Hanbit eBook
Realtime 70

MFC 프로그래밍

# 주식 분석 프로그램 만들기

김세훈 지음

한빛미디어 Hanbit Media, Inc.

MFC 프로그래밍

# 주식 분석 프로그램 만들기

## MFC 프로그래밍 주식 분석 프로그램 만들기

**초판발행** 2014년 06월 24일

**지은이** 김세훈 / **펴낸이** 김태헌
**펴낸곳** 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
**전화** 02-325-5544 / **팩스** 02-336-7124
**등록** 1999년 6월 24일 제10-1779호
**ISBN** 978-89-6848-664-7 15000 / **정가** 12,000원

**책임편집** 배용석 / **기획** 정지연 / **편집** 정지연·이세진
**디자인** 표지 여동일, 내지 스튜디오 [밈], 조판 최송실
**영업** 김형진, 김진불, 조유미 / **마케팅** 박상용, 서은옥, 김옥현

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
**한빛미디어 홈페이지** www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea


**지금 하지 않으면 할 수 없는 일이 있습니다.**
**책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.**
**한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.**

# 저자 소개

**지은이_김세훈**

"나이에 비해서 경력이 없네요." 2006년 36살의 나이에 한 회사의 CTO에게 들은 말이다. 실력이 없으면 내 인생을 내 마음대로 결정할 수 없다 생각했는데, 우리나라는 나이가 아주 중요한 요소임을 간과했었다. 사회는 변한다. 고령화 사회를 걱정하며, 먼 훗날 젊은이들이 부양해야 할 노인들이 많다고 언론에선 말한다. 그러나 고령화 사회는 노인이 돼서도 일해야 하는 사회다. 저자는 노인이 돼서도 즐겁게 일하기 위하여 실력이 나이를 극복할 수 있다고 믿으며 현재도 자신을 만들어 가는 중이다. 또한, 인생의 나머지 반을 어떻게 살 것인지 항상 고민한다.

# 저자 서문

우리나라에서는 외국인들이 프로그램 매매로 주식을 한다고 알고 있다. 프로그램 매매라는 단어에서 전문가적인 느낌을 받지만, 간단히 말하면 주식 매매하는 사람들이 컴퓨터 프로그램도 만들 수 있다는 것이다. 물론, 주식 분석가와 프로그래머 등 여러 사람이 모여 그룹을 형성하여 함께 만들어나간 것이라고 보면 된다. 일반적으로 주식 매매를 하는 사람들은 컴퓨터 프로그램에 대하여 잘 알지 못하고, 프로그래머는 주식을 깊이 있게 알지 못한다. 하지만 앞으로 주식 매매를 하거나 주식을 깊이 있게 이해하기 위해서는 스스로 프로그램을 만들 필요가 있다.

요즘은 증권회사에서 주식 데이터를 가져올 수 있다. 예전에는 증권회사에서 제공하는 데이터와 주식 정보만을 봐야 했다면 이제는 일반 사람들이 가공되지 않은 주식 데이터를 가져와서 스스로 원하는 차트Chart를 만들 수 있다. 즉, 프로그래밍할 줄 안다면 자신이 주로 보는 차트를 직접 만들어 실시간으로 변화를 확인할 수 있다. 선택한 종목의 데이터를 1초마다 계속 가져와서 새로 그려주면(컴퓨터의 속도는 우리가 생각하는 것 이상으로 빠르므로) 사용자는 큰 불편 없이 1초마다 변하는 데이터를 보면서 매매할 수 있다. 이 책에서는 실시간으로 데이터를 가져와 차트를 업데이트하는 방법은 구현하지 않지만, 이 책을 이해한 독자라면 큰 문제 없이 실시간 주식 프로그램을 만들 수 있다.

주식 분석할 때 주로 보는 보조 지표는 MACD와 볼린저 밴드Bollinger Band다. 일반 사람들이 주로 보는 이평선보다 참고자료로 사용하기에 아주 좋다. 그러나 주식을 하는 사람들은 자신이 생각한 것을 믿는 경향이 있다. 예를 들면, MACD라는 보조 지표를 사용하면 30%만이 종목에 맞게 적용되는데도 MACD를 맹신하는 사람들은 자신에게 잘 맞는 보조 지표라고 생각하며 믿곤 한다. 그러나 보조 지표는 말 그

대로 참고용이며, MACD와 볼린저 밴드를 함께 보는 것이 좋다. 주식에서도 철저한 준비가 필요하다. 자신이 사용하는 알고리즘이 있다면 증권회사에서 받은 데이터를 가지고 시뮬레이션을 하는 것이 좋다.

현재 사회는 잘해야 하는 시대이다. 노동으로 먹고살던 시대에는 근면 성실하기만 하면 잘 살 수 있었지만, 지금은 노동의 시간이 아닌 결과물의 시대가 되었다. 주식을 한다는 것은 머리로 생각하며 올바른 분석을 해야 하는 일이다. 또한, 주식에서 프로그래밍한다는 것은 정보의 분석뿐 아니라 남들보다 빨리 정보를 얻을 힘이 생긴 것이라고 볼 수 있다. 주식 프로그램에서 궁극의 목적은 컴퓨터가 알아서 매매하는 것인데, 머지않은 미래에는 프로그램을 누가 더 잘 만들었는지가 중요할 것이며, 주식 시장은 컴퓨터 간의 경쟁이 될 것이다.

이 책을 나에게 주식을 가르쳐 주신 박준근 형님에게 바친다. 또한, 이 책이 출간되기를 가장 많이 기다린 친구 성연준의 둘째 아들 성주호(2008년생)에게 선물로 준다. 주호는 가끔 전화로 나에게 말했다. “삼촌, 내 책은 언제 줄 거야?”

**집필을 마치며**

**김세훈**

# 일러두기

이 책은 'How-to Series'의 두 번째 책으로 MFC를 이용하여 주식 프로그램을 만드는 방법에 대해 알아본다. 주식 프로그램은 증권회사에서 데이터를 가져와 분석한 후, 자동매매를 함으로써 완성된다. 즉, 시스템 트레이딩 프로그램을 잘 구축하는 것이 중요하다. 하지만 이 책은 MFC 프로그래밍에 초점을 맞추어 주식이라는 주제를 다루고자 한다. 이 책을 이해한 독자라면 증권사에서 데이터를 가져오는 API를 이용하여 자신만의 자동매매 시스템을 쉽게 구현할 수 있다. 여기서 다루는 프로그램은 윈도우 MFC에 기반을 두고 있으며, 이 책을 통해 MFC 프로그래밍을 이해할 수 있는 계기가 될 것이다.

이 책에서 다룰 내용은 다음과 같다.

- 주식 그래프 그리기
- 이평선 구현과 알고리즘 검증
- MACD Moving Average Convergence/Divergence, 이동평균 수렴/확산 지수 구현과 자동매매 검증
- 볼린저 밴드 Bollinger Band 구현과 자동매매 검증

'주식 그래프 그리기'에서는 MFC로 그래프를 구현하는 방법과 MFC의 몇 가지 기능을 다룬다. 첫 번째 항목을 완벽히 이해하면 나머지 항목들은 주식 알고리즘을 중점적으로 다루므로 프로그램을 쉽게 이해할 수 있다. 'How-to Series'의 첫 번째 책인 『소수와 RSA 알고리즘으로 배우는 Big Number 연산』(한빛미디어, 2013)[01]에서 구조체 사용에 대해 설명한 대로, 주식 프로그램도 구조체를 이용하여 필요한 데이터를 자유자재로 가져다 쓸 수 있고, 개인적으로 구현한 알고리즘을 검증하는

---

01 http://www.hanbit.co.kr/ebook/look.html?isbn=9788968486074

데 큰 무리가 없다.

주식은 세계 증시, 유가, 회사 공시 등 많은 정보를 가지고 주식을 판단해야 하지만, 이 책에서는 오직 다섯 개의 정보(시가/고가/저가/종가/거래량)만 가지고 프로그램을 구현하고 공개된 알고리즘만 다룬다.

이 책을 읽는 독자가 차트를 분석하는 데 초보자라면 차트 분석을 경험할 수 있을 것이고, 분석이 가능한 중급 이상의 경력자로 자신만의 분석 알고리즘이 있다면 직접 알고리즘을 구현하여 검증할 수 있을 것이다. MACD와 볼린저 밴드에서 다룰 자동매매는 일봉을 가지고 검증하고, 상수값을 다양한 값들로 변경하면서 검증할 것이다. 프로그래밍을 위해서는 구현할 알고리즘을 정확히 이해해야 하므로 여기서 다루는 보조 지표의 특성을 좀 더 깊이 있게 알 수 있을 것이다.

부록에서는 이트레이드증권의 API인 Xing API를 간단히 설명하고, 이 책에서 사용하는 데이터를 Xing API를 통하여 받아오는 프로그램을 제공한다. 주식 분석은 가장 최근 데이터를 가지고 분석하는 것이 좋으므로 부록에 수록된 프로그램으로 원하는 데이터를 가져오는 것도 좋다. 또한, 일봉 데이터를 가지고 진행하지만, 주봉 데이터를 받는다면 주간 데이터를 가지고 프로그램의 검증이 이루어질 수 있을 것이다. 필자의 경험으로는 현재 이트레이드증권의 Xing API가 MFC 프로그램을 만들기에는 가장 적당하며 풍부한 자료를 제공한다. 주식을 분석한다는 취지의 프로그램 책이지만, 잘 만들어진 주식 프로그램은 증권사 API를 사용하여 실시간으로 데이터를 가져와 분석하여 자동매매를 하는 것이 궁극의 목표일 수 있다. 필자는 비록 실제로 주식을 하지는 않지만, 오래전부터 데이터를 가지고 분석 작업을

진행해 왔다. 따라서 이 책의 내용은 분석 작업의 시작일 수 있으며 독자들이 확실하게 자신만의 알고리즘을 만든다면 모의투자 방식으로 검증 후 실제 매매에 적용할 수도 있을 것이다.

이 책은 주식 프로그램을 간단히 다루었지만, 언젠가는 뛰어난 알고리즘을 구현하는 한국 사람이 뉴욕의 주식 시장에서 시스템 트레이딩으로 그들과 경쟁하기를 바라는 마음이다. 현재는 외국의 프로그램 매매를 우리가 따라가고 있지만, 나중에는 우리도 그들의 시장에서 함께 할 수 있을 것이다. 그런 의미에서 'How-to Series'의 두 번째 불을 조용히 지펴주기 바란다.

# 대상 독자 및 참고사항

초급 **초중급** 중급 중고급 고급

이 책은 C++를 알고 있는 초급자 이상을 대상으로 윈도우 프로그램을 만드는 MFC 프로그래밍을 설명합니다. 주제가 있는 프로그램을 배우자는 목적으로 주식이라는 주제로 프로그램을 만들면서 설명합니다.

주식을 하는 사람들은 자신만의 프로그램을 만들어 운용하기를 원하지만, 방법을 모르기에 어떻게 시작을 해야 할지 모를 수 있습니다. 이 책은 MFC 프로그램의 버튼 만들기부터 시작하여 프로그램으로 구현하는 방법을 단계적으로 설명합니다.

주식 분석은 증권회사에서 제공하는 HTS 프로그램을 이용하는데, HTS 프로그램은 MFC로 만들었습니다. 따라서 이 책을 이해한 독자는 간단한 개인용 HTS를 만들어서 응용할 수 있습니다. 또한, 주식의 보조 지표인 이평선이 어떻게 만들어졌는지를 알 수 있으며, 주식을 좀 더 깊게 분석할 때 주로 사용하는 보조 지표인 MACD와 볼린저 밴드를 구현하는 방법도 이해할 수 있어서 두 보조 지표를 활용하는 데 도움이 될 것입니다.

보조 지표가 어떻게 만들어지는지 아는 것과 모르고 그냥 보는 것에는 큰 차이가 있습니다. 보조 지표를 만들 수 있다는 것은 그만큼 보조 지표를 여러 측면으로 분석하는 힘이 있다는 것입니다. 이 책에서 제공하는 방법의 하나는 데이터를 이용해서 가상 시뮬레이션을 해보는 알고리즘 검증으로, 자신만의 알고리즘이 있다면 주식 데이터로 검증하는 법도 이해할 수 있습니다. MFC 프로그래밍에 대한 책이지

만, 주식 초보자는 주식 분석을 이해하는 데 도움이 되고, 중급자 이상은 자신만의 알고리즘을 검증하는 데 도움이 될 것입니다.

이 책에 수록된 코드들은 윈도우에서 생성하고 실행시켜야 하며 Visual studio 2008에서 구현되었습니다. 'How-to Series'는 방법론적인 부분에 초점을 맞추어 만들어진 책입니다. 일방적인 지식의 전달이라기보다는 "이러한 방법도 있구나" 하고 이해하는 것이 중요합니다.

**예제 테스트 환경**

| 사용 프로그램 | 설명 |
|---|---|
| Visual Studio 2008 이상 | 예제는 Windows 환경에서 테스트하였다. |
| Windows OS | Windows 환경에서만 실행할 수 있다. |

책에서 사용한 예제 파일은 다음 웹 사이트에서 내려받을 수 있다.

- https://www.hanbit.co.kr/exam/2664

# 한빛 eBook 리얼타임

한빛 eBook 리얼타임은 IT 개발자를 위한 eBook입니다.

요즘 IT 업계에는 하루가 멀다 하고 수많은 기술이 나타나고 사라져 갑니다. 인터넷을 아무리 뒤져도 조금이나마 정리된 정보를 찾는 것도 쉽지 않습니다. 또한 잘 정리되어 책으로 나오기까지는 오랜 시간이 걸립니다. 어떻게 하면 조금이라도 더 유용한 정보를 빠르게 얻을 수 있을까요? 어떻게 하면 남보다 조금 더 빨리 경험하고 습득한 지식을 공유하고 발전시켜 나갈 수 있을까요? 세상에는 수많은 종이책이 있습니다. 그리고 그 종이책을 그대로 옮긴 전자책도 많습니다. 전자책에는 전자책에 적합한 콘텐츠와 전자책의 특성을 살린 형식이 있다고 생각합니다.

한빛이 지금 생각하고 추구하는, 개발자를 위한 리얼타임 전자책은 이렇습니다.

1. eBook Only - 빠르게 변화하는 IT 기술에 대해 핵심적인 정보를 신속하게 제공합니다.

   500페이지 가까운 분량의 잘 정리된 도서(종이책)가 아니라, 핵심적인 내용을 빠르게 전달하기 위해 조금은 거칠지만 100페이지 내외의 전자책 전용으로 개발한 서비스입니다. 독자에게는 새로운 정보를 빨리 얻을 수 있는 기회가 되고, 자신이 먼저 경험한 지식과 정보를 책으로 펴내고 싶지만 너무 바빠서 엄두를 못 내는 선배, 전문가, 고수 분에게는 보다 쉽게 집필할 수 있는 기회가 될 수 있으리라 생각합니다. 또한 새로운 정보와 지식을 빠르게 전달하기 위해 O'Reilly의 전자책 번역 서비스도 하고 있습니다.

2. 무료로 업데이트되는 전자책 전용 서비스입니다.

   종이책으로는 기술의 변화 속도를 따라잡기가 쉽지 않습니다. 책이 일정 분량 이상으로 집필되고 정리되어 나오는 동안 기술은 이미 변해 있습니다. 전자책으로 출간된 이후에도 버전 업을 통해 중요한 기술적 변화가 있거나 저자(역자)와 독자가 소통하면서 보완하여 발전된 노하우가 정리되면 구매하신 분께 무료로 업데이트해 드립니다.

### 3. 독자의 편의를 위해 DRM-Free로 제공합니다.

구매한 전자책을 다양한 IT 기기에서 자유롭게 활용할 수 있도록 DRM-Free PDF 포맷으로 제공합니다. 이는 독자 여러분과 한빛이 생각하고 추구하는 전자책을 만들어 나가기 위해 독자 여러분이 언제 어디서 어떤 기기를 사용하더라도 편리하게 전자책을 볼 수 있도록 하기 위함입니다.

### 4. 전자책 환경을 고려한 최적의 형태와 디자인에 담고자 노력했습니다.

종이책을 그대로 옮겨 놓아 가독성이 떨어지고 읽기 힘든 전자책이 아니라, 전자책의 환경에 가능한 한 최적화하여 쾌적한 경험을 드리고자 합니다. 링크 등의 기능을 적극적으로 이용할 수 있음은 물론이고 글자 크기나 행간, 여백 등을 전자책에 가장 최적화된 형태로 새롭게 디자인하였습니다.

앞으로도 독자 여러분의 충고에 귀 기울이며 지속해서 발전시켜 나가도록 하겠습니다.

# 차례

# 1 | MFC 시작하기

현재 가장 많이 사용하는 컴퓨터 운영체제는 윈도우로, 윈도우에서 실행되는 프로그램은 사용하기 쉽도록 그래픽 부분이 강조된다. 마우스가 없던 시절에는 텍스트만 사용해서 컴퓨터를 작동시켰지만, 윈도우 프로그램은 주로 마우스로 업무를 수행할 수 있다. 이렇게 사용자 편의 위주의 프로그램을 만드는 것은 프로그래머에게 더 많은 코딩을 하게 만든다.

MFCMicrosoft Foundation Class는 윈도우 프로그램을 만드는 확장된 C++ 라이브러리Library다. MFC는 개발자가 좀 더 쉽게 윈도우 프로그램을 만들 수 있는 기능을 제공한다. MFC에서 사용하는 언어는 비주얼 C++라고 부른다. 단어에서도 알 수 있듯이 시각적인Visual 윈도우 응용 프로그램Application을 만들기 위한 프로그래밍 언어다. 지금은 MFC를 사용하는 프로그래머가 많지 않으나, 윈도우 프로그램을 다룰 줄 알면 프로그래밍의 범위를 확장할 수 있다.

증권회사에서 제공하는 API는 MFC에 기반을 두고 있어서 증권이나 주식 관련 데이터를 다루려면 좀 더 MFC에 친숙해져야 한다. 물론, Excel의 매크로 기능으로 주식 데이터를 다룰 수 있지만, 주식 데이터를 좀 더 자유자재로 다루기 위해서는 프로그래밍 언어를 사용하는 것이 훨씬 효과적이다.

여기서는 MFC의 기능을 모두 설명하지 않는다. 기능을 전부 다루려면 두꺼운 책 한 권 분량이 되므로 프로그램을 만들면서 필요한 부분만을 설명한다. 그리고 프로그램을 만들기 위해 시중에 나와 있는 MFC 책을 독파할 필요도 없다. 단지, 구현해야 할 기능들만 인터넷이나 안내서를 통하여 습득하는 것이 좋다. 프로그래밍에서는 모든 기능을 알고 만드는 것보다 필요한 것이 무엇인지를 아는 것이 중요하다.

## 1.1 Visual Studio

비주얼 스튜디오는 윈도우 프로그램을 만드는 데 필요한 응용 프로그램으로, 컴파일 기능이 있는 편집기Editor라고 할 수 있다. 이 책에서는 비주얼 스튜디오 2008을 사용한다. 대부분의 증권회사 API가 비주얼 스튜디오 2008을 기반으로 만들어졌으므로 최신 버전의 비주얼 스튜디오가 굳이 필요하지 않다. 또한, 이 책에서 구현하는 프로그램들은 상위 버전의 비주얼 스튜디오에서도 실행할 수 있다.

여기서는 비주얼 스튜디오의 자세한 사용법을 명시하지 않으며, 프로그램을 만들면서 필요한 부분만을 설명한다. 자세한 내용은 인터넷 등을 찾아서 습득하기 바란다.

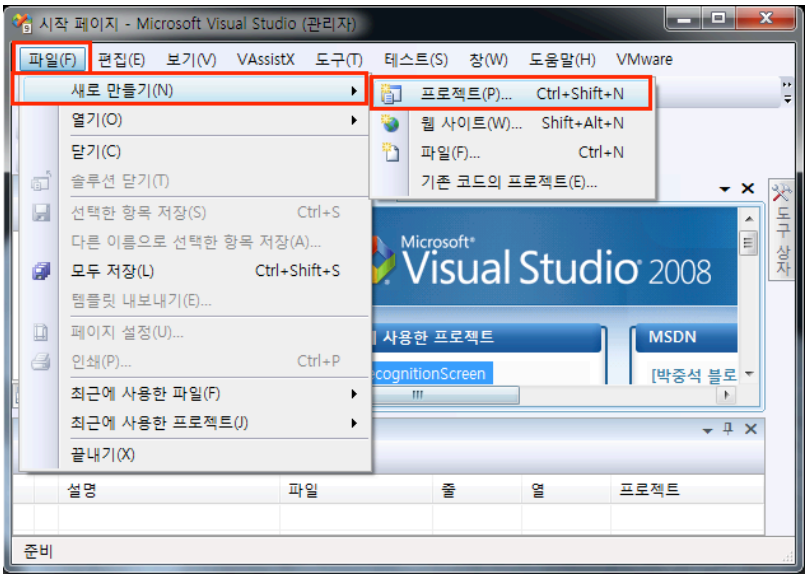
## 1.2 MFC 프로그래밍을 위한 준비

### 1.2.1 프로젝트 생성하기

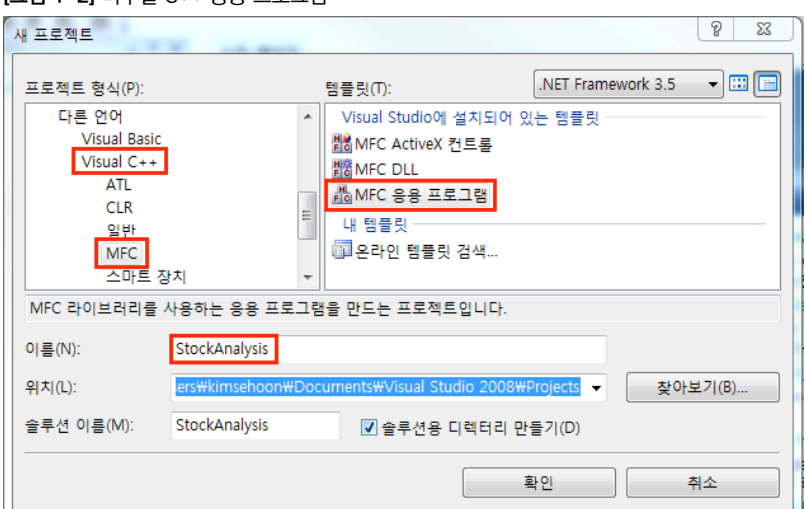
MFC는 비주얼 스튜디오를 실행하고 새 프로젝트를 만든다. [그림 1-1]과 같이 비주얼 스튜디오의 상위 메뉴에서 '파일(F)'을 선택한 후, '새로 만들기(N)'와 '프로젝트(P)'를 선택하여 프로젝트를 생성한다.

새 프로젝트 창이 열리면 [그림 1-2]와 같이 Visual C++을 선택하여 MFC 응용 프로그램을 만들고, '이름(N)'에는 새 프로젝트의 이름을 써준다. 이 책에서 구현하는 모든 프로그램은 StockAnalysis주식 분석이라는 이름을 사용한다.

**[그림 1-1]** 새 프로젝트 만들기

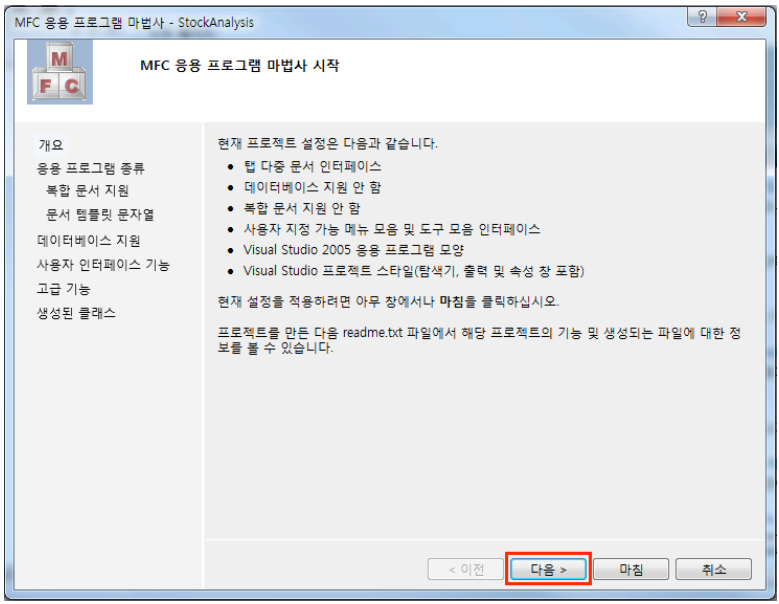



**[그림 1-2]** 비주얼 C++ 응용 프로그램



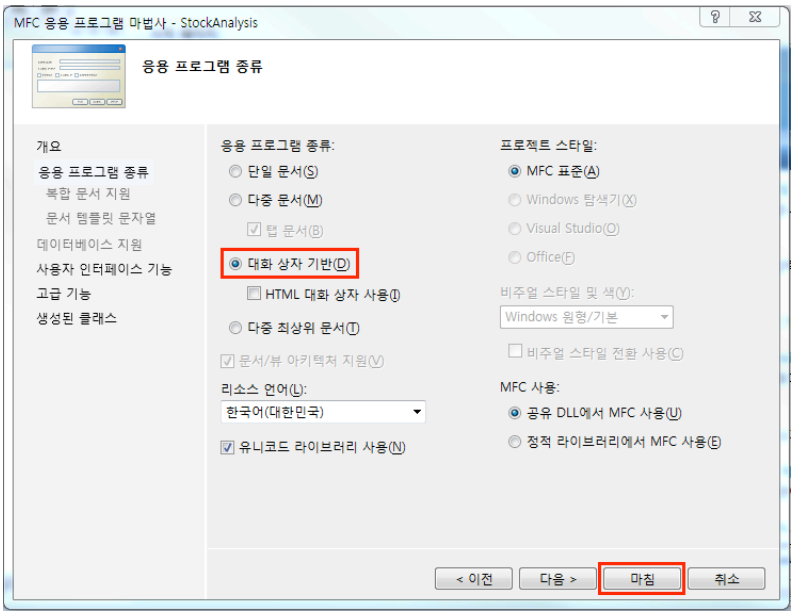
[그림 1-3]처럼 MFC 마법사가 시작되면 '다음'을 눌러서 응용 프로그램의 종류를 선택한다.

**[그림 1-3]** MFC 마법사 시작




응용 프로그램의 종류는 [그림 1-4]와 같이 '대화 상자 기반(D)'를 선택하고 '마침'을 누르면 새로운 프로젝트가 생성된다. '대화 상자 기반'은 다이얼로그Dialog 프로그램을 만드는 것으로, 응용 프로그램에 메뉴Menu 등을 생성하지 않고 빈Empty 화면에서 프로그램을 만들기 위해서다. 필자는 MFC에서 다이얼로그로 프로그램을 생성하는 것을 좋아하는데, 상업적인 프로그램이 아닌 간단한 프로그램을 만들기에 좋다. 메뉴 등을 만들면 프로그램의 완성도를 높이기 위해 코드 수정이 많이 필요하다. 이 책에서는 내용 전달에 우선을 두었으므로 기능적인 부분들은 생략한다.

**[그림 1-4]** 대화 상자 기반(다이얼로그) 프로그램 생성

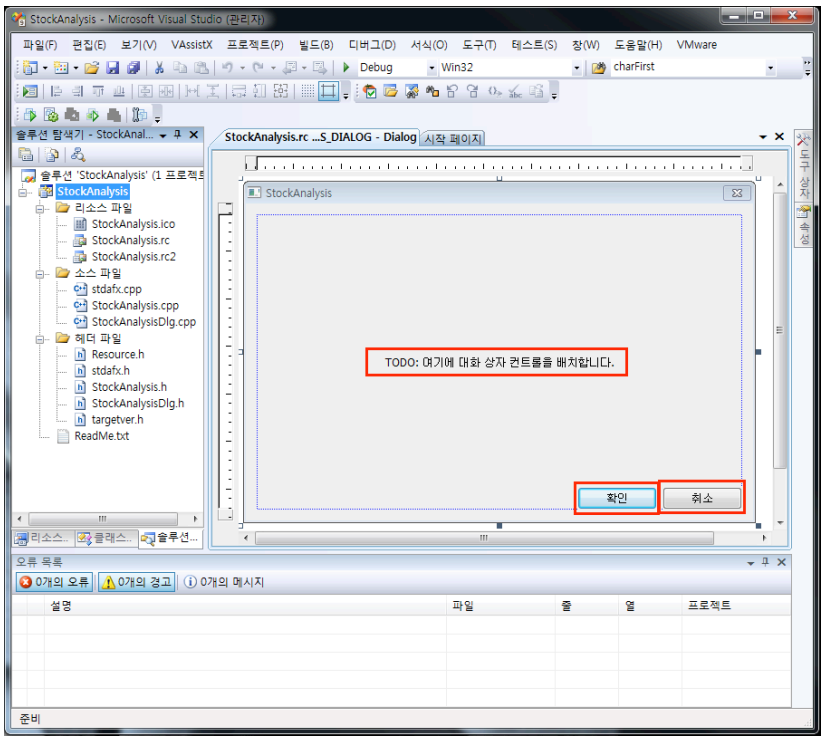



MFC에서 프로젝트를 만들면 기본적으로 생성되는 파일을 보일러플레이트Boilerplate라고 한다. 윈도우 환경에서 프로그램을 구동하는 데 필요한 파일들은 이 보일러플레이트에서 생성하고 필요한 곳에 프로그램을 만들어 나간다.

> **NOTE**
>
> 보일러플레이트를 자세히 알면 좋지만, 윈도우 API를 이해해야 하므로 필자는 권하지 않는다. 마우스나 터치를 이용한 그래픽 응용 프로그램들을 만들 때 기본적으로 생성되는 코드라고 생각하면 된다. 안드로이드나 아이폰에서도 응용 프로그램들을 만들 때 기본적인 코드가 생성된다. 그러므로 기본적으로 생성되는 코드를 어려워할 필요가 없으며 그중에서 무엇이 필요한지만 알면 된다. 고맙게도 인터넷에 많은 자료가 있어서 우리가 원하는 정보를 어렵지 않게 찾을 수 있다. 프로그래밍을 잘하는 데는 필요한 정보를 얼마나 잘 찾아 응용할 수 있는지도 중요하다. 필자는 프로그래밍에서 문제 해결 능력이 가장 중요하다고 강조하고 싶다.

MFC 응용 프로그램 마법사를 끝내면 [그림 1-5]와 같이 다이얼로그 화면이 뜬다. 다이얼로그 바탕에 기본으로 나오는 글자와 버튼들은 삭제한다. 프로그램에 필요한 모든 내용을 직접 구현하므로 하나씩 만들어 가면서 설명한다.

**[그림 1-5]** 생성된 다이얼로그 기반 프로그램




[그림 1-5]의 왼쪽에 자동으로 생성된 파일 중에서 다음 세 개의 파일이 중요하다.

- StockAnalysisDlg.h
- StockAnalysisDlg.cpp
- stdafx.h

생성된 보일러플레이트에서는 앞의 세 개 파일들만을 가지고 코드 수정이 이루어진다. 파일명 끝에 Dlg라고 이름 붙은 두 파일은 다이얼로그 응용 프로그램에서 그래프를 그리거나 버튼을 눌렀을 때 동작하는 이벤트Event 처리를 위한 코드를 만드는 데 사용된다. 이 책에서는 'StockAnalysisDlg.h'와 'StockAnalysisDlg.cpp' 두 개의 파일과 새로운 클래스Class 파일들을 생성하여 프로그램을 구현한다. 'stdafx.h' 파일은 프로그램 전체에서 전역 변수Global Variable를 사용하는 데 필요하지만, 이 책에서는 사용하지 않는다.

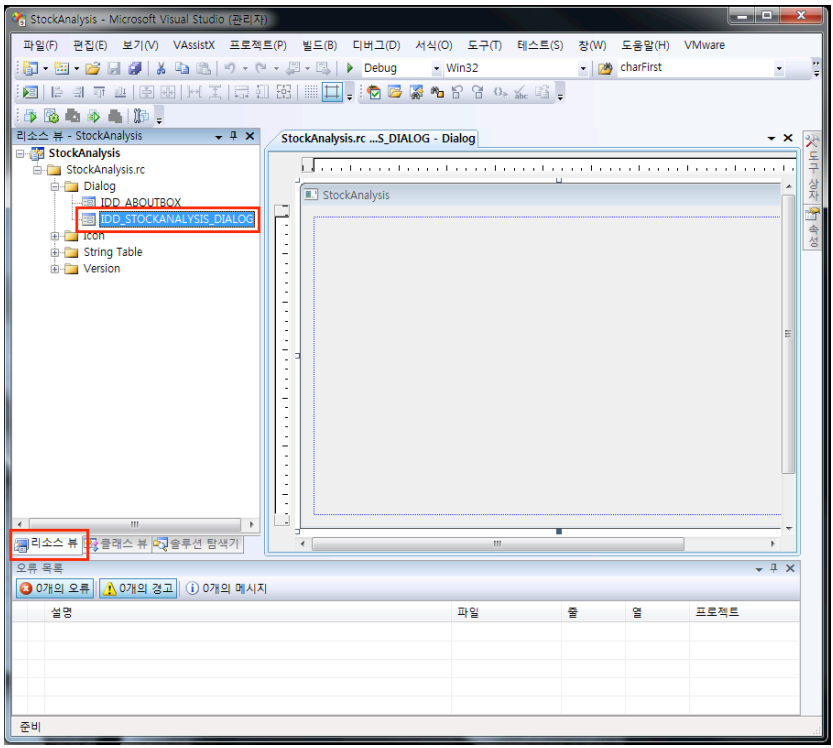
### 1.2.2 Button 만들기

앞으로 출간할 'How-to Series'에서는 다수의 MFC 프로그램을 다룬다. 이후 출간될 책들은 시리즈의 전작에서 설명한 내용은 생략한다. 따라서 이 책에서 다루는 MFC의 기초 부분은 다음 시리즈에서는 생략될 것이다.

이 절에서는 MFC의 도구 상자를 이용하여 버튼 만드는 법을 자세히 설명하겠다. 여기서 버튼 만드는 법을 이해한다면 다른 도구들도 버튼과 같은 방법으로 만들면 되므로 인터넷이나 기존에 출간된 MFC 관련 서적을 본다 해도 어렵지 않게 내용을 이해할 수 있을 것이다.

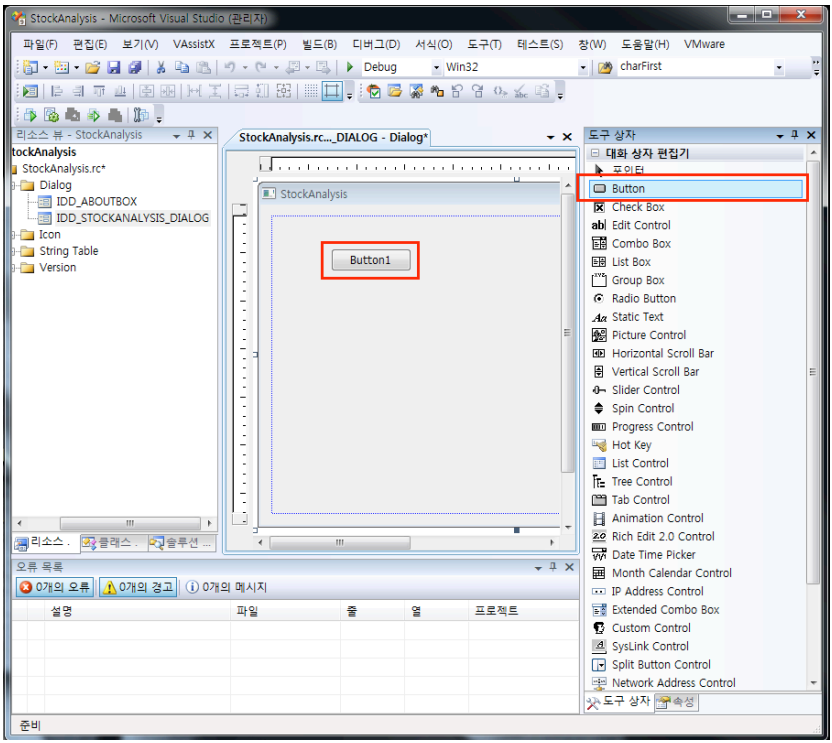
시각적인 프로그램에서 기본으로 생각할 것은 개별 단위들이 속성Property을 가진다는 것이다. 예를 들면, MFC에서는 도구 상자 안의 모든 도구가 속성을 가지며, 웹 프로그램에서는 개별 태그Tag에 속성을 주어 좀 더 보기 좋은 프로그램을 만들 수 있다.

**[그림 1-6]** 도구를 그리기 위한 다이얼로그




버튼을 그리려면 [그림 1-6]과 같이 '리소스 뷰'를 선택한 후 'Dialog' 폴더의 'IDD_STOCKANAYSIS_DIALOG'를 선택한다. 다이얼로그 창이 열리면 [그림 1-6]의 오른쪽에 위치한 '도구 상자' 탭을 클릭하여 [그림 1-7]과 같이 도구 상자를 연다. 도구 상자에서 Button을 선택한 다음 다이얼로그 창에서 마우스 왼쪽 버튼을 클릭하거나 드래그Drag하면 버튼이 자동으로 화면에 그려진다.

**[그림 1-7]** 도구 상자의 다양한 도구들

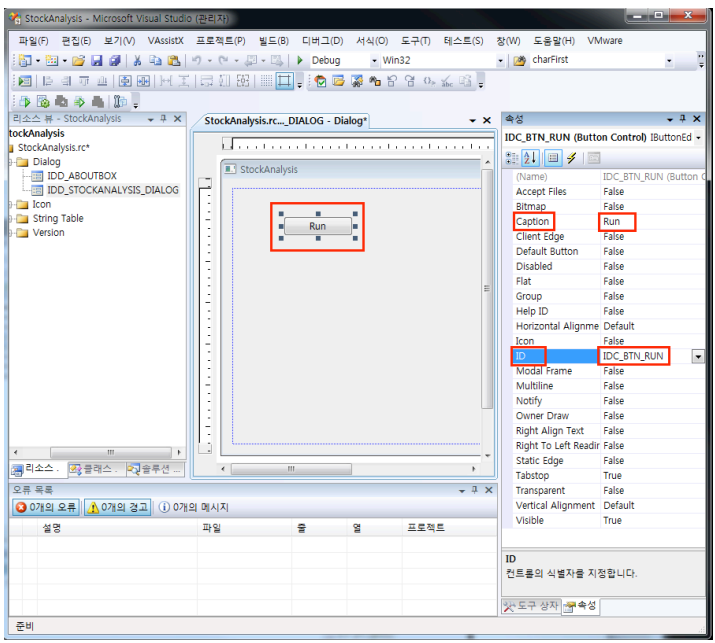



도구 상자에는 매우 다양한 도구들이 있다. 윈도우 프로그램에서 흔히 볼 수 있는 도구들로 원하는 도구를 선택해서 다이얼로그에 그리면 된다. 이 책에서는 다루지 않지만, 도구를 그릴 때 비주얼 스튜디오는 프로그램에 해당 도구를 위한 기본적인 코드들을 자동으로 생성한다. 따라서 프로그램의 깊은 부분까지 이해하지 않아도 프로그램을 쉽게 만들 수 있다.

[그림 1-6]의 오른쪽 '속성' 탭을 선택하면 [그림 1-8]과 같이 속성 창이 열린다. 다이얼로그 창에 그려진 'Button1'을 선택하면 도구 상자에서 해당 'Button'의 속성 정보가 나타나는데, 여기서 중요한 것은 'Caption'과 'ID'다. 'Caption'은 선택한

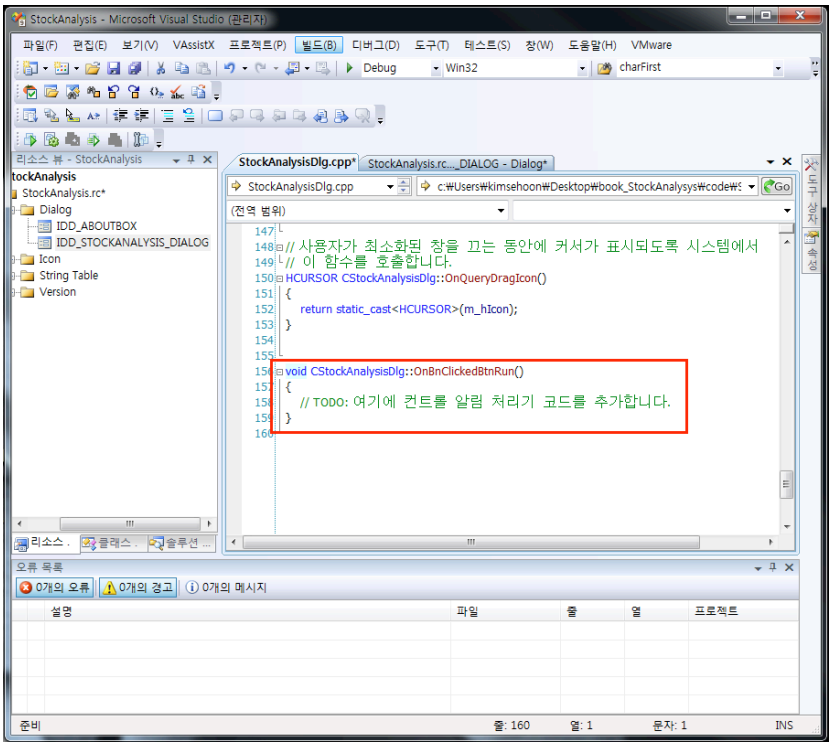
버튼 위에 써지는 텍스트 'Button1'이며(수정 가능), 'ID'는 프로그램에서 사용되는 버튼의 고유 'ID'를 지정한다. 예제에서는 'ID'를 'IDC_BUTTON1'에서 'IDC_BTN_RUN'으로 바꾼다. 이름에서 'BTN'은 'Button'의 약어이고, 뒤에 'RUN'과 같이 버튼의 이름을 써주는 것이 좋다. 프로그램에서 버튼이 많아졌을 때 이처럼 새롭게 이름을 작성해 주면 코드에서 해당 버튼을 쉽게 구분할 수 있다. 속성의 나머지 부분들은 버튼의 시각적인 부분들을 위한 것이므로 여기서는 다루지 않는다.

**[그림 1-8]** 버튼의 속성 설정하기




다이얼로그 창에 그려진 'Run' 버튼을 마우스로 더블 클릭하면 [그림 1-9]와 같이 코드가 자동으로 생성되고, StockAnalysisDlg.cpp 파일로 이동한다. 코드 아랫부분의 OnBnClickedBtnRun()은 실행 프로그램에서 'Run' 버튼을 클릭했을 때 실행되는 함수다.

**[그림 1-9]** 버튼을 눌렀을 때 실행되는 함수




실제 버튼 클릭을 위해 새롭게 생성된 코드는 [그림 1-9]에 나와 있는 것 외에 다음과 같은 내용이 더 있다.

**[StockAnalysisDlg.h]**

```
afx_msg void OnBnClickedBtnRun();
```

**[StockAnalysisDlg.cpp]**

```
BEGIN_MESSAGE_MAP(CStockAnalysisDlg, CDialog)
```

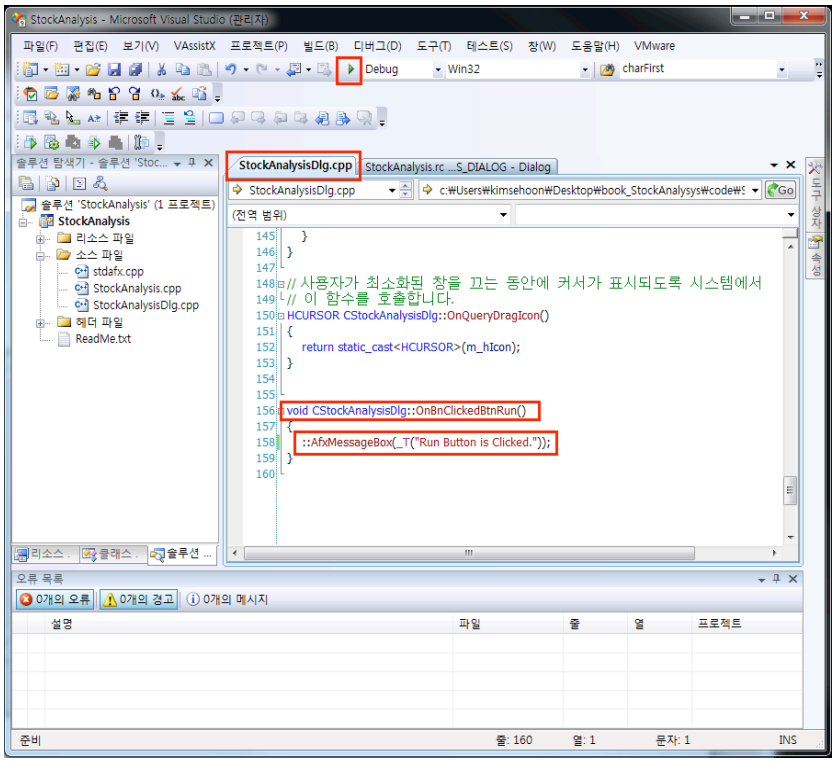
```
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BTN_RUN, &CStockAnalysisDlg::OnBnClickedBtnRun)
END_MESSAGE_MAP()
```

첫 번째 코드는 버튼을 클릭하면 실행되는 함수를 선언한 것이고, 두 번째 코드의 회색 부분은 버튼을 클릭했을 때 버튼과 해당 함수를 연결한다. 두 번째 코드는 ID가 'IDC_BTN_RUN' 버튼을 클릭했을 때(ON_BN_CLICKED) 'OnBnClickedBtnRun()' 함수를 실행하도록 연결하는 부분이다. 비주얼 스튜디오는 자동으로 이러한 코드를 만들기 때문에 프로그래머가 좀 더 쉽게 프로그래밍을 할 수 있으며, 해당 함수 내에서만 코드 작업을 해주면 된다.

OnBnClickedBtnRun() 함수를 눌렀을 때 메시지 창Message Box를 출력하려면 [그림 1-10]과 같이 OnBnClickedBtnRun() 함수 안에 AfxMessageBox() 함수를 사용한다. 필자는 MFC 프로그램에서 디버깅Debugging할 때 매우 유용한 AfxMessageBox() 함수를 자주 사용한다. AfxMessageBox() 함수의 괄호 안에는 MFC에서만 쓰는 'CString'이라는 문자열 클래스Class를 사용해야 하지만, '_T' 매크로를 이용하여 직접 문자열을 사용해도 된다. AfxMessageBox() 함수는 다음과 같이 사용하여 "Run Button is Clicked."라는 메시지를 출력할 수 있다.

```
::AfxMessageBox(_T( " Run Button is Clicked. " ));
```

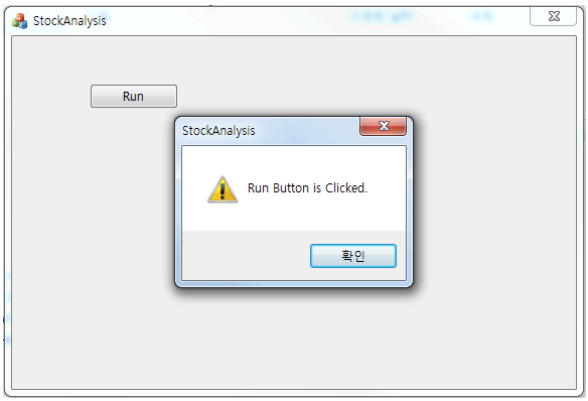
**[그림 1-10]** 버튼을 눌렀을 때 실행되는 코드




마지막으로 실행 프로그램을 만들어 테스트해 보자. 비주얼 스튜디오의 메뉴에 있는 '플레이Play' 버튼을 눌러도 되고, 단축키로 'Ctrl + F5'를 사용해서 컴파일하고 실행 프로그램을 만들어도 된다. 프로그램을 만들면서 실행 프로그램을 자주 만들어야 하므로 단축키 'Ctrl + F5' 사용을 권한다. 모든 단축키를 숙지하는 것은 힘들지만, 자주 사용하는 단축키 정도는 기억하는 것이 좋다.

프로그램을 실행하면 [그림 1-11]과 같이 'Run' 버튼 하나만 나타나며, 'Run' 버튼을 누르면 "Run Button is Clicked."라는 메시지를 출력하는 창이 생성된다.

**[그림 1-11]** 실행 프로그램




프로그래밍 언어 기초 책을 보면 항상 나오는 것이 "Hello World"라는 문자열을 출력하는 것이다. 지금까지 버튼을 이용해서 "Hello World"라는 아주 간단한 프로그램을 만들었다고 보면 된다. "Hello World" 프로그램이 항상 먼저 나오는 이유는 오류Error 없이 프로그램을 만들어서 실행되는지를 알 수 있기 때문이며, 이는 중요한 부분이다. 이후부터는 요소를 하나씩 추가하면서 프로그램을 만들어 나가면 된다. 또한, 코드를 조금씩 추가하면서 실행 프로그램을 자주 만들어서 테스트하는 것도 중요하다.[01]

도구 상자 안에 있는 다른 도구들도 이와 같은 방식으로 그려주고 속성을 사용하면 된다. 직접 프로그래밍해 보고 시행착오를 거쳐 알게 된 내용이 오래 기억되므로 여러 가지 도구들을 직접 그려보기 바란다. 또한, 인터넷에 개별 도구들에 대한 자세한 설명이 나와 있으므로 직접 찾아보기를 권한다. 경험상 프로그래밍은 내가 아는 것만을 가지고 프로그램을 만드는 것은 극히 일부분이며, 무엇이 필요한지를 알고 방법을 찾아서 새로운 것을 만드는 일이라고 생각한다.

---

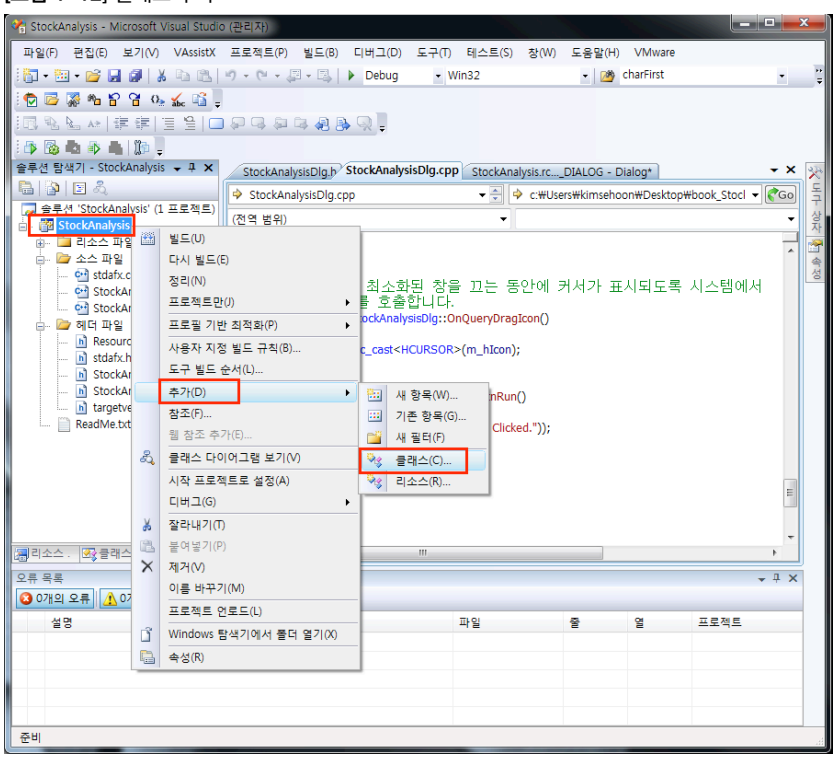
01 'How-to Series'의 첫 번째 책인 『소수와 RSA 알고리즘으로 배우는 Big Number 연산』(한빛미디어, 2013)의 부록에 이 부분이 자세히 설명되어 있으므로 참고하기 바란다.

### 1.2.3 Class 추가하기

MFC는 C++에 기반을 둔 프로그램이므로 구현하려는 C++ 클래스를 추가하여 프로그램을 만든다. 이 책의 내용 대부분은 새롭게 생성된 클래스 안에서 구현할 것이다. 단지 그래프를 그리거나 버튼 등을 눌렀을 때 동작하는 부분만 StockAnalysisDlg.cpp에 구현한다. 따라서 프로그램의 알고리즘과 같이 중요한 부분들은 추가된 클래스 안에서 새롭게 구현한다.

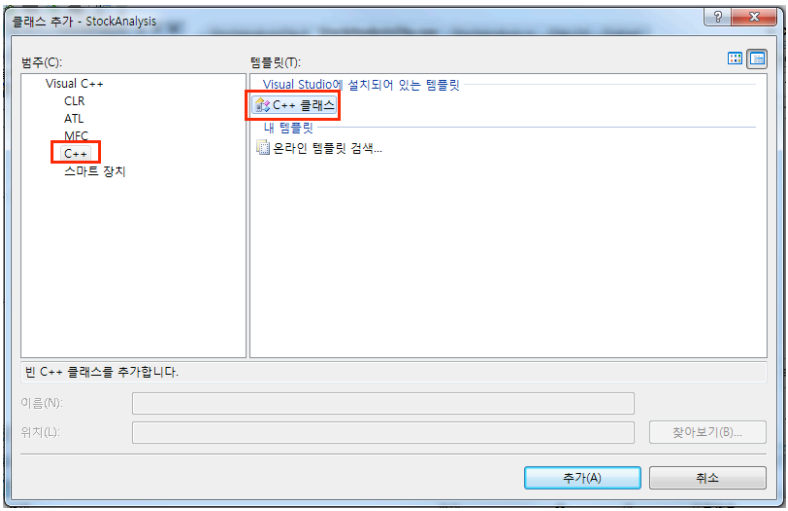
새로운 클래스를 추가하려면 [그림 1-12]와 같이 프로젝트 이름에서 마우스 오른쪽 버튼을 눌러, '추가(D) → 클래스(C)'를 선택한다.

**[그림 1-12]** 클래스 추가



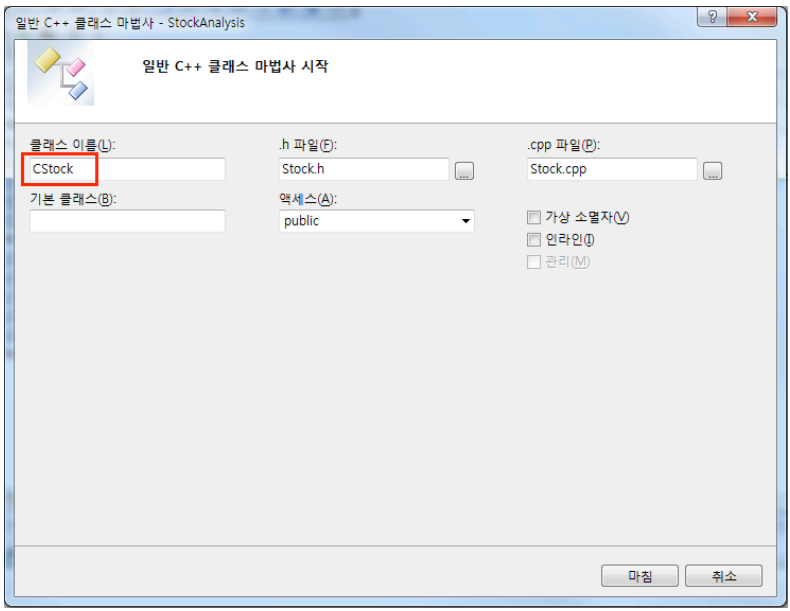
[그림 1-13]과 같이 '클래스 추가' 창이 열리면 'C++ → C++ 클래스'를 선택하고 '추가' 버튼을 누른다.

**[그림1-13]** C++ 클래스 추가



[그림 1-14]와 같이 C++ 클래스 마법사 창이 열리면 클래스 이름을 'CStock'으로 지정한다.[02] 그러면 첫 문자 C가 생략된 'Stock.h'와 'Stock.cpp' 파일이 생성되고, 'Stock'이라는 클래스에 앞으로 구현하게 될 코드들이 생성된다.

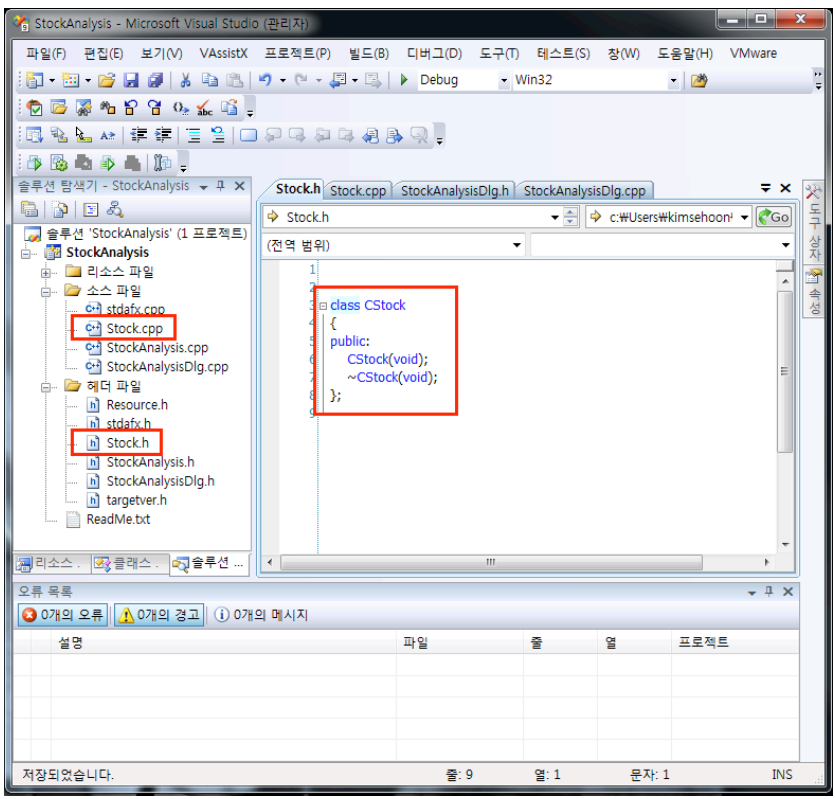
**[그림 1-14]** 클래스 이름 정하기




02 MFC는 클래스 이름을 정할 때 항상 대문자 'C'를 앞에 써주는 것이 좋다.

[그림 1-15]를 보면 'CStock' 클래스와 관련된 두 개의 파일이 생성되었고, 내용이 없는 빈 클래스가 선언되었음을 알 수 있다. 이 책에서는 두 개의 C++ 클래스에 코드가 구현되는데, 'Stock'이라는 클래스와 그래프를 그리기 위해 계산을 하는 'Graph' 클래스다. 'CGraph' 클래스도 'CStock' 클래스와 같은 방법으로 추가한다.

**[그림 1-15]** 추가된 클래스




MFC 프로그램을 만들어 본 독자라면 지금까지의 내용은 너무 간단했을 것이다. 하지만 이 책에서는 MFC 프로그래밍을 처음 해보는 독자들도 많을 것으로 판단되어 각 단계를 좀 더 구체적으로 기술하였다.

다음 장은 이 책에서 가장 중요한 부분으로, 데이터를 읽어와서 그래프를 어떻게 그리는지를 설명한다. 여기서 만드는 프로그램은 상업적인 프로그램이 아니므로 간단한 프로그램을 통하여 기본적인 부분을 익히고, 자신만의 기능을 추가하여 개인 프로그램을 만들어 보기 바란다.

# 2 | Chart 그리기

이번 장에서는 MFC에서 주식 그래프를 그려본다. 주식 데이터를 다루는 'CStock' 클래스와 그래프를 그리기 위해 계산하는 'CGraph' 클래스, 그리고 화면에 그래프를 그려주는 'CStockAnalysisDlg' 클래스에서 프로그램이 구현된다. 여기서 사용되는 데이터는 'data.txt' 파일에 저장되어 있는데, 그 내용은 다음과 같으며 최근 데이터는 부록으로 제공되는 'GetStockData' 프로그램을 사용해서 얻을 수 있다.

```
2130
A000020  동화약품  250
20130823  6140  6220  6020  6120  83828
20130822  6300  6300  6110  6120  119820
……생략……
20120823  5400  5510  5390  5490  84425
A000040  S&T모터스  250
20130823  510  513  502  506  255864
20130822  500  513  488  498  721021
……생략……
```

'data.txt'에서 가장 위에 '2130'은 전체 종목 수를 나타낸다. 즉, 'data.txt'에는 2,130개의 종목 데이터가 저장되어 있다. 두 번째 줄에서 'A000020'은 종목 번호이며, '동화약품'은 종목 이름, 그리고 '250'이라는 숫자는 해당 종목과 관련한 250개의 데이터가 밑에 기록되어 있음을 나타낸다. 250개의 데이터는 250줄에 기록되어 있는데, 한 줄에는 날짜, 시가, 고가, 저가, 종가, 거래량 순으로 저장되어 있다. 'data.txt'에 저장된 형식은 필자가 임의대로 정한 것으로, 개인적으로 프로그램을 만든다면 데이터를 보기 편한 방식으로 저장하면 된다.

## 2.1 Data Structure

주식 프로그램의 시작은 데이터를 읽어와서 프로그램의 구조체 안에 저장하는 것이다. 그렇다면 데이터를 담기 위한 구조체Structure를 어떻게 만들면 좋을까? 필자는 개별 종목 하나를 하나의 구조체에 담고, 이렇게 만들어진 구조체를 배열Array로 관리한다. 프로그램을 만들 때 같은 자료형과 형태의 데이터는 배열로 만드는 것이 좋다. 종목 데이터를 구조체에 넣는 방법은 여러 가지가 있는데 다음과 같이 두 가지 형태의 구조체로 표현할 수 있다.

**[코드 2-1]** Stock Data 구조체 (1)

```
struct Company {
      CString strJongMok, strName;      //종목 번호, 종목 명
      int quantity;                     //데이터 개수
      long date[250];                   //날짜
      long startVal[250];               //시가
      long highVal[250];                //고가
      long lowVal[250];                 //저가
      long lastVal[250];                //종가
      long vol[250];                    //거래량
};

Company allCompany[2130];         //2130개의 모든 종목 구조체를 배열로 선언
```

**[코드 2-2]** Stock Data 구조체 (2)

```
struct Data {
      long date;                        //날짜
      long startVal;                    //시가
      long highVal;                     //고가
      long lowVal;                      //저가
```

```
    long lastVal;                    //종가
    long vol;                        //거래량
};

struct Company {
    CString strJongMok, strName;     //종목 번호, 종목 명
    int quantity;                    //데이터 개수
    Data data[250];                  //주가 데이터
};

Company companies[2130];             //2130개의 모든 종목 구조체를 배열로 선언
```

[코드 2-1]의 구조체는 이전에 개인적으로 만든 프로그램에서 구현한 방법이고, [코드 2-2]의 구조체는 이 책에서 만드는 프로그램에 사용할 방법이다. [코드 2-1]은 주식 데이터를 날짜가 아닌 최소 단위의 데이터 배열로 나타냈고, [코드 2-2]는 날짜에 따른 데이터를 구조체에 담아서 구조체를 배열로 만들었다. [코드 2-2]에서 'Company' 구조체를 배열로 만들었는데 여기의 'companies'라는 변수를 가지고 모든 데이터에 접근할 수 있다.

프로그램을 만들 때 중요한 것 중 하나는 많이 사용하는 상수를 정의해서Define 사용하는 것이다. 헤더 파일에 'Data' 구조체의 배열 크기와 모든 종목 개수의 최대 크기를 정의하는 것이 좋다. 이 책에서는 'Stock.h'에 다음과 같이 상수들을 정의한다.

```
#define  MAX_DATA 250            //한 종목당 250개의 데이터를 가진다.
#define  MAX_COMPANY 2500        //2130개 종목보다 큰 수로 설정한다.
```

이처럼 상수를 정의하면 헤더 파일에서 상수값만 변경해서 프로그램의 크기를 쉽게 바꿀 수 있다. 예를 들어, 종목당 250개의 데이터를 1000으로 바꾸면 종목당

1,000개의 데이터를 가지고 처리할 수 있는 프로그램으로 쉽게 바뀐다. 또한, 상수를 정의하면 프로그램의 가독성을 높여준다. 코드 안에서 250이라는 숫자를 사용하면 이것이 무엇을 의미하는지 알기가 힘들다. 그러나 숫자 대신 'MAX_DATA'라는 새롭게 정의된 상수를 써주면 이름을 통하여 데이터의 최대값임을 쉽게 알 수 있다. 이처럼 사용하면 프로그램 구현이 쉬워지므로 상수를 정의해서 사용하기를 권한다.

다음 [코드 2-3]은 'Stock.h'에서 사용하는 코드다.

**[코드 2-3]** Data 구조체 (Stock.h)

```
#define MAX_DATA     250
#define MAX_COMPANY  2500

struct Data {
    long date;                    //날짜
    long startVal;                //시가
    long highVal;                 //고가
    long lowVal;                  //저가
    long lastVal;                 //종가
    long vol;                     //거래량
};

struct Company {
    CString strJongMok, strName;  //종목 번호, 종목 명
    int quantity;                 //데이터 개수
    Data data[MAX_DATA];          //주가 데이터
};

struct AllCompany {
    int quantity;                 //전체 종목 개수
    Company companies[MAX_COMPANY]; //2500개 종목 구조체를 배열로 선언
```

```
};

class CStock
{
public:
02      AllCompany allCompanies;
public:
    CStock(void);
    ~CStock(void);
};
```

01 AllCompany 구조체는 이후에 추가되는 이평선Day Moving Average, 이동평균선이나 보조 지표 등 모든 종목의 데이터를 포함하기 위해 추가한다. 여기서는 companies라는 Company 구조체 배열만을 가지고 있지만, 이후에 만들어지는 구조체는 이곳에 변수로 선언된다.

02 CStock 클래스에서 모든 데이터에 접근하려면 allCompanies라는 변수를 public으로 선언한다. 외부에서 CStock 클래스를 생성하면 모든 종목의 데이터에 대한 접근이 가능하다.

이전 책[01]에서 강조했듯이 프로그램에서 구조체를 이해하는 것이 가장 중요하다. 데이터가 어떻게 프로그램에서 구조화되는지를 이해해야만 나머지 부분을 쉽게 만들 수 있다. 특히, 주식 프로그램에서는 데이터를 가져와서 구조화할 수 있다면 프로그램의 반은 끝났다고 볼 수 있다. 물론, 여기서는 증권회사에서 실제 데이터를 가져오지는 않았지만, 직접 증권회사에서 데이터를 가져와서 구조체로 넣는다면 실시간 자동매매 프로그램을 쉽게 구현할 수 있을 것이다.

## 2.2 File Data 읽기

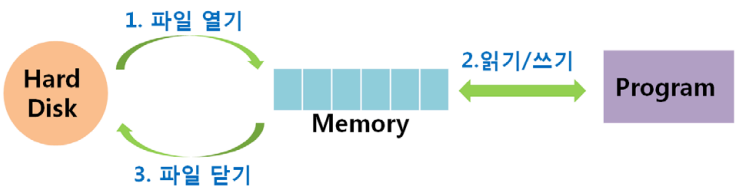
MFC에서 파일을 읽고 쓰는 것은 C++에 기반을 두고 있어서 특별히 고려해야 할 것이 없다. 또한, C++에서 C 함수도 사용할 수 있으므로 이 책에서는 C 함수를 사용하여 파일에 읽고 쓰는 작업을 할 것이다.

01 『소수와 RSA 알고리즘으로 배우는 Big Number 연산(http://goo.gl/fNfOVw)』(한빛미디어, 2013)

MFC에서 C 함수를 사용할 때 고려해야 할 점은 자료형Data Type이다. 즉, 데이터를 MFC에서만 사용하는 문자열 자료형인 'CString'으로 변환하는 작업이 필요하다. 파일에서 데이터를 읽을 때는 C 함수로 읽어서 'char *' 형태의 문자열을 'CString'으로 형변환해야 하며, 파일에 기록할 때는 'CString'을 'char *' 형태로 변환해서 C 함수로 기록한다.

프로그램을 만들다 보면 파일 입출력을 많이 다루게 된다. 잠시 파일 입출력이 어떤 원리로 이루어지는지 알아보자.

**[그림 2-1]** 파일 입출력




[그림 2-1]과 같이 파일은 하드디스크에 저장되어 있다. 프로그램에서 파일을 열면Open 파일의 모든 내용은 메모리에 올라간다. 파일을 열 때 읽기Read만 가능한지 아니면 쓰기Write도 가능한지를 지정하는데, 읽기만 가능한 파일은 메모리에서 읽을 수만 있고, 쓰기가 가능한 파일은 읽고 쓰기가 가능하다. 또한, 파일의 마지막부터 추가하여 쓰기가 가능한 'Append' 속성을 부여할 수도 있다. 그리고 파일이 메모리에 올려져 있으므로 특정 위치에서 읽기와 쓰기도 할 수도 있다. 이 내용은 프로그램 관련 기초 서적이나 인터넷에서 쉽게 찾을 수 있으므로 여기서는 다루지 않는다.

파일 열기로 파일이 메모리에 올려진 후에 프로그램은 메모리에서만 읽기/쓰기를 하게 된다. 프로그램에서 읽기/쓰기 작업이 다 이루어진 후 '파일 닫기'를 하면, 읽기 속성의 파일은 연결이 끊어지고 쓰기 속성의 파일은 메모리의 내용을 하드디스

크에 저장한다. 간단하게 파일 입출력은 크게 '파일 열기', '메모리에서 파일 내용 작업', '파일 닫기'의 세 단계로 구분하여 이해하는 것이 좋다.

파일에서 읽기를 간단히 표현하면 다음과 같이 나타낼 수 있다.

```
FILE *fp;                              //파일 자료형 - FILE
fp = fopen( " fileName " , "rt");      //파일 열기 - fopen
fscanf(fp, "%d \n", &variable);        //파일 읽기 - fscanf
fclose(fp);                            //파일 닫기 - fclose
```

파일을 열 때 fopen( ) 함수 안의 'rt'로 속성을 주는데, 'r'은 Read를, 't'는 Text로 읽겠다는 의미이다. 모든 줄에 사용되는 변수 fp는 File Pointer로, 메모리에 올려진 파일의 주소를 가리킨다. 즉, 'FILE *fp'는 변수 fp를 파일 포인터로 사용한다는 것이고, 두 번째 줄의 'fp = fopen("fileName", "rt")'는 'fileName'이라는 파일을 하드디스크에서 읽어와 메모리에 올리고 fp가 메모리에 올려진 파일을 가리키고 있다고 생각하면 된다. 세 번째 줄의 'fscanf(fp, "%d \n", &variable)'은 fp 파일에서 정수형(%d) 값을 'variable'이라는 변수로 읽어오는 것이다. 이때 주의할 것은 변수 앞에 '&' 문자를 써주어야 한다. '&' 문자를 사용하면 메모리에서 변수의 주소값을 나타내므로 "메모리에 있는 variable 변수의 공간에 데이터를 집어넣어라"는 의미가 된다. 마지막 줄의 'fclose(fp)'은 처리가 끝난 fp 파일을 닫는다.

파일에 쓰기는 읽기와 비슷하며 다음과 같이 나타낼 수 있다.

```
FILE *fp;                              //파일 자료형 - FILE
fp = fopen( " fileName " , "wt");      //파일 열기 - fopen
fprintf(fp, "%d \n", variable);        //파일 쓰기 - fprintf
fclose(fp);                            //파일 닫기 - fclose
```

다음은 Cstock( ) 함수를 선언하고 기본 뼈대를 구현한 부분이다.

**[코드 2-4]** CStock 함수 선언(Stock.h)

```
class CStock
{
public:
	AllCompany allCompanies;
public:
	CStock(void);
	~CStock(void);

	void Run();
	void ReadDataFromFile();
	void WriteDataToFile();
};
```

**[코드 2-5]** CStock 함수 구현 전 뼈대(Stock.cpp)

```
void CStock::Run()
{
	ReadDataFromFile();
	WriteDataToFile();
}

void CStock::ReadDataFromFile()
{
}

void CStock::WriteDataToFile()
{
}
```

**[코드 2-6]** CStock 변수 정의(StockAnalysisDlg.h)

```
#include "Stock.h"

#pragma once

class CStockAnalysisDlg : public CDialog
{
private:
     CStock *stock;
     ##### 생략 #####
};
```

**[코드 2-7]** CStock 변수 생성과 실행(StockAnalysisDlg.cpp)

```
BOOL CStockAnalysisDlg::OnInitDialog()
{
     ##### 생략 #####
     stock = new CStock();
     ##### 생략 #####
}

##### 생략 #####

void CStockAnalysisDlg::OnBnClickedBtnRun()
{
     stock->Run();
}
```

지금까지 구현한 코드를 'Ctrl + F5' 키로 컴파일이 잘 되는지 확인한다. 이제는 CStock 클래스에 선언된 'ReadDataFromFile()' 함수와 'WriteDataToFile()' 함수 안의 내용을 구현한다. 'ReadDataFromFile()' 함수는 [코드 2-8]과 같이 구현한다.