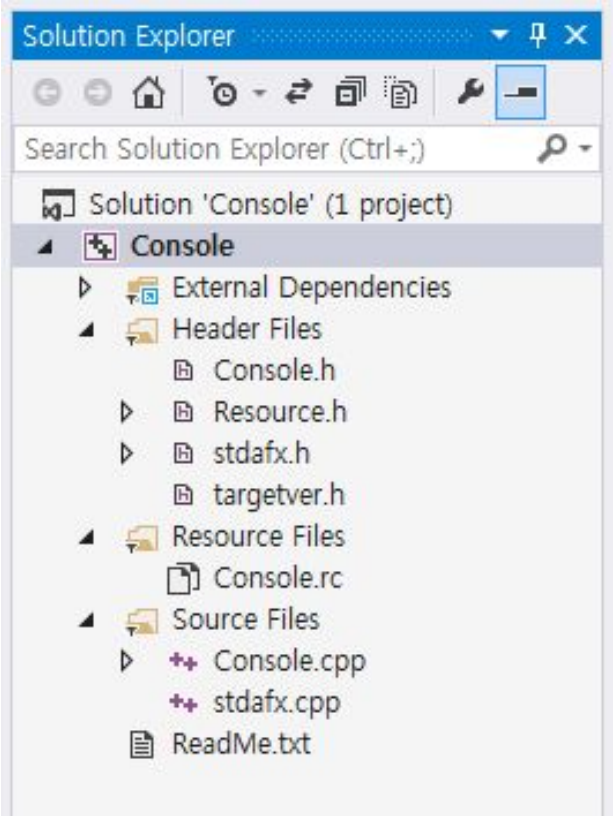
(6주차)

학습개요

- 학습 목표
  - MFC 응용 프로그램의 전체 파일 구성과 파일의 역할을 이해한다.
  - 유틸리티 클래스 객체 생성 및 사용법을 익힌다.
  - 배열, 리스트, 맵 클래스 동작 원리와 사용법을 익힌다.

- 학습 내용
  - MFC 클래스 실습을 위한 준비
  - 유틸리티 클래스
  - 집합 클래스
  - 실습

MFC 콘솔 응용 프로그램 분석 (1)



- 프로젝트 구성 파일
  - stdafx.h, stdafx.cpp : 사전에 컴파일된 헤더 정보를 제공
  - Console.rc : 리소스 스크립트 파일
  - Resource.h : 리소스 ID를 코드에서 접근할 수 있도록 상수 정의
  - Console.h, Console.cpp : 프로그램 구현 코드
  - targetver.h : 개발 대상이 되는 윈도우 운영체제의 버전 지정

MFC 콘솔 응용 프로그램 분석 (2)

- Console 예제 코드 (1/2)

```
#include "stdafx.h"
#include "Console.h"
...
CWinApp theApp;
using namespace std;

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[]) {
    int nRetCode = 0;

    HMODULE hModule = ::GetModuleHandle(NULL);
    if (hModule != NULL) {
        if (!AfxWinInit(hModule, NULL, ::GetCommandLine(), 0)) {
            _tprintf(_T("Fatal Error: MFC initialization failed\n"));
            nRetCode = 1;
```

MFC 콘솔 응용 프로그램 분석 (3)

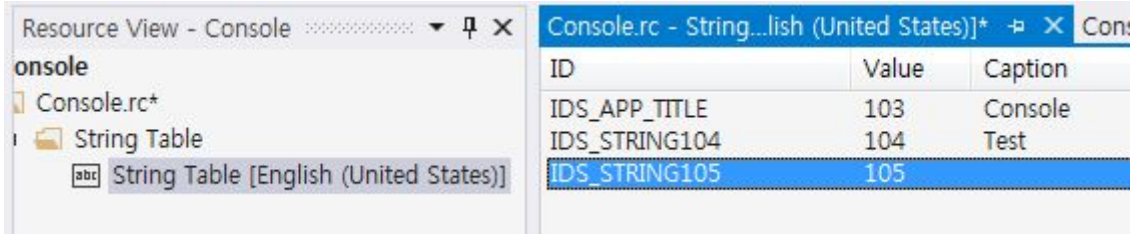
- Console 예제 코드 (2/2)

```
        } else {
            CString str;
            str.LoadString(IDS_APP_TITLE);
            _tprintf(_T("Hello from %s!\n"), str);
        }
    } else {
        _tprintf(_T("Fatal Error: GetModuleHandle failed\n"));
        nRetCode = 1;
    }

    return nRetCode;
}
```

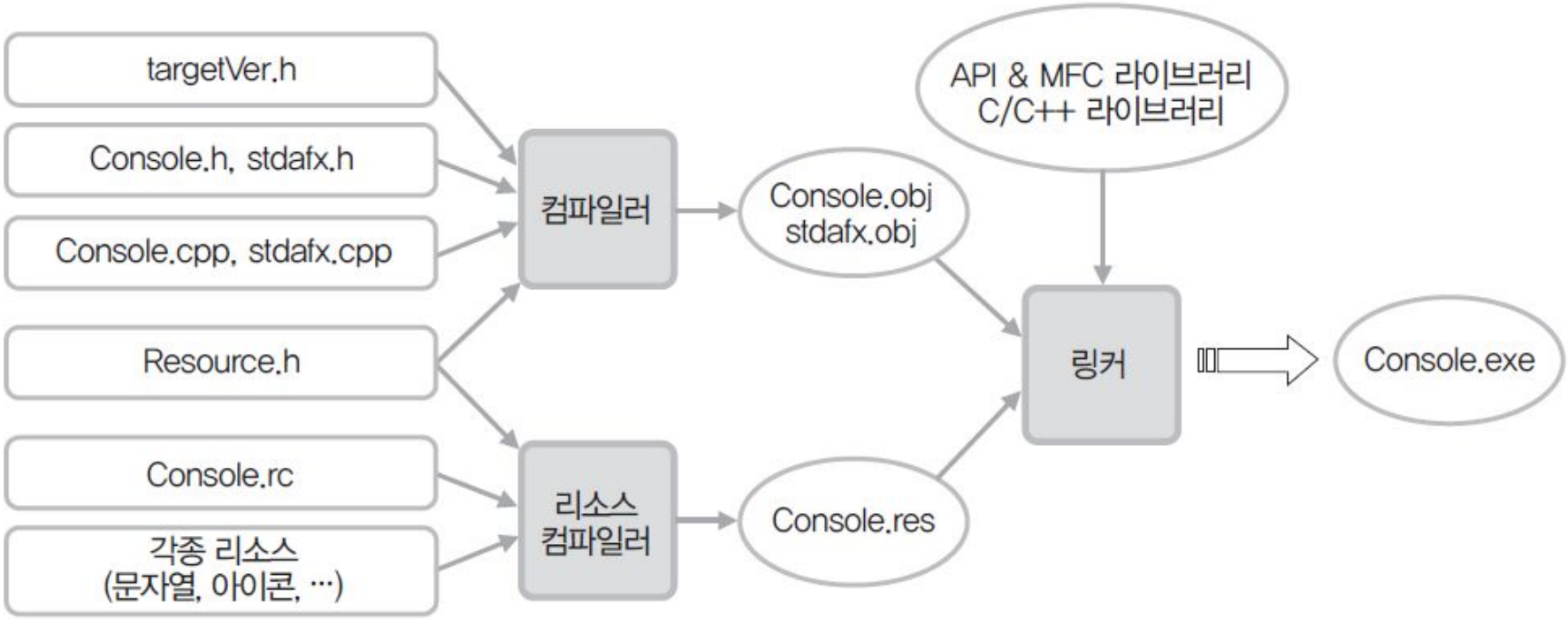
MFC 콘솔 응용 프로그램 분석 (4)

- 리소스 뷰에서 문자열 리소스 편집



MFC 콘솔 응용 프로그램 분석 (5)

- 빌드 과정



실습 – MFC 클래스 실습을 위한 준비

API 데이터 타입 (1)

- 기본형

데이터 타입	정의 또는 용도
BOOL 또는 BOOLEAN	TRUE 또는 FALSE
BYTE, WORD, DWORD, LONG	8비트, 16비트, 32비트, 32비트 (LONG을 제외하고는 모두 unsigned)
U*	unsigned * 예) UCHAR, UINT, ULONG, …
HANDLE	32비트(또는 64비트) 핸들
H*	*을 가리키는 핸들 예) HBITMAP(비트맵), HBRUSH(브러시), HCURSOR(커서), HDC(디바이스 컨텍스트), HFONT(폰트), HICON(아이콘), HINSTANCE(인스턴스), HMENU(메뉴), HPEN(펜), HWND(윈도우), …
P* = LP*	*에 대한 포인터 예1) PBOOL, PBYTE, PLONG, … 예2) LPBOOL, LPBYTE, LPLONG, …
(L)PSTR, (L)PCSTR	ANSI 문자열
(L)PTSTR, (L)PCTSTR	ANSI 또는 유니코드 문자열
COLORREF	32비트 RGB(red, green, blue 각각 8비트) 색상값

API 데이터 타입 (2)

- 구조체

데이터 타입	정의	용도
POINT	typedef struct tagPOINT { LONG x; LONG y; } POINT, *PPOINT;	점의 x, y 좌표
RECT	typedef struct tagRECT { LONG left; LONG top; LONG right; LONG bottom; } RECT, *PRECT;	직사각형의 왼쪽 상단과 오른쪽 하단 좌표
SIZE	typedef struct tagSIZE { LONG cx; LONG cy; } SIZE, *PSIZE;	폭과 높이

CString 클래스 (1)

- ANSI 또는 유니코드 문자열 지원
  - 프로젝트 속성에 따라 ANSI 또는 유니코드 문자열을 자동으로 지원

```
// 콘솔에서 한글(유니코드)을 출력하려면 필요
_tsetlocale(LC_ALL, _T( “” ));
```

- 가변 길이 문자열 지원
  - 프로그램 실행 중에 문자열 길이를 자유롭게 변경 가능
  - 문자열에 할당할 수 있는 메모리의 최대 크기: INT_MAX
- const TCHAR * 또는 LPCTSTR 대신 CString 객체를 직접 사용 가능

CString 클래스 (2)

- CString 객체 생성과 초기화

```
CString str1;
str1 = _T("안녕하세요."); // 문자열을 직접 대입한다.
CString str2(_T("오늘은")); // 문자열을 생성자 인자로 전달한다.
CString str3(str2); // CString 객체를 생성자 인자로 전달한다.
// CString 객체와 문자열을 붙인다.
CString str4 = str1 + _T(" ") + str2 + _T(" 즐거운 날입니다.");
_tprintf(_T("%s₩n"), str1);
_tprintf(_T("%s₩n"), str2);
_tprintf(_T("%s₩n"), str3);
_tprintf(_T("%s₩n"), str4);
// + = 연산자를 이용하여 기존 문자열에 새로운 문자열을 덧붙인다.
str4 += _T(" 하하하");
_tprintf(_T("%s₩n"), str4);
```

CString 클래스 (3)

- CString::Format() 함수

```
CString str;
str.Format(_T("x = %d, y = %d₩n"), 100, 200);
MessageBox(NULL, str, _T("CString::Format() 연습"), MB_OK);
```

- CString::LoadString() 함수

```
CString str;
str.LoadString(IDS_APP_TITLE); // 문자열 리소스를 로드한다.
str.Insert(0, _T("Hello from ")); // 맨 앞에 문자열을 삽입한다.
str.Append(_T("!")); // 맨 끝에 문자열을 덧붙인다.
MessageBox(NULL, str, _T("CString::LoadString() 연습"), MB_OK);
```

CPoint, CRect, CSize 클래스

- 클래스 정의

클래스 이름	정의
CPoint	class CPoint : public POINT {...};
CRect	class CRect : public RECT {...};
CSize	class CSize : public SIZE {...};

- 업캐스팅

```
void SomeFunc(POINT pt); // 함수 원형 선언
POINT pt1 = { 100, 200 }; // POINT 구조체 변수 정의
CPoint pt2(300, 400); // CPoint 클래스 객체 정의
SomeFunc(pt1); // OK! (타입 일치)
SomeFunc(pt2); // OK! (업캐스팅)
```

CPoint 클래스

- 생성자

```
CPoint::CPoint(int x, int y);
```

- 사용 예

```
CPoint pt1(10, 20); // x, y 좌표를 생성자 인자로 전달한다.
POINT pt = {30, 40};
CPoint pt2(pt); // POINT 타입 변수를 생성자 인자로 전달한다.
_tprintf(_T("%d, %d₩n"), pt1.x, pt1.y);
_tprintf(_T("%d, %d₩n"), pt2.x, pt2.y);
pt1.Offset(40, 30); // x, y 좌표에 각각 40, 30을 더한다.
pt2.Offset(20, 10); // x, y 좌표에 각각 20, 10을 더한다.
_tprintf(_T("%d, %d₩n"), pt1.x, pt1.y);
_tprintf(_T("%d, %d₩n"), pt2.x, pt2.y);
if(pt1 == pt2) // 두 객체의 내용이 같은지 확인한다.
   _tprintf(_T("두 점의 좌표가 같습니다.₩n"));
else
   _tprintf(_T("두 점의 좌표가 다릅니다.₩n"));
```

CRect 클래스 (1)

- 생성자

```
CRect::CRect(int l, int t, int r, int b);
```

- 직사각형의 폭과 높이

```
int CRect::Width( );
int CRect::Height( );
```

- 좌표의 포함 여부 판단

```
BOOL CRect::PtInRect(POINT point);
```

CRect 클래스 (2)

```
CRect rect1(0, 0, 200, 100); // 직사각형의 좌표를 생성자의 인자로 전달한다.
CRect rect2;
rect2.SetRect(0, 0, 200, 100); // 직사각형의 좌표를 실행 중에 설정한다.
if(rect1 == rect2) // 두 객체의 내용이 같은지 확인한다.
   _tprintf(_T("두 직사각형의 좌표가 같습니다.₩n"));
else
   _tprintf(_T("두 직사각형의 좌표가 다릅니다.₩n"));
RECT rect = {100, 100, 300, 200};
CRect rect3(rect); // RECT 타입 변수를 생성자 인자로 전달한다.
_tprintf(_T("%d, %d₩n"), rect3.Width(), rect3.Height());

CPoint pt(200, 150);
if(rect3.PtInRect(pt)) // 점이 직사각형 내부에 있는지 판단한다.
   _tprintf(_T("점이 직사각형 내부에 있습니다.₩n"));
else
   _tprintf(_T("점이 직사각형 외부에 있습니다.₩n"));
```

CSize 클래스

- 생성자

```
CSize::CSize(int x, int y);
```

```
CSize size1(100, 200); // 폭과 높이를 생성자 인자로 전달한다.
SIZE size = {100, 200};
CSize size2(size); // SIZE 타입 변수를 생성자 인자로 전달한다.
_tprintf(_T("%d, %d₩n"), size2.cx, size2.cy);
if(size1 == size2) // 두 객체의 내용이 같은지 확인한다.
   _tprintf(_T("크기가 같습니다.₩n"));
else
   _tprintf(_T("크기가 다릅니다.₩n"));
```

CTime 클래스

- 절대적인 시간(예: 현재 시각) 처리
- 내부적으로 시간값을 64비트로 저장

```
// CTime::GetCurrentTime() 함수로 현재 시각을 구한다.
CTime tm;
tm = CTime::GetCurrentTime();
// 여러 형식으로 화면에 출력한다.
CString str = tm.Format(_T("%A, %B %d, %Y"));
_tprintf(_T("%sWn"), str);
str.Format(_T("현재 시각은 %d시 %d분 %d초입니다."),
   tm.GetHour(), tm.GetMinute(), tm.GetSecond());
_tprintf(_T("%sWn"), str);
```

CTimeSpan 클래스

- 시간의 차이값 처리
- 내부적으로 시간값을 64비트로 저장

```
CTime startTime = CTime::GetCurrentTime();
Sleep(2000); // 2000밀리초 지연
CTime endTime = CTime::GetCurrentTime();
CTimeSpan elapsedTime = endTime - startTime;
CString str;
str.Format(_T("%d초 지남!"), elapsedTime.GetTotalSeconds());
_tprintf(_T("%s₩n"), str);
```

실습 – 유틸리티 클래스

배열 클래스 (1)

- MFC 배열 클래스
  - 배열 인덱스를 잘못 참조하는 경우 오류를 발생시킴
  - 배열 크기가 가변적임

- 템플릿 배열 클래스
  - afxtempl.h 헤더 파일

클래스 이름	데이터 타입	사용 예
CArray	프로그래머가 결정	CArray<CPoint, CPoint&> array;

배열 클래스 (2)

- 비템플릿 배열 클래스
  - afxcoll.h 헤더 파일

클래스 이름	데이터 타입	사용 예
CByteArray	BYTE	CByteArray array;
CWordArray	WORD	CWordArray array;
CDWordArray	DWORD	CDWordArray array;
CUIntArray	UINT	CUIntArray array;
CStringArray	CString	CStringArray array;
CPtrArray	void 포인터	CPtrArray array;
CObArray	CObject 포인터	CObArray array;

배열 클래스 (3)

- 배열 생성과 초기화
  - MFC 배열 클래스를 이용한 배열 생성 순서
    ① 배열 객체 생성
    ② SetSize( ) 함수를 호출하여 크기 설정

```
CUIntArray array; // 객체 생성
array.SetSize(10); // 배열 크기 설정

for (int i = 0; i<10; i++)
    array[i] = (i + 1) * 10; // 값 대입

for (int i = 0; i<10; i++)
    _tprintf(_T("%d "), array[i]); // 값 출력

_tprintf(_T("\n"));
```

배열 클래스 (4)

- 배열 생성과 초기화
  - CStringArray 클래스를 이용하면 배열에 CString 객체 저장 가능

```
_tsetlocale(LC_ALL, _T(""));
CStringArray array; // 객체 생성
array.SetSize(5); // 배열 크기 설정

for (int i = 0; i<5; i++) {
    CString string;
    string.Format(_T("%d번."), (i + 1) * 10);
    array[i] = string; // 값 대입
}

for (int i = 0; i<5; i++)
    _tprintf(_T("%s\n"), array[i]); // 값 출력
```

배열 클래스 (5)

- 배열 원소 삽입과 삭제

```
CUIntArray array;
array.SetSize(5);
for (int i = 0; i<5; i++)
array[i] = i;

/* 배열 원소 삽입 */
array.InsertAt(3, 77); // 인덱스 3 위치에 원소를 삽입한다.
for (int i = 0; i<array.GetSize(); i++) // 변경된 배열의 크기만큼 반복한다.
_tprintf(_T("%d "), array[i]);
_tprintf(_T("\n"));

/* 배열 원소 삭제 */
array.RemoveAt(4); // 인덱스 4 위치의 원소를 삭제한다.
for (int i = 0; i<array.GetSize(); i++)
_tprintf(_T("%d "), array[i]);
_tprintf(_T("\n"));
```

배열 클래스 (6)

- 템플릿 배열 클래스 (1/2)

```
#include "stdafx.h"
#include "ArrayTest2.h"
#include <afxtempl.h> // 템플릿 클래스 정의를 담고 있다.
...
// 3차원 좌표를 저장할 수 있는 클래스
// 모든 멤버가 public일 때는 class 대신 struct를 사용하면 좀더 편리하다.
struct Point3D {
    int x, y, z;
    Point3D() {}
    // 템플릿 클래스에 사용할 때는 반드시 기본 생성자가 필요하다.
    Point3D(int x0, int y0, int z0) { x = x0; y = y0; z = z0; }
};
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    ...
```

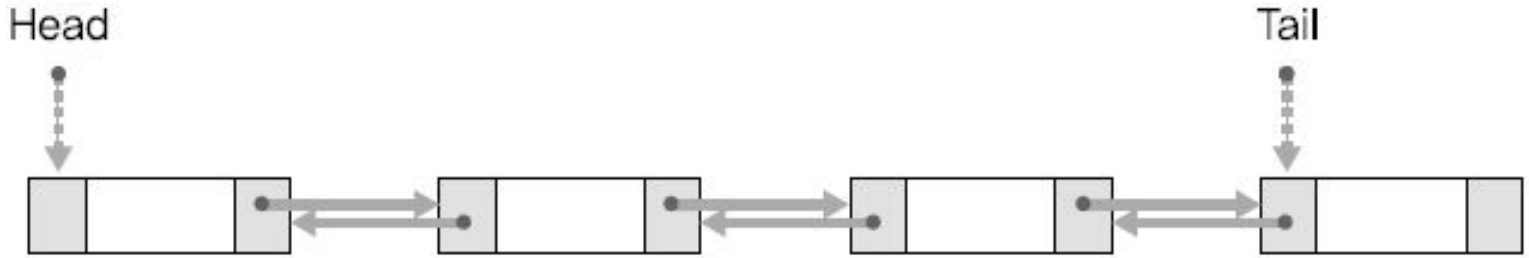
배열 클래스 (7)

- 템플릿 배열 클래스 (2/2)

```
        else
        {
          //Point3D 객체를 저장할 수 있는 배열 객체를 생성한다.
          CArray<Point3D, Point3D&> array;
          array.SetSize(5);
          for(int i=0; i<5; i++){
            Point3D pt(i, i*10, i*100);
            array[i] = pt;
        }
        for(int i=0; i<5; i++){
            Point3D pt = array[i];
            _tprintf(_T("%d, %d, %d₩n"), pt.x, pt.y, pt.z);
        }
...
```

리스트 클래스 (1)

- MFC 리스트 클래스



- 템플릿 리스트 클래스
  - afxtempl.h 헤더 파일

클래스 이름	데이터 타입	사용 예
CList	프로그래머가 결정	CList<CPoint, CPoint&> list;

리스트 클래스 (2)

- 비템플릿 리스트 클래스
  - afxcoll.h 헤더 파일

클래스 이름	데이터 타입	사용 예
CObList	CObject 포인터	CObList list;
CPtrList	void 포인터	CPtrList list;
CStringList	Cstring 객체	CStringList list;

리스트 클래스 (3)

- 리스트 생성과 초기화
  - 리스트 클래스를 이용한 리스트 생성/초기화 순서
    ① 리스트 객체 생성
    ② AddHead( ) 또는 AddTail( ) 함수를 호출하여
    원소를 리스트의 앞쪽 또는 뒤쪽에 추가

```
_tsetlocale(LC_ALL,  _T(""));
// CString 객체는 물론이고 일반 문자열도 리스트에 추가할 수 있다.
TCHAR *szFruits[] = {
  _T("사과"), _T("딸기"), _T("포도"), _T("오렌지"), _T("자두")
};

CStringList list; // 리스트 객체를 생성한다.
for (int i = 0; i<5; i++)
  list.AddTail(szFruits[i]); // 리스트 끝에 데이터를 추가한다.
```

리스트 클래스 (4)

- 리스트 순환

```
// 리스트 맨 앞에서부터 순환하면서 데이터를 출력한다.
POSITION pos = list.GetHeadPosition();
while (pos != NULL) {
    CString str = list.GetNext(pos);
    _tprintf(_T("%s "), str);
}
_tprintf(_T("\n"));

// 리스트 맨 뒤에서부터 순환하면서 데이터를 출력한다.
pos = list.GetTailPosition();
while (pos != NULL) {
    CString str = list.GetPrev(pos);
    _tprintf(_T("%s "), str);
}
_tprintf(_T("\n"));
```

리스트 클래스 (5)

- 리스트 항목 삽입과 삭제

```
pos = list.Find(_T("포도")); // 데이터의 위치를 얻는다.
list.InsertBefore(pos, _T("살구")); // 앞쪽에 데이터를 삽입한다.
list.InsertAfter(pos, _T("바나나")); // 뒤쪽에 데이터를 삽입한다.
list.RemoveAt(pos); // 데이터를 삭제한다.

// 리스트 맨 앞에서부터 순환하면서 데이터를 출력한다.
pos = list.GetHeadPosition();
while (pos != NULL) {
    CString str = list.GetNext(pos);
    _tprintf(_T("%s "), str);
}
_tprintf(_T("\n"));
```

리스트 클래스 (6)

- 템플릿 리스트 클래스 (1/2)

```
#include "stdafx.h"
#include "ListTest2.h"
#include <afxtempl.h> // 템플릿 클래스 정의를 담고 있다.
...
// 3차원 좌표를 저장할 수 있는 클래스
// 모든 멤버가 public일 때는 class 대신 struct를 사용하면 좀더 편리하다.
struct Point3D {
    int x, y, z;
    Point3D() {}
    // 템플릿 클래스에 사용할 때는 반드시 기본 생성자가 필요하다.
    Point3D(int x0, int y0, int z0) { x = x0; y = y0; z = z0; }
};
```

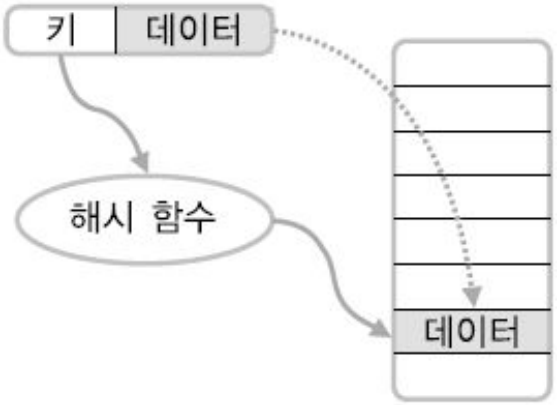
리스트 클래스 (7)

- 템플릿 리스트 클래스 (2/2)

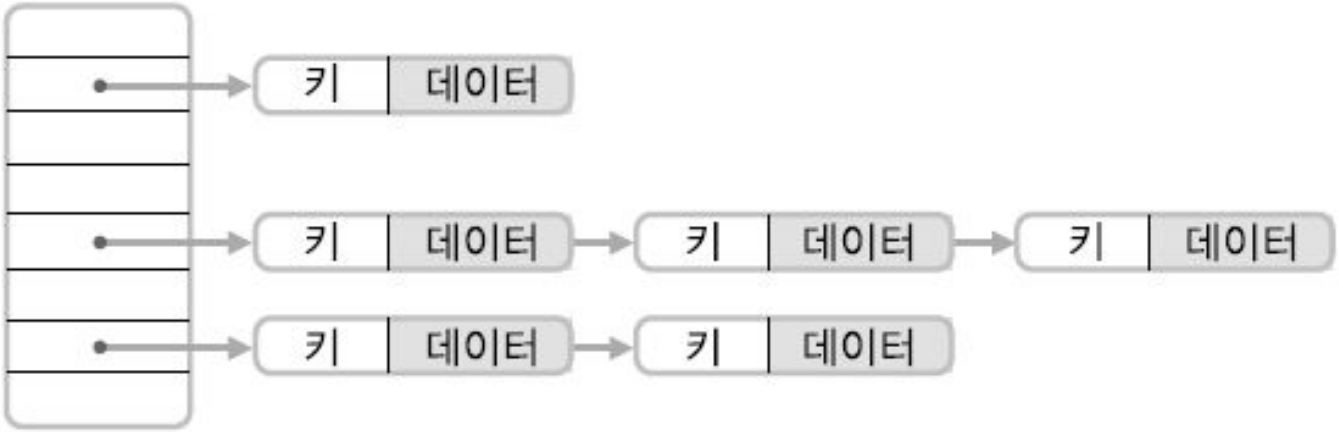
```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    ...
      else
      {
          CList<Point3D, Point3D&> list;
      // Point3D 객체를 저장할 수 있는 리스트 객체를 생성한다.
          for(int i=0; i<5; i++) / / 리스트 끝에 데이터를 추가한다.
          list.AddTail(Point3D(i, i*10, i*100));
      // 리스트 맨 앞에서부터 순환하면서 데이터를 출력한다.
      POSITION pos = list.GetHeadPosition();
      while(pos != NULL){
          Point3D pt = list.GetNext(pos);
          _tprintf(_T("%d, %d, %dWn"), pt.x, pt.y, pt.z);
      }
...
```

맵 클래스 (1)

- 맵 동작 원리



- MFC의 맵 구현



맵 클래스 (2)

- 템플릿 맵 클래스
  - afxtempl.h 헤더 파일

클래스 이름	데이터 타입	사용 예
CMap	프로그래머가 결정	CMap<CString, CString&, CPoint, CPoint&> map;

맵 클래스 (3)

- 비템플릿 맵 클래스
  - afxcoll.h 헤더 파일

클래스 이름	데이터 타입(키 → 데이터)	사용 예
CMapWordToOb	WORD → CObject 포인터	CMapWordToOb map;
CMapWordToPtr	WORD → void 포인터	CMapWordToPtr map;
CMapPtrToWord	void 포인터 → WORD	CMapPtrToWord map;
CMapPtrToPtr	void 포인터 → void 포인터	CMapPtrToPtr map;
CMapStringToOb	문자열 → CObject 포인터	CMapStringToOb map;
CMapStringToPtr	문자열 → void 포인터	CMapStringToPtr map;
CMapStringToString	문자열 → 문자열	CMapStringToString map;

맵 클래스 (4)

- 맵 생성과 초기화 및 검색
  ① 맵 객체 생성
  ② [ ] 연산자를 이용한 맵 초기화(맵 객체[키] =데이터)
  ③ 맵 객체.Lookup(키, 검색된 데이터를 담을 변수)
  형식으로 함수를 호출하여 특정 키값을 가진 데이터 검색

```
_tsetlocale(LC_ALL, _T(""));
CMapStringToString map; // 맵 객체를 생성하고 초기화한다.
map[_T("사과")] = _T("Apple");
map[_T("딸기")] = _T("Strawberry");
map[_T("포도")] = _T("Grape");
map[_T("우유")] = _T("Milk");

CString str;
if (map.Lookup(_T("딸기"), str)) // 특정 키값을 가진 데이터를 검색한다.
    _tprintf(_T("딸기 -> %s\n"), str);
```

맵 클래스 (5)

- 맵 순환

```
_tprintf( T("\n"));
// 맵을 순환하면서 모든 키와 데이터값을 출력한다.
POSITION pos = map.GetStartPosition();

while (pos != NULL){
    CString strKey, strValue;
    map.GetNextAssoc(pos, strKey, strValue);
    _tprintf(_T("%s -> %s\n"), strKey, strValue);
}
```

맵 클래스 (6)

- 맵 데이터 삽입과 삭제

```
_tprintf(_T("\n"));
map.RemoveKey(_T("우유")); // 키값 "우유"에 해당하는 데이터를 삭제한다.
map[_T("수박")] = _T("Watermelon"); // map.SetAt(_T("수박"), _T("Watermelon"));

// 맵을 순환하면서 모든 키와 데이터값을 출력한다.
pos = map.GetStartPosition();

while (pos != NULL) {
    CString strKey, strValue;
    map.GetNextAssoc(pos, strKey, strValue);
    _tprintf(_T("%s -> %s\n"), strKey, strValue);
}
```

맵 클래스 (7)

- 맵 최적화

```
CMapStringToString map; // 맵 객체를 생성한다.
map.InitHashTable(12007); // 해시 테이블 크기를 12007로 바꾼다.
```

맵 클래스 (8)

- 템플릿 맵 클래스 (1/3)

```
#include "stdafx.h"
#include "MapTest2.h"
#include <afxtempl.h> // 템플릿 클래스 정의를 담고 있다.

...

// CString 타입에 해당하는 해시 함수가 없으므로 정의한다.
template <> UINT AFXAPI HashKey(CString& str)
{
    LPCTSTR key = (LPCTSTR)str;
    return HashKey(key); // LPCTSTR 타입의 해시 함수를 재호출한다.
}
```

맵 클래스 (9)

- 템플릿 맵 클래스 (2/3)

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    ...
      else
      {
        // 콘솔에서 한글(유니코드)을 출력하려면 필요하다.
        _tsetlocale(LC_ALL, _T(""));

        // 맵(CString -> UINT) 객체를 생성하고 초기화한다.
        CMap<CString, CString&, UINT, UINT&> map;
        map[CString(_T("사과"))] = 10;
        map[CString(_T("딸기"))] = 25;
        map[CString(_T("포도"))] = 40;
        map[CString(_T("수박"))] = 15;
```

맵 클래스 (10)

- 템플릿 맵 클래스 (3/3)

```
        // 특정 키값을 가진 데이터를 검색한다.
        UINT nCount;
        if(map.Lookup(CString(_T("수박")), nCount))
          _tprintf(_T("수박 %d상자가 남아 있습니다.₩n"), nCount);
      }
  }
  else
  {
      // TODO: 오류 코드를 필요에 따라 수정합니다.
      _tprintf(_T("심각한 오류: GetModuleHandle 실패₩n"));
      nRetCode = 1;
  }

    return nRetCode;
}
```

실습 – 집합 클래스

학습정리

- 윈도우 응용 프로그램의 리소스(.res)는 .rc 파일과 리소스 파일(.ico, bmp, ...), Resource.h 파일을 리소스 컴파일로 컴파일 후 링커에 의해 실행파일에 연결됩니다.
- 문자열을 다룰 때는 프로젝트의 문자세트 속성에 따라 사용할 수 있도록 TEXT() 또는 _T() 매크로를 사용합니다.
- MFC의 CString 클래스는 프로젝트 속성에 따라 ANSI 또는 유니코드 문자열을 자동으로 지원하며, 가변 길이 문자열을 지원합니다.
- 문자열 리소스는 LoadString() 멤버함수 호출을 통해 메모리에 적재해 사용할 수 있습니다.
- CTime 클래스는 특정 시각을 나타내기 위해 사용하며, CTimeSpan은 시간의 간격을 나타내기 위해 사용하는 클래스입니다.
- MFC 집합 클래스는 배열, 리스트, 맵에 대한 템플릿 기반 클래스(afxtempl.h)와 비템플릿 기반 클래스(afxcoll.h)를 제공합니다.
- MFC 배열 클래스는 크기가 가변적인 동적 배열의 기능을 제공하며, 인덱스 범위를 벗어나 참조할 수 없습니다.
- MFC 리스트 클래스는 양방향 리스트 구조를 지원하며, 추가, 순회, 삽입, 삭제 기능을 제공합니다.
- MFC 맵 클래스는 키-데이터 쌍의 정보를 관리할 목적으로 제공되며, 주어진 키에 대해 해시함수의 값을 주소로 데이터를 저장합니다. 만일 키에 대한 해시 값 충돌 시 키-데이터 쌍에 대한 리스트를 구성합니다.