Window Programming
Visual C++ MFC Programming

Lecture 11

김예진
Dept. of Game Software

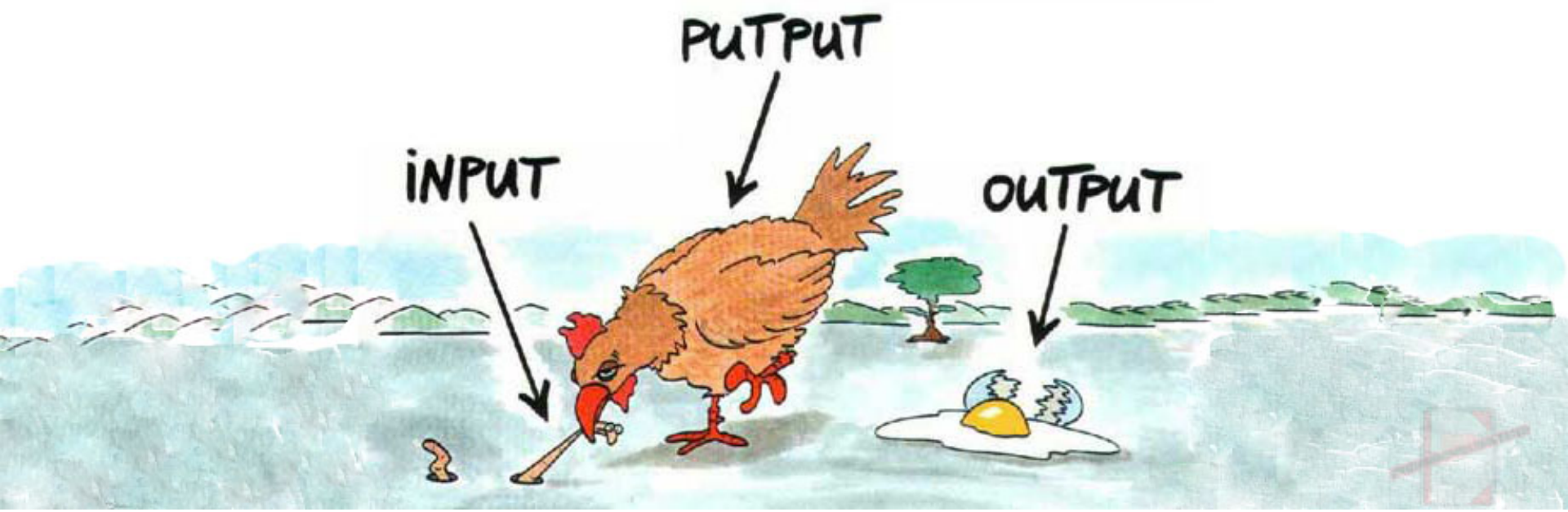
Announcement

- 03/20: HW 1 (Due: 03/26) → Avg: 8.85
- 04/05: HW 2 (Due: 04/13) → Avg: 6.90
- 04/19: Midterm → Avg: 3.70
  - 5문제, ~75 min., 강의록 1~8
- 05/10: HW 3 (Due: 05/17)

Plan: 파일 입출력 (File I/O)

- CFile Class 활용
  - 연습: CFile로 파일 저장 및 로딩
- 직렬화 (Serialization)
  - 연습: 타이핑한 문자 저장 및 로딩
  - 연습: 그린 원 파일 저장 및 로딩

파일 입출력 (File I/O)
CFile Class 활용

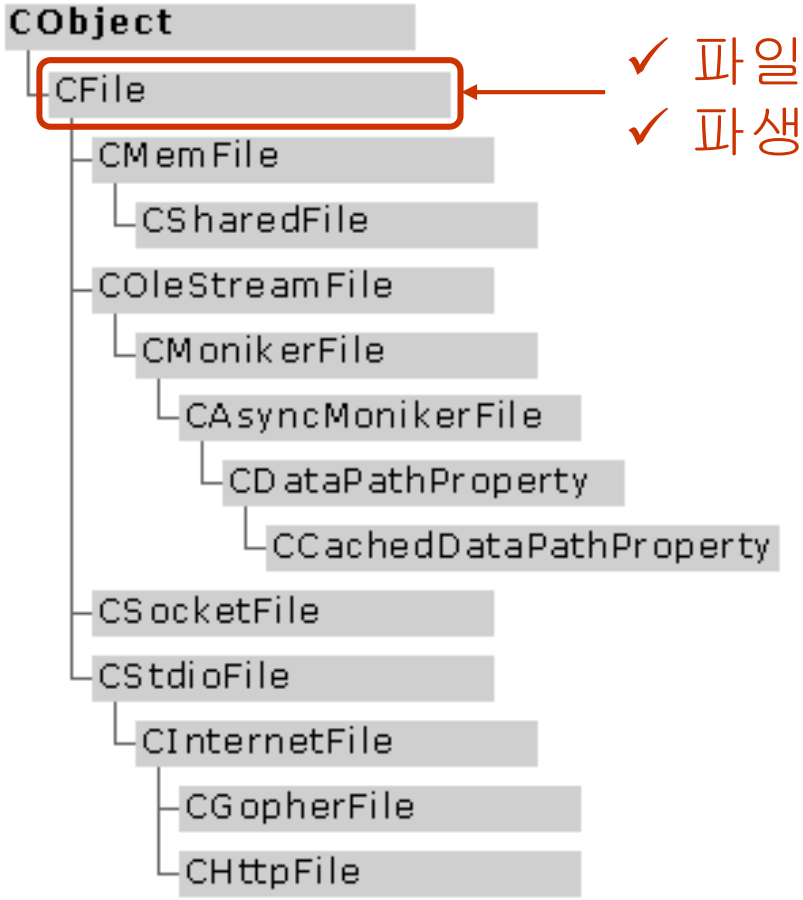


개요 (1/2)

- 파일 입출력 방법
  - 일반 파일 입출력
    - CFile (파생) 클래스
    - Read(), Write() 등의 함수 이용
    - 범용
    - 저수준의 file I/O
  - 직렬화
    - CArchive 클래스
    - << 또는 >> 연산자 이용
    - 편리한 이용
    - 사용용도의 제한

개요 (2/2)

- MFC 클래스 계층도



✓ 파일 입출력 기능 제공
✓ 파생 클래스에 공통의 인터페이스 제공

CFile 클래스

- 핵심 입출력 연산

  - 파일을 열거나 생성한다(Open).
  - 파일 포인터의 위치에서 데이터를 읽는다(Read).
  - 파일 포인터의 위치에 데이터를 쓴다(Write).
  - 파일 포인터의 위치를 변경한다(Seek).
  - 파일을 닫는다(Close).

CFile 로 파일 열기/생성

1. Open 맴버함수를 이용한 파일 열기/생성

```
CFile file;
file.Open( filename, mode, error );
```

2. 생성자를 이용한 파일 열기/생성

```
CFile file ( filename, mode );
```

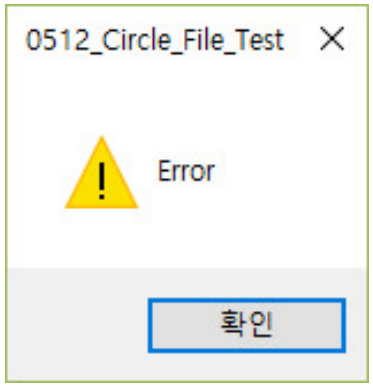
CFile 클래스 (1/6)

- 열기와 생성: Open 맴버함수 사용

```
CFile file;
file.Open( filename, mode, error );
```

```
CFile file;

if (file.Open(_T("mytest.txt"), CFile::modeRead) == false)
    AfxMessageBox(_T("Error"));
```

- 실행
  - 알수 없는 에러:



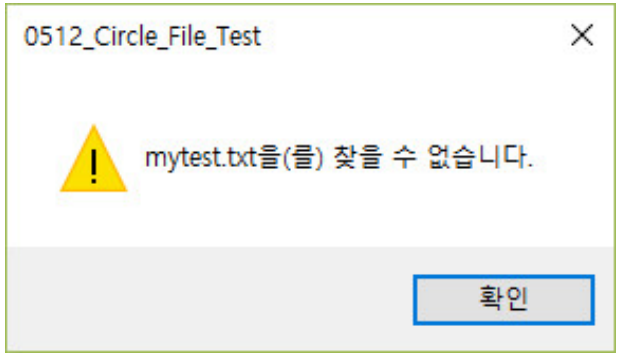
CFile 클래스 (1/6)

- 열기와 생성: Open 맴버함수 사용

```
CFile file;
file.Open( filename, mode, error );
```

```
CFile file;
CFileException e;

if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite, &e))
    e.ReportError();
```

- 실행
  - 좀 더 상세한 설명:



CFile 클래스 (1/6)

- 열기와 생성: 생성자의 사용

```
CFile file ( filename, mode );
```

```
try
{
    CFile file(_T("mytest.txt"), CFile::modeReadWrite);
}

catch (CFileException* e)
{
    e->ReportError();
    e->Delete();
}
```

CFile 클래스 (2/6)

- 파일 접근/공유 모드

플래그	의미
CFile::modeCreate	파일을 무조건 생성한다. 같은 이름의 파일이 있다면 크기를 0으로 바꾼다.
CFile::modeNoTruncate	| 연산자를 이용하여 CFile::modeCreate 플래그와 더불어 사용하면 같은 이름의 파일이 있을 경우 크기를 0으로 바꾸지 않고 이 파일을 연다.
CFile::modeRead	읽기 전용 모드으로 파일을 열거나 생성한다.
CFile::modeReadWrite	읽기 및 쓰기 모드로 파일을 열거나 생성한다.
CFile::modeWrite	쓰기 전용 모드로 파일을 열거나 생성한다.
CFile::shareDenyNone	다른 프로세스에게 파일에 대한 읽기/쓰기를 허용한다.
CFile::shareDenyRead	다른 프로세스에게 파일에 대한 읽기를 금지한다.
CFile::shareDenyWrite	다른 프로세스에게 파일에 대한 쓰기를 금지한다.
CFile::shareDenyExclusive	다른 프로세스에게 파일에 대한 읽기/쓰기를 금지한다.

CFile 클래스 (3/6)

- 파일 닫기: 소멸자에서 자동으로 닫아준다

```
void CExFileView::OnLButtonDblClk(UINT nFlags, CPoint point)
{
  CFile file;
  CFileException e;

  if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite|CFile::modeCreate, &e))
  {
      e.ReportError();
      return;
  }
    // 생략 ...
} // -> CFile::~CFile() 함수가 호출된다.
```

CFile 클래스 (4/6)

- 닫기: Close 맴버 함수 이용
  - 한 객체로 여러 파일을 다룰때

```
void CExFileView::OnLButtonDblClk(UINT nFlags, CPoint point)
{
    CFile file;
    CFileException e;
    if (!file.Open(_T("mytest.txt"), CFile::modeReadWrite | CFile::modeCreate |
                   CFile::modeNoTruncate, &e))
    {
          e.ReportError();
          return;
          }

    file.Close();
}
```

CFile 클래스 (5/6)

- 읽기와 쓰기

```
UINT CFile::Read (void* lpBuf, UINT nCount);
void CFile::Write (const void* lpBuf, UINT nCount);
```

- 파일 포인터 위치 변경

```
ULONGLONG CFile::Seek (LONGLONG lOff, UINT nFrom);
```

nFrom	의미
CFile::begin	파일의 처음 위치부터 lOff만큼 파일 포인터 이동
CFile::current	현재의 파일 포인터 위치부터 lOff만큼 파일 포인터 이동
CFile::end	파일의 끝 위치부터 lOff만큼 파일 포인터 이동

CFile 클래스 (5/6)

- 변수 저장

```
void CFile::Write (const void* lpBuf, UINT nCount) ;
```

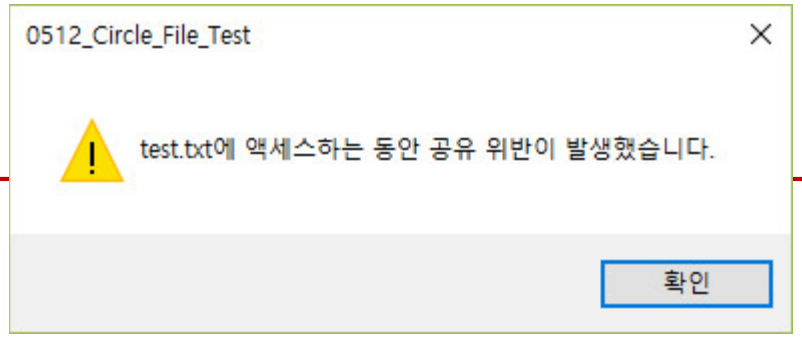
```
CFile file(_T("test.txt"), CFile::modeCreate | CFile::modeWrite);

int a = 30;
int b = 20;

file.Write(&a, sizeof(a));
file.Write(&b, sizeof(b));
```

CFile 클래스 (5/6)

- 변수 읽기

```
UINT CFile::Read (void* lpBuf, UINT nCount);
```

```
// 위에 이어서 추가…
CFile file2;
CFileException e;

if (!file2.Open(_T("test.txt"), CFile::modeRead, &e))
{
        e.ReportError();
        return;
};

int a2, b2;
file2.Read(&a2, sizeof(a2));
file2.Read(&b2, sizeof(b2));
```



CFile 클래스 (5/6)

- 변수 저장 수정

```
void CFile::Write (const void* lpBuf, UINT nCount);
```

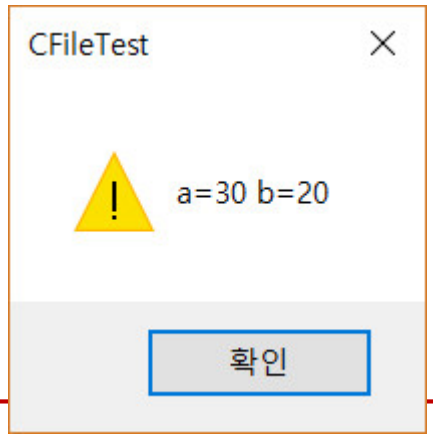
```
CFile file(_T("test.txt"), CFile::modeCreate | CFile::modeWrite);

int a = 30;
int b = 20;

file.Write(&a, sizeof(a));
file.Write(&b, sizeof(b));

file.Close();
```

CFile 클래스 (5/6)

- 읽은 변수 출력

```
UINT CFile::Read (void* lpBuf, UINT nCount);
```

```
// 위에 이어서 추가…
CFile file2;
CFileException e;

if (!file2.Open(_T("test.txt"), CFile::modeRead, &e))
{
        e.ReportError();
        return;
};

int a2, b2;
file2.Read(&a2, sizeof(a2));
file2.Read(&b2, sizeof(b2));

CString str;
str.Format(_T("a=%d b=%d"), a2, b2);
AfxMessageBox(str);
```

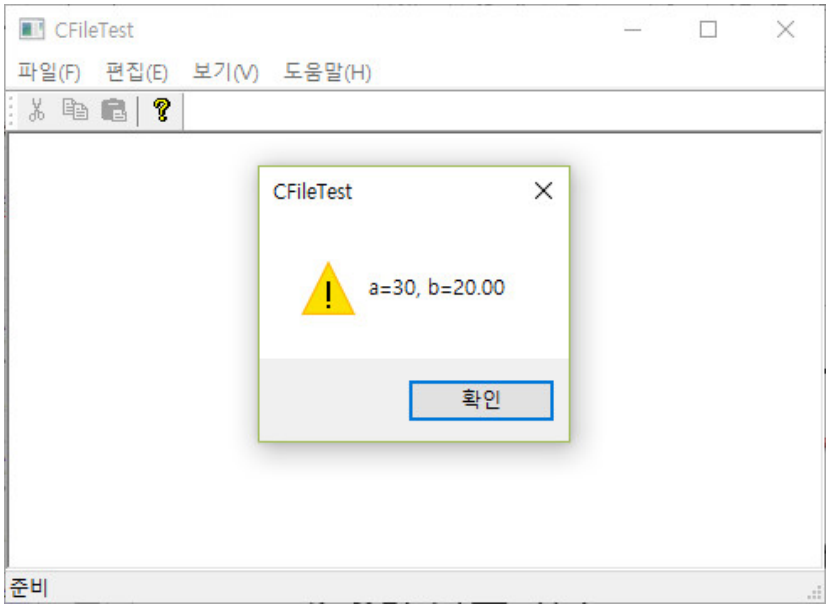


CFile 클래스 (6/6)

- 기타 함수
  - CFile::GetLength(), CFile::SetLength()
    - 파일의 현재 크기를 얻거나 변경한다.
  - CFile::GetPosition()
    - 현재의 파일 포인터 위치를 얻는다.
  - CFile::LockRange(), CFile::UnlockRange()
    - 파일의 일정 영역을 잠그거나 해제한다. 잠근 영역은 다른 프로세스가 접근할 수 없다.
  - CFile::GetFilePath(), CFile::GetFileName()
    - 파일의 전체 경로(Full Path)와 이름을 얻는다.

연습: CFile로 파일 저장 및 로딩

1. 변수 int a, float b를 만든다
2. 마우스 왼쪽 클릭을 받으면 “test.dat”란 파일을 만들고, 변수 a, b의 값을 기록
3. 마우스 오른쪽 클릭을 받으면 “test.dat” 파일을 열어 변수 값을 읽고 변수 a, b에 저장
4. AfxMessageBox함수를 이용하여 변수 값을 출력



CMemFile 클래스

- 메모리 속에 가상의 파일을 만들어 줌

```
void CExFileView::OnLButtonDblClk(UINT nFlags, CPoint point)
{
    CMemFile file;

    // 메모리 파일에 쓰기
    int a = 100;
    file.Write(&a, sizeof(a));

    // 메모리 파일에서 읽기
    file.SeekToBegin();
    int b;
    file.Read(&b, sizeof(b));
    TRACE("b = %d\n", b);
}
```

CStdioFile 클래스 (1/2)

- Text 파일을 읽고 열 때 쓰는 것
  - 문자열을 읽어오는 함수: ReadString

```
CString str;
file.ReadString(str);
```

  - 문자열을 기록하는 함수: WriteString

```
CString str =_T("Output");
file.WriteString(str);
```

CStdioFile 클래스 (2/2)

- 문자를 읽어서 대문자로 바꿔 저장하는 예

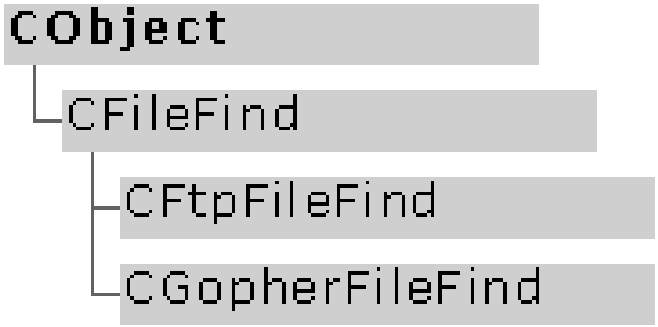
```
CStdioFile file1;
file1.Open(_T("test1.txt"), CFile::modeRead);

CStdioFile file2;
file2.Open(_T("test2.txt"),CFile::modeWrite|CFile::modeCreate);

CString str;
while(file1.ReadString(str))
{
    str.MakeUpper();
    file2.WriteString(str + "\n");
}
```

CFileFind 클래스 (1/2)

- 로컬 디스크에 대한 파일 검색 제공
  - 사용법 FindFile() 맴버 함수 사용

```
CFileFind finder;
bool bExist = finder.FindFile("MyText.txt");
```

- MFC 클래스 계층도

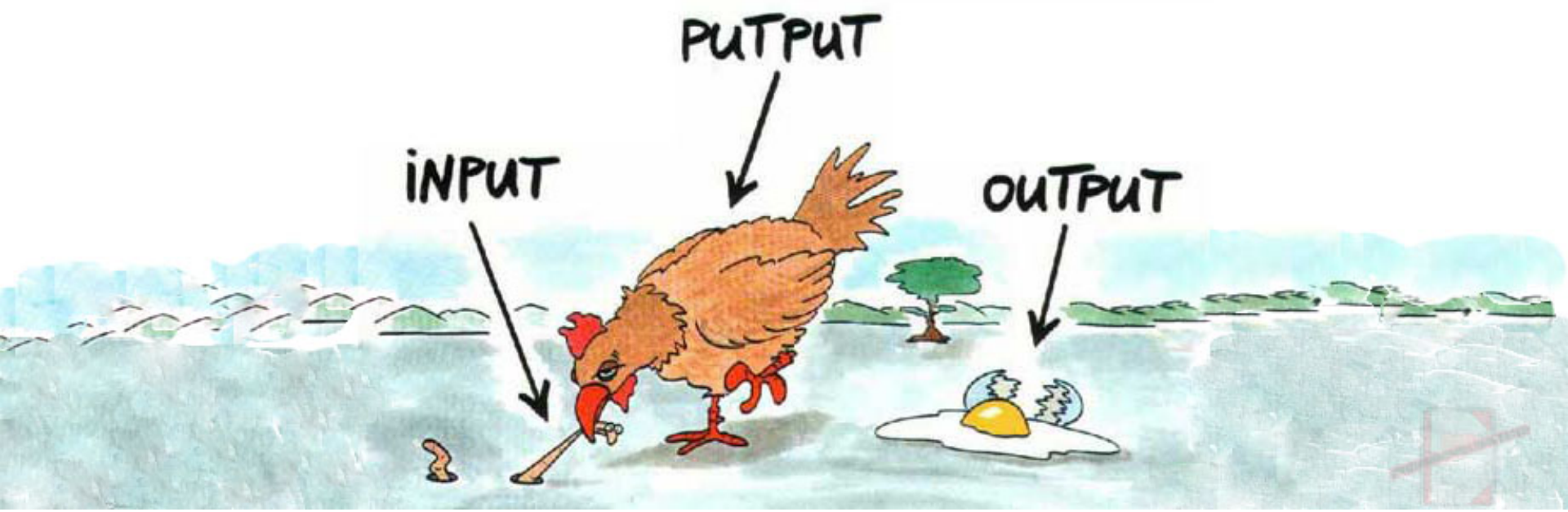


CFileFind 클래스 (2/2)

- 사용 예: 현재 디렉토리의 모든 파일을 보여줌

```
CFileFind finder;
BOOL bWorking = finder.FindFile("*.*");

while(bWorking)
{
    bWorking = finder.FindNextFile();
    if(finder.IsDirectory())
        TRACE("[%s]\n", (LPCTSTR)finder.GetFileName());
    else
        TRACE("%s\n", (LPCTSTR)finder.GetFileName());
}
```

파일 입출력 (File I/O)
직렬화 (Serialization)



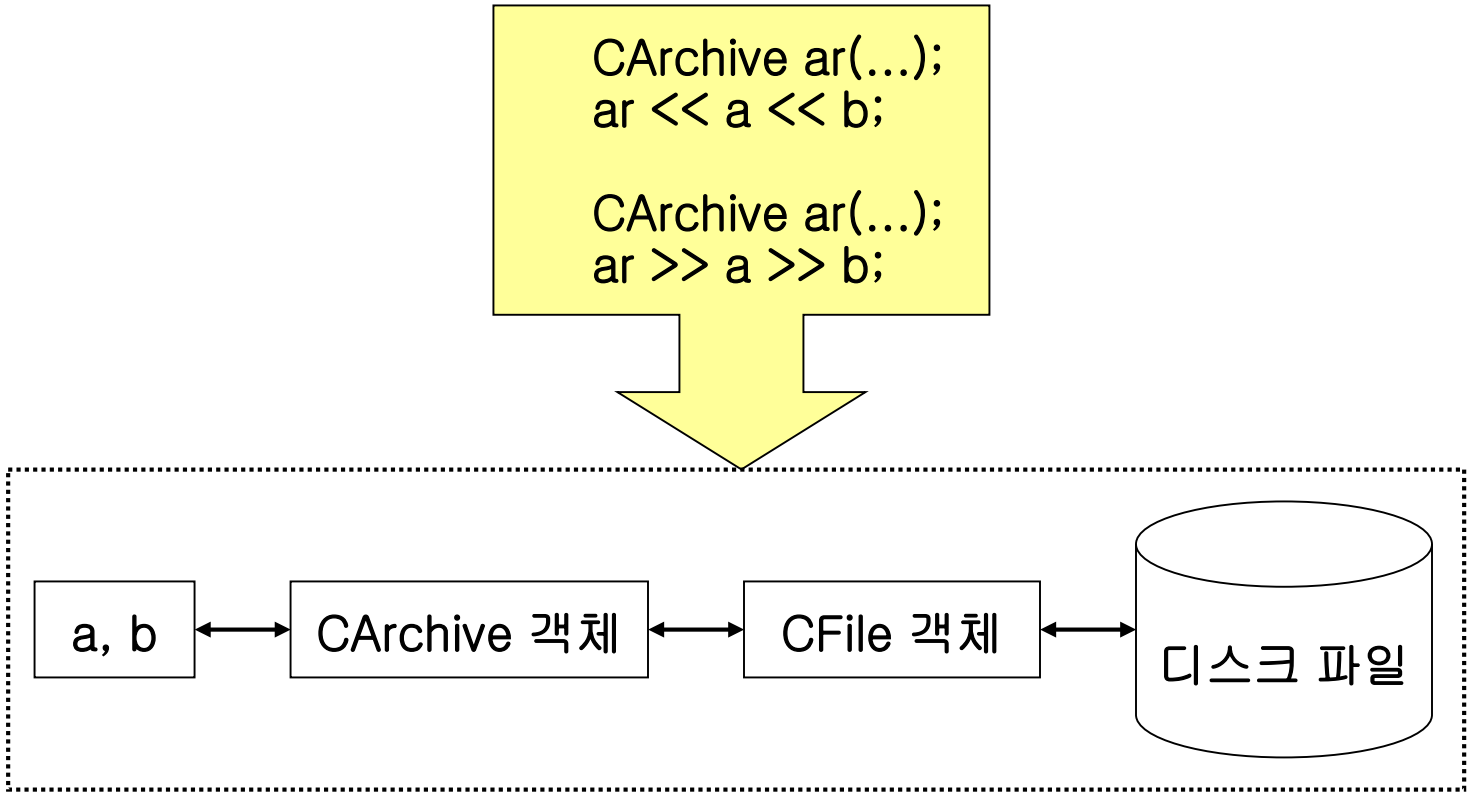
직렬화 기초 (1/8)

- 직렬화 (Serialization)
  - 영속적인(Persistitve) 저장 매체(예-디스크 파일)에 객체의 내용을 저장하거나 읽어오는 기법

  - **serialization** is … (from Wikipedia)
    - *the process of saving an object onto a storage medium (such as a file, or a memory buffer) to transmit it across a network connection link in binary form.*
    - *The series of bytes or the format can be used to re-create an object that is identical in its internal state to the original object (actually, a clone).”*

직렬화 기초 (2/8)

- 직렬화 원리



직렬화 기초 (3/8)

- 데이터 읽기 - 일반 파일 입출력

```
CFile file;
CFileException e;
if(!file.Open(_T("mytest.dat“),
    CFile::modeReadWrite|CFile::modeCreate, &e))
{
    e.ReportError();
    return
}

int a = 100;
int b = 200;
file.Write(&a, sizeof(a));
file.Write(&b, sizeof(b));
```

직렬화 기초 (4/8)

- 데이터 읽기 - 직렬화

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"), CFile::modeReadWrite | CFile::modeCreate, &e))
{
    e.ReportError();
    return;
}

int a = 100;
int b = 200;
CArchive ar (&file, CArchive::store);
ar << a << b;
```

직렬화 기초 (5/8)

- 데이터 쓰기 - 일반 파일 입출력

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"), CFile::modeRead, &e))
{
    e.ReportError();
    return;
}

int a, b;
file.Read(&a, sizeof(a));
file.Read(&b, sizeof(b));
TRACE("a = %d, b = %d\n", a, b);
```

직렬화 기초 (6/8)

- 데이터 쓰기 - 직렬화

```
CFile file;
CFileException e;

if(!file.Open(_T("mytest.dat"), CFile::modeRead, &e))
{
    e.ReportError();
    return;
}

int a, b;
CArchive ar (&file, CArchive::load);
ar >> a >> b;
TRACE("a = %d, b = %d\n", a, b);
```

직렬화 기초 (7/8)

- CArchive 클래스 생성자

```
CArchive::CArchive (CFile* pFile, UINT nMode, int nBufSize = 4096,
   void* lpBuf = NULL) ;
```

  - pFile
    - CFile 객체의 주소
  - nMode
    - CArchive::load 또는 CArchive::store
  - nBufSize
    - 내부에서 사용할 버퍼 크기
  - lpBuf
    - 사용자 정의 버퍼의 주소

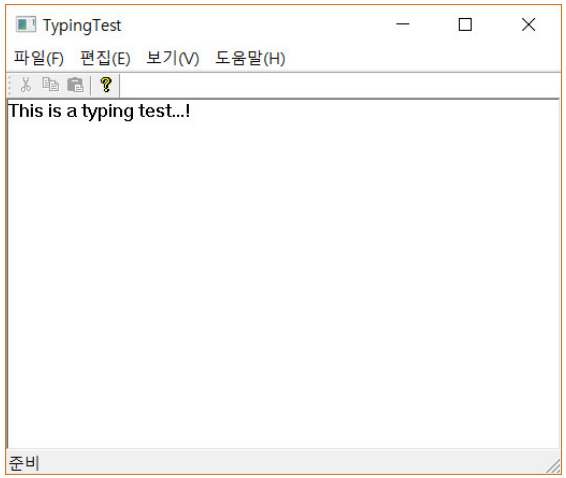
직렬화 기초 (8/8)

- 직렬화 가능한 데이터 타입

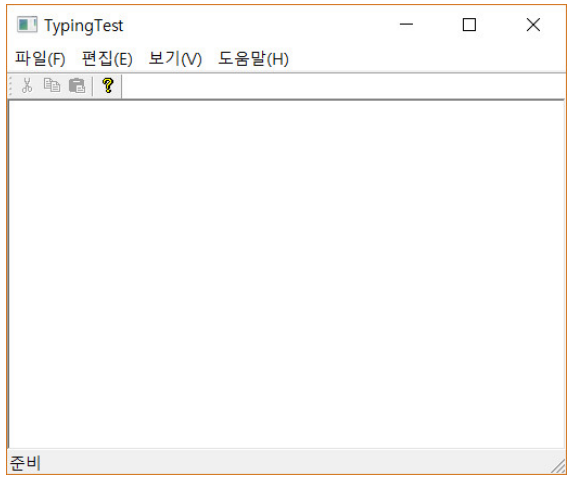
구분	데이터 타입
기본형	BYTE, WORD, LONG, DWORD, float, double, int, short, char, wchar_t, unsigned, bool, ULONGLONG, LONGLONG
비 기본형	RECT, POINT, SIZE, CRect, CPoint, CSize, CString, CTime, CTimeSpan, COleVariant, COleCurrency, COleDateTime, COleDataTimeSpan

연습: 타이핑 문자 저장 및 로딩

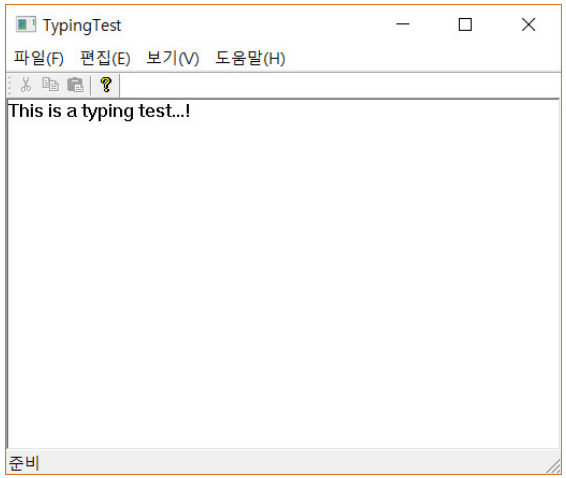
- 키를 입력 받아 글을 쓰는 프로그램 작성
  - 마우스 왼쪽 클릭: 내용을 “testString.dat” 로 저장
  - 마우스 오른쪽 클릭: “testString.dat” 내용 화면에 표시



타이핑한 문자열 저장



프로그램 재시작



저장한 문자열 로딩

직렬화 클래스 구현 (1/5)

- 사용자 정의 클래스

```
class CMyData
{
     public:
          CString m_str;
          COLORREF m_color;
     public:
          CMyData(CString &str, COLORREF &color) {
               m_str = str; m_color = color;
          }
          virtual ~CMyData();
};
```

직렬화 클래스 구현 (2/5)

- 직렬화 ⇨ X

```
void CYourChildView::SaveOrLoad(CArchive& ar)
{
    if (ar.IsStoring())
    {
        ar << m_data;
    }
    else
    {
        ar >> m_data;
    }
}
```

사용자 정의 클래스에 적합한 <<, >> 연산자가 정의 안되어 있어 동작하지 않음.
또한, standard 형식이 아님.

직렬화 클래스 구현 (3/5)

- 사용자 정의 클래스 직렬화 가능하게 변경 (Microsoft 의 standard에 따름)

```
// 클래스 선언부
class CMyData : public CObject ①
{
    DECLARE_SERIAL(CMyData) ②
public:
    CString m_str;
    COLORREF m_color;
public:
    CMyData() { } ③
    CMyData(CString &str, COLORREF &color) {
        m_str = str; m_color = color;
    }
    virtual ~CMyData();
    void Serialize(CArchive& ar); ④
};
```

직렬화 클래스 구현 (4/5)

- 사용자 정의 클래스 변경 (cont'd)

```
// 클래스 구현부
CMyData::~CMyData()
{

}

IMPLEMENT_SERIAL(CMyData, CObject, 1) ⑤

void CMyData::Serialize (CArchive& ar) ⑥
{
    CObject::Serialize(ar);
    if(ar.IsStoring())
        ar << m_str << m_color;
    else
        ar >> m_str >> m_color;
}
```

직렬화 클래스 구현 (5/5)

- 직렬화 ⇨ O

```
void CYourChildView ::SaveOrLoad(CArchive& ar)
{
    if (ar.IsStoring())
    {
        m_data.Serialize(ar);
    }
    else
    {
        m_data.Serialize(ar);
    }
}
```

Serialize 라는 함수를 호출하여 직렬화 수행: Standard에 따름.
<<, >> 연산자는 pointer에 대해서만 적용 (수업 예제 참고).

직렬화의 단점

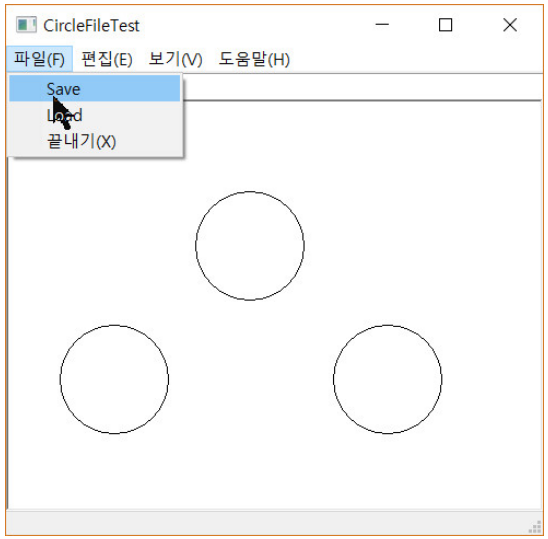
- 데이터를 불러올 때 항상 파일 전체를 읽음
- 데이터를 저장할 때 항상 파일 전체를 씀
- 데이터 이외의 정보가 기록됨

- 파일의 일부 내용만 읽어서 수정 후 다시 저장할 경우?
- 프로그래머가 원하는 부분만 저장할 경우?

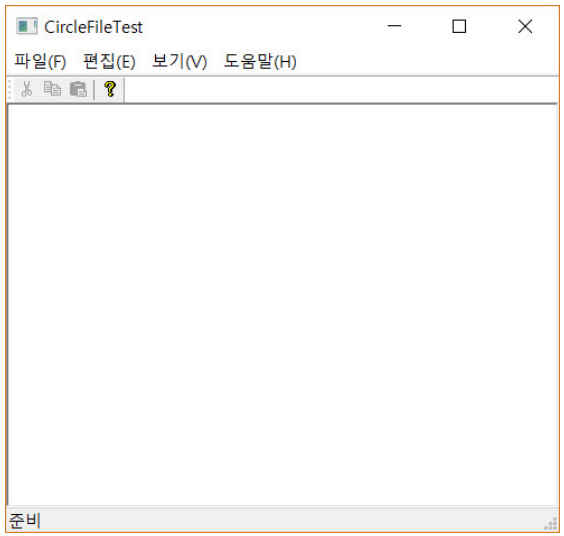
→ CFile 클래스를 직접 사용하여 입출력 부분을 작성

연습: 그린 원 파일 저장 및 로딩

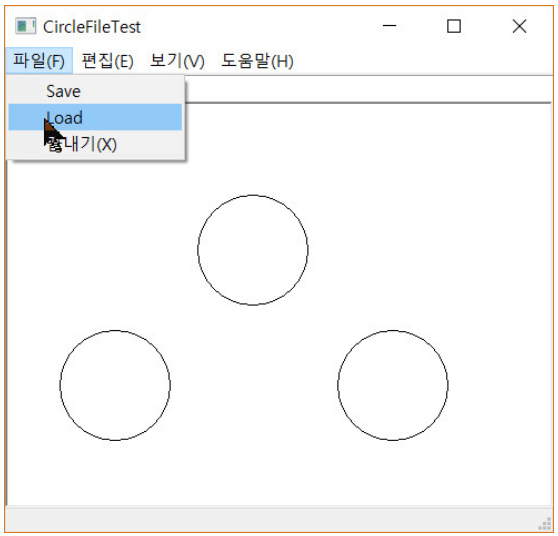
- 마우스 왼쪽 마우스를 클릭하여 원을 다수 그린다.
  - Save라는 메뉴를 추가하여 “circle.dat”라는 이름으로 현재의 원들의 정보를 저장한다.
  - Load라는 메뉴를 추가하여, “circle.dat”라는 파일을 읽어들여 저장된 원들의 위치를 복원한다.



그린 원 저장



프로그램 재시작



저장한 원 로딩

Q & A