

# Chapter 1. A Tutorial Introduction

March, 2016  
Seungjae Baek

Dept. of software  
Dankook University

<http://embedded.dankook.ac.kr/~baeksj>

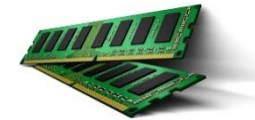
- 컴퓨터 구조를 이해한다.
- 하드웨어와 소프트웨어를 이해한다.
- 개발환경에 대한 이해
- C 프로그램을 컴파일하는 방법을 이해한다.
- C 언어의 기본 요소를 이해
- printf 함수 이해
- 이 장의 결론

## ■ Hardware / Software

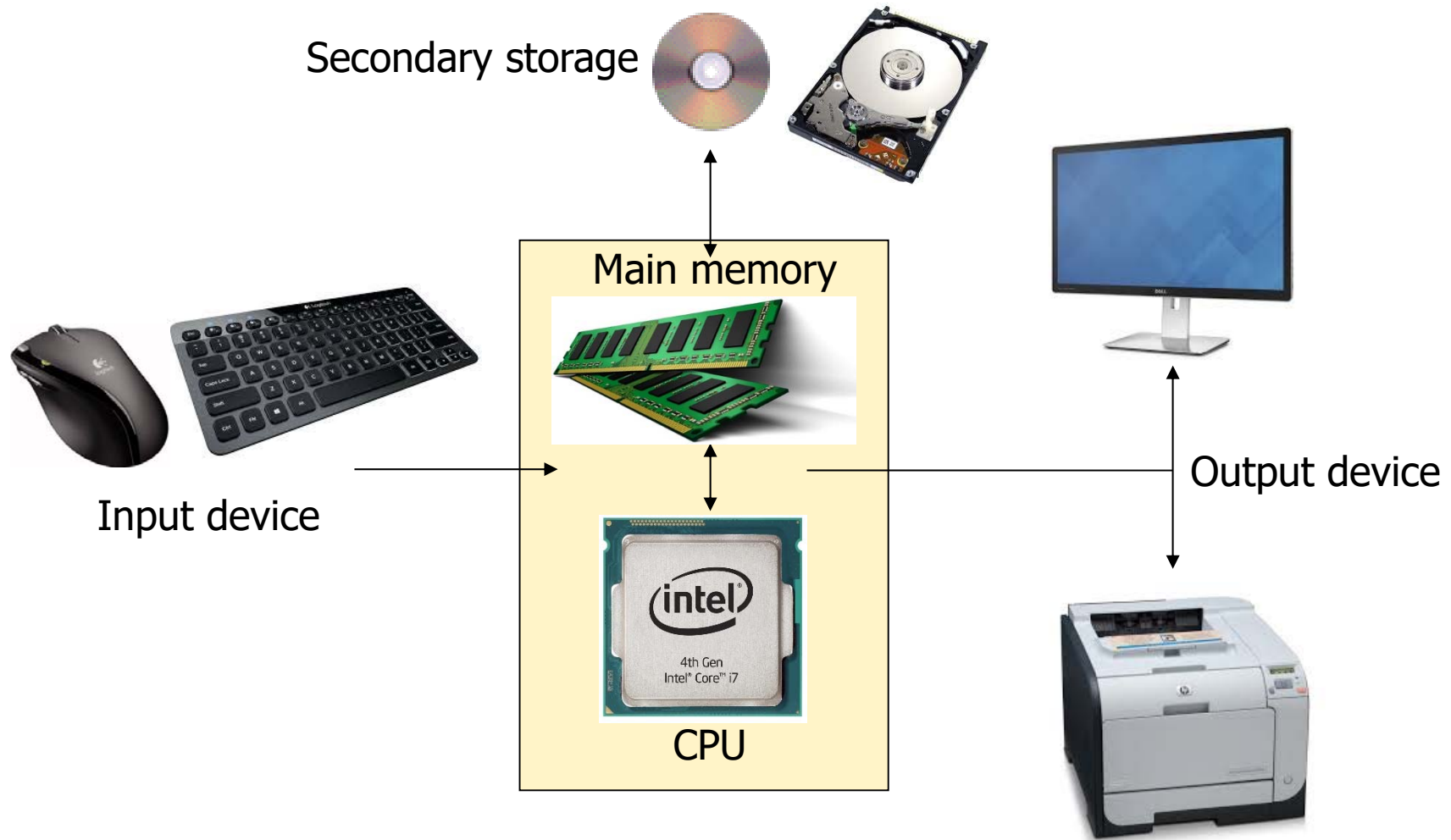
소프트웨어



하드웨어

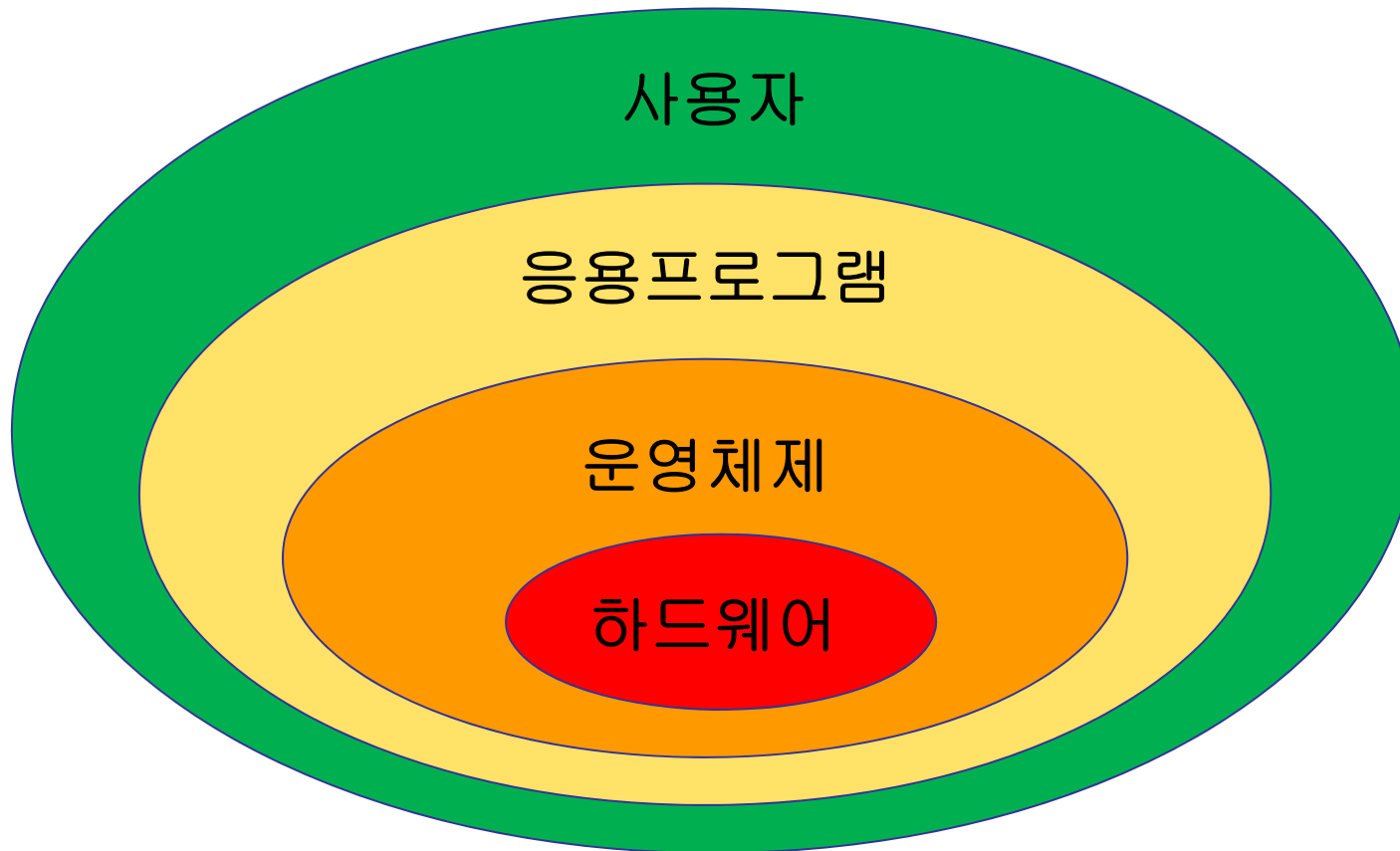


## ■ 하드웨어



- 하드웨어 구성 요소
  - ✓ 중앙 처리 장치 (central processing unit)
    - 컴퓨터의 두뇌 : 데이터 연산, 논리 연산 (ALU), 제어(control Unit)
    - 레지스터 (register)
    - x86, ARM, PPC, Sparc, Alpha, MIPS, SH4, XScale,
  - ✓ 메인 메모리 (memory) : 휘발성
    - 메모리 셀 : 메모리 내의 개별적인 저장 공간
    - 바이트(byte)와 비트(bit)
    - Address → Memory → Data
  - ✓ 저장 장치 (storage device) : 비휘발성
    - 디스크, CD-ROM, 플로피, Flash Memory(NOR, NAND 등)
  - ✓ 입출력 장치
    - 입력 장치 (input device) : 키보드, 마우스, Key Pad, Touch Screen
    - 출력 장치 (output device) : 모니터, 프린터, LCD
  - ✓ 통신 장치
    - 모뎀(modem), 이더넷(Ethernet), IrDA, CDMA, Bluetooth

- 소프트웨어



## ■ 소프트웨어 구성 요소

### ✓ 운영체제

- 자원 관리자(resource manager)
- 물리적 자원/추상적 자원

### ✓ 응용 프로그램

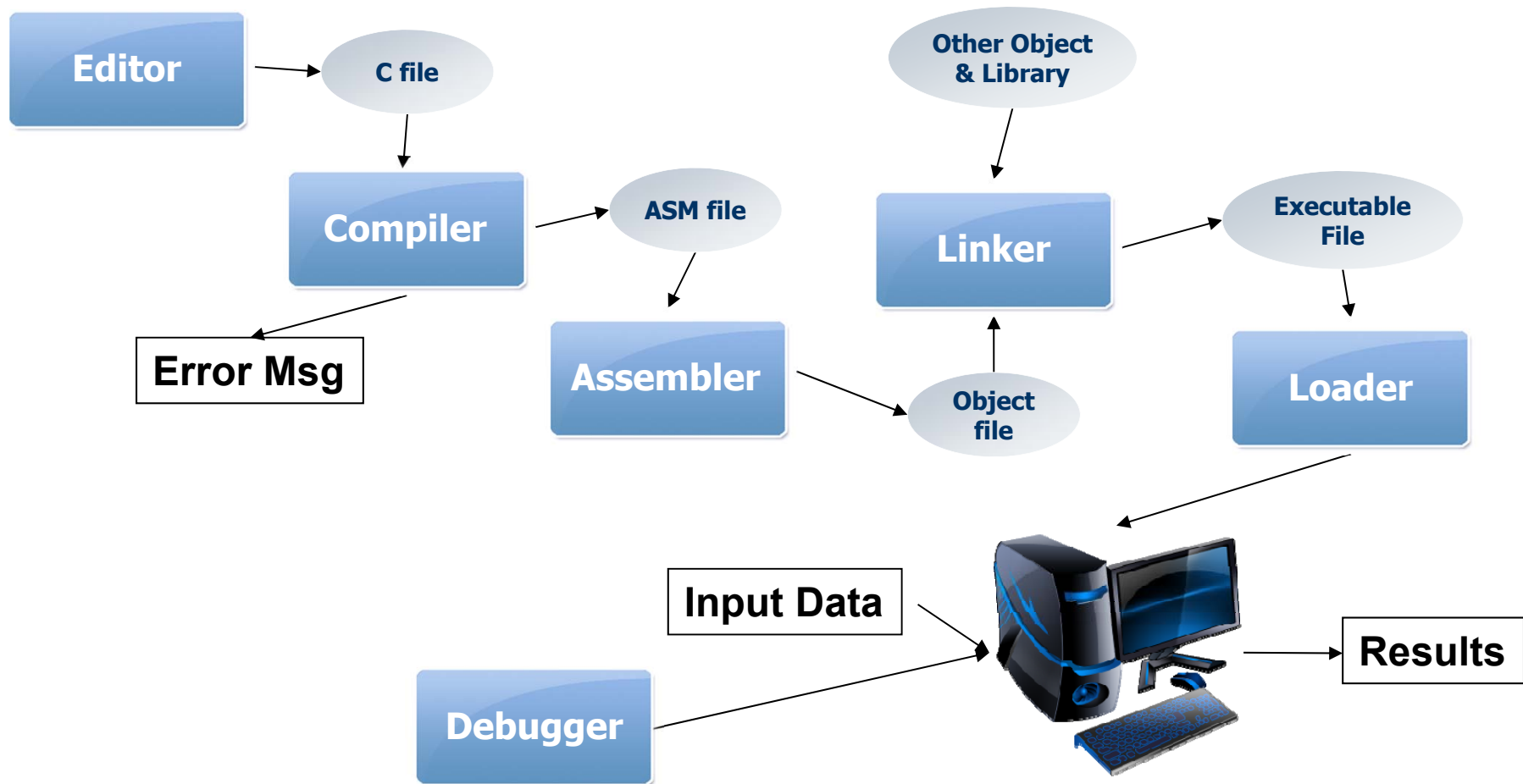
- 사용자의 특정 목적을 달성하게 하는 소프트웨어
- 워드프로세서, 게임 소프트웨어
- 데이터베이스
- 컴파일러

### ✓ 프로그램 언어

- 프로그래밍 언어
  - ✓ 개발자와 컴퓨터간에 약속
  - ✓ 구문(syntax)과 의미(semantics)로 구성
- 프로그래밍 언어 구분
  - ✓ 기계어 (machine language): 컴퓨터가 바로 실행 가능
    - 이진(binary) 코드, CPU의 종류에 따라 다름
  - ✓ 어셈블리어 (assembly language)
    - 기계어 명령어와 1:1 대응
    - 어셈블러 (assembler)
  - ✓ 고급 언어 (high-level language)
    - 하나의 명령문이 어셈블리어 명령문 여러 개에 해당
    - 컴파일러(compiler) 또는 인터프리터(interpreter)
    - C, C++, Java, Basic, Pascal, Perl, ...



## Overall of Compilation



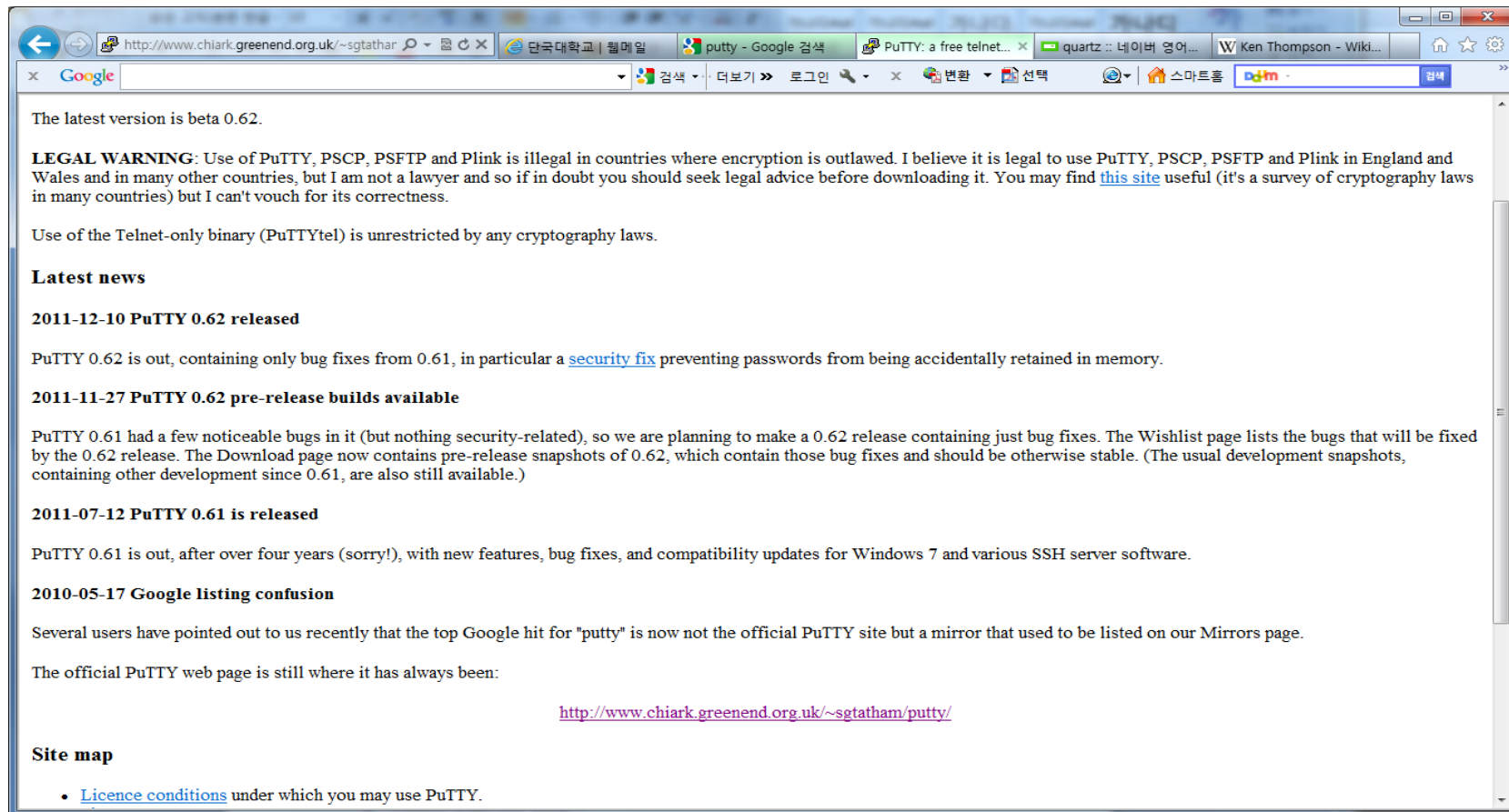
- Standalone (usually with multi-boot)
- Virtualization
- Client-Server



- ✓ In our course
  - Client: terminal emulator (telnet client, putty, ...)
  - Server: Linux system (PC)
    - IP: 220.149.236.4

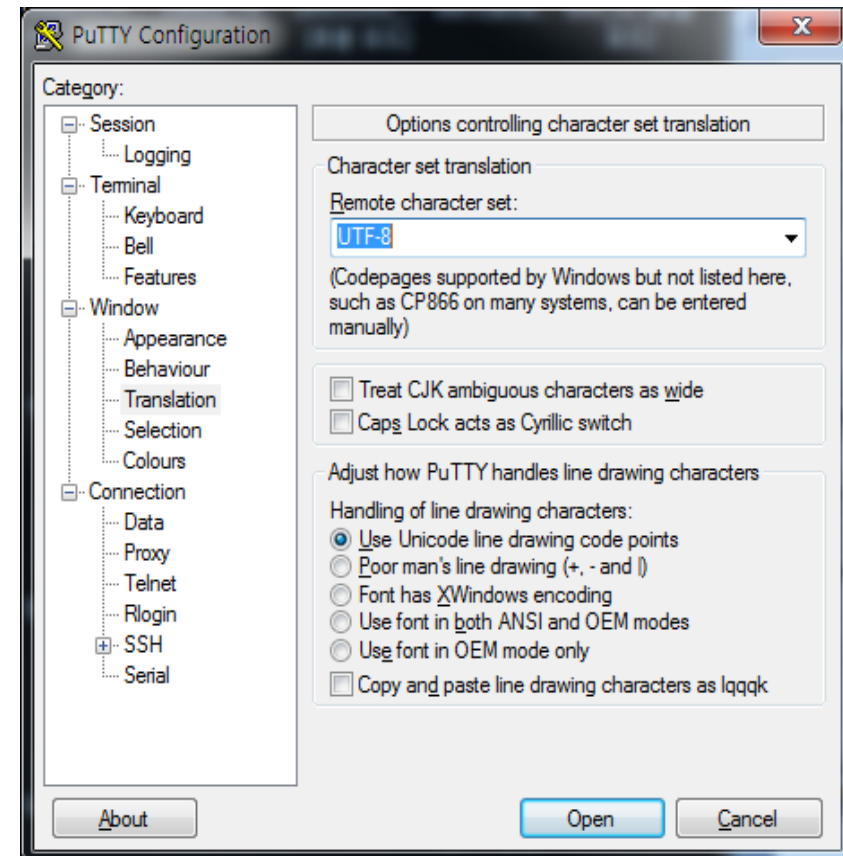
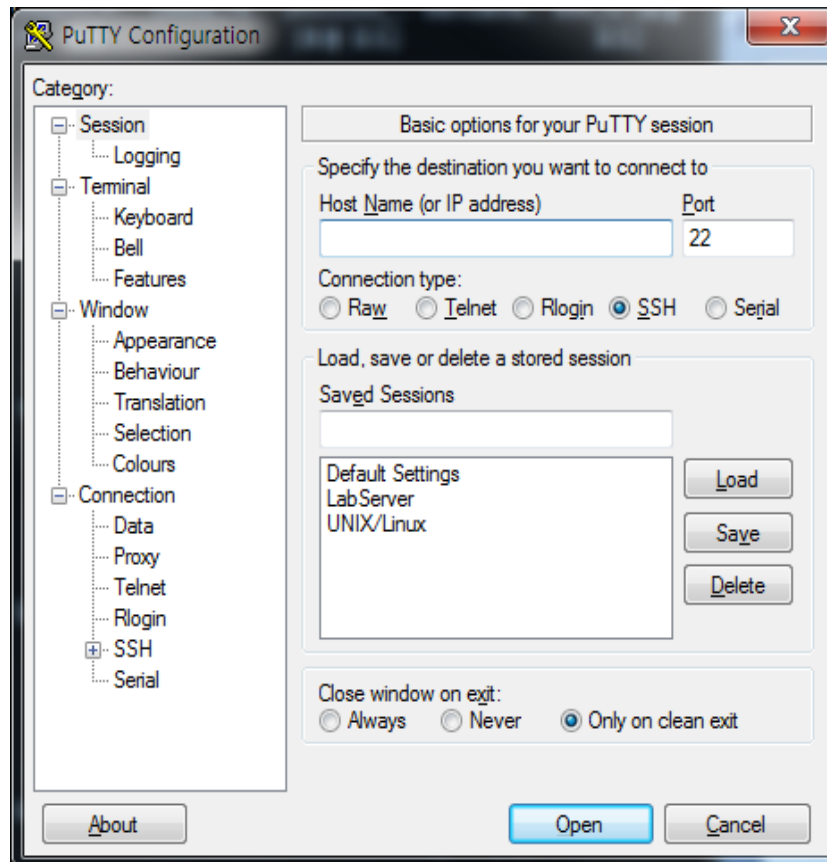
## ■ Client

- ✓ telnet, ssh, ping, ...
- ✓ putty, SecureCRT, mlterm, ...



## ■ Putty with ssh

- ✓ IP: 220.149.236.4 (type이 ssh인지, port가 22인지 확인)
- ✓ Translation: choose “UTF-8”



## ■ Login and shell

```
ssh -2 baeksj@220.149.236.4 - ssh - 80x24
baeksj@220.149.236.4's password: █
```

```
baeksj@embedded4:~ - ssh - 80x33
baeksj@220.149.236.4's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.11.0-15-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your current Hardware Enablement Stack (HWE) is no longer supported
since 2014-08-07. Security updates for critical parts (kernel
and graphics stack) of your system are no longer available.

For more information, please see:
http://wiki.ubuntu.com/1204_HWE_EOL

There is a graphics stack installed on this system. An upgrade to a
supported (or longer supported) configuration will become available
on 2014-07-16 and can be invoked by running 'update-manager' in the
Dash.

Last login: Sun Sep 13 16:16:39 2015 from baeksj-imac.local
baeksj@embedded4:~$ passwd
baeksj에 대한 암호 변경 중
(현재) UNIX 암호:
새 UNIX 암호 입력:
새 UNIX 암호 재입력:
passwd: 암호를 성공적으로 업데이트했습니다
baeksj@embedded4:~$ ls
examples.desktop
baeksj@embedded4:~$ █
```

- ✓ ID: sys123456 (학번)
- ✓ Default passwd: \*\*\*\*\* (passwd 명령을 사용하여 바꿀 것)

## ■ Overview of UNIX/Linux Commands

### COMMANDS

Currently defined functions include:

```
[, [, addgroup, adduser, adjtimex, ar, arp, arping, ash, awk,
basename, bbconfig, bbsh, brctl, bunzip2, busybox, bzip2,
cal, cat, catv, chat, chatr, chcon, chgrp, chmod, chown,
chpasswd, chpst, chroot, chrt, chvt, cksum, clear, cmp, comm, cp,
cpio, crond, crontab, cryptpw, ctttyhack, cut, date, dc, dd,
deallocvt, delgroup, deluser, depmod, devfsd, df, dhcprelay,
diff, dirname, dmesg, dnsd, dos2unix, dpkg, dpkg_deb, du,
dumpkmap, dumpleases, e2fsck, echo, ed, egrep, eject, env,
envdir, envuidgid, ether_wake, expand, expr, fakeidentd, false,
fbset, fbsplash, fdflush, fdformat, fdisk, fetchmail, fgrep,
find, findfs, fold, free, freeramdisk, fsck, fsck_minix, ftpget,
ftpput, fuser, getenforce, getopt, getsebool, getty, grep,
gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid, hostname,
httpd, hush, hwclock, id, ifconfig, ifdown, ifenslave, ifup,
inetd, init, inotifyd, insmod, install, ip, ipaddr, ipcalc,
ipcrm, ipcs, iplink, iproute, iprule, iptunnel, kbd_mode, kill,
```

```
killall, killall5, klogd, lash, last, length, less, linux32,
linux64, linuxrc, ln, load_policy, loadfont, loadkmap, logger,
login, logname, logread, losetup, lpd, lpq, lpr, ls, lsattr,
lsmod, lzmacat, makedevs, man, matchpathcon, md5sum, mdev, msg,
microcom, mkdir, mke2fs, mkfifo, mkfs_minix, mknod, mkswap,
mktemp, modprobe, more, mount, mountpoint, msh, mt, mv, nameif,
nc, netstat, nice, nmeter, nohup, nslookup, od, openvt, parse,
passwd, patch, pgrep, pidof, ping, ping6, pipe_progress,
pivot_root, pkill, poweroff, printenv, printf, ps, pscan, pwd,
raidautorun, rdate, rdev, readahead, readlink, readprofile,
realpath, reboot, renice, reset, resize, restorecon, rm, rmdir,
rmmmod, route, rpm, rpm2cpio, rtcwake, run_parts, runcon,
runlevel, runsv, runsvdir, rx, script, sed, selinuxenabled,
sendmail, seq, sestatus, setarch, setconsole, setenforce,
setfiles, setfont, setkeycodes, setlogcons, setsebool, setsid,
setuidgid, sh, shasum, showkey, slattach, sleep, softlimit,
sort, split, start_stop_daemon, stat, strings, stty, su, sulogin,
sum, sv, svlogd, swapoff, swapon, switch_root, sync, sysctl,
syslogd, tac, tail, tar, taskset, tcpsvd, tee, telnet, telnetd,
test, tftp, tftpd, time, top, touch, tr, traceroute, true, tty,
ttysize, tune2fs, udhcpc, udhcpd, udpsvd, umount, uname,
uncompress, unexpand, uniq, unix2dos, unlzma, unzip, uptime,
usleep, uudecode, uuencode, vconfig, vi, vlock, watch, watchdog,
wc, wget, which, who, whoami, xargs, yes, zcat, zcip
```

- man : Displays the System Manual

- ✓ \$ man passwd
- ✓ \$ man 5 passwd

Section	Finding manual on
1	User Commands
2	System Calls
3	Subroutines
4	Devices
5	File Formats
6	Games
7	Miscellaneous
8	System Administration
9	Kernel
10	New

## ■ file related command

- ✓ 파일 생성
  - vi, gcc, mknod, ...
- ✓ 파일 복사/이동
  - cp, mv, ln, ...
- ✓ 파일 삭제
  - rm
- ✓ 파일 이름 보기
  - ls
- ✓ 파일 내용 보기
  - more, cat, head, tail, objdump, hexdump
- ✓ 파일 속성 제어
  - chmod, chown, chgrp, touch
- ✓ 파일 redirection
  - >

```
root@localhost:~/sp/cmd — ssh — 80x24
[root@localhost cmd]# ls
test.txt
[root@localhost cmd]# cat test.txt
abcdef
[root@localhost cmd]# cp test.txt test_new.txt
[root@localhost cmd]# ls
test_new.txt test.txt
[root@localhost cmd]# more test.txt
abcdef
[root@localhost cmd]# more test_new.txt
abcdef
[root@localhost cmd]# rm test_new.txt
rm: remove regular file `test_new.txt'? y
[root@localhost cmd]#
[root@localhost cmd]# man rm
```

```
root@localhost:~/sp/cmd — ssh — 80x24
RM(1) User Commands RM(1)
NAME
  rm - remove files or directories
SYNOPSIS
  rm [OPTION]... FILE...
DESCRIPTION
  This manual page documents the GNU version of rm.  rm removes each
  specified file.  By default, it does not remove directories.
  If the -I or --interactive=once option is given, and there are more
  than three files or the -f, -R, or --recursive are given, then rm
  prompts the user for whether to proceed with the entire operation.  If
  the response is not affirmative, the entire command is aborted.
  Otherwise, if a file is unwritable, standard input is a terminal, and
  the -f or --force option is not given, or the -i or --interac-
  tive=always option is given, rm prompts the user for whether to remove
  the file.  If the response is not affirmative, the file is skipped.
OPTIONS
  :
```



## ■ 디렉터리 (directory)

- ✓ a set of files
- ✓ 계층 구조를 제공
- ✓ home directory, root directory, current directory
- ✓ relative path, absolute path

## ■ directory related command

- ✓ 생성
  - mkdir
- ✓ 이동
  - cd
- ✓ 삭제
  - rmdir
- ✓ 현재 위치
  - pwd

```
root@localhost:~/sp/cmd/programs — ssh — 80x24
[root@localhost cmd]# ls
test.txt
[root@localhost cmd]# pwd
/root/sp/cmd
[root@localhost cmd]# mkdir programs
[root@localhost cmd]# mkdir test
[root@localhost cmd]# mkdir reports
[root@localhost cmd]# ls
programs reports test test.txt
[root@localhost cmd]# cd programs/
[root@localhost programs]#
[root@localhost programs]# ls
[root@localhost programs]#
[root@localhost programs]# ls ../
programs reports test test.txt
[root@localhost programs]#
[root@localhost programs]# ls -a
. . .
[root@localhost programs]# cp ../test.txt ./test_new.txt
[root@localhost programs]#
[root@localhost programs]#
```

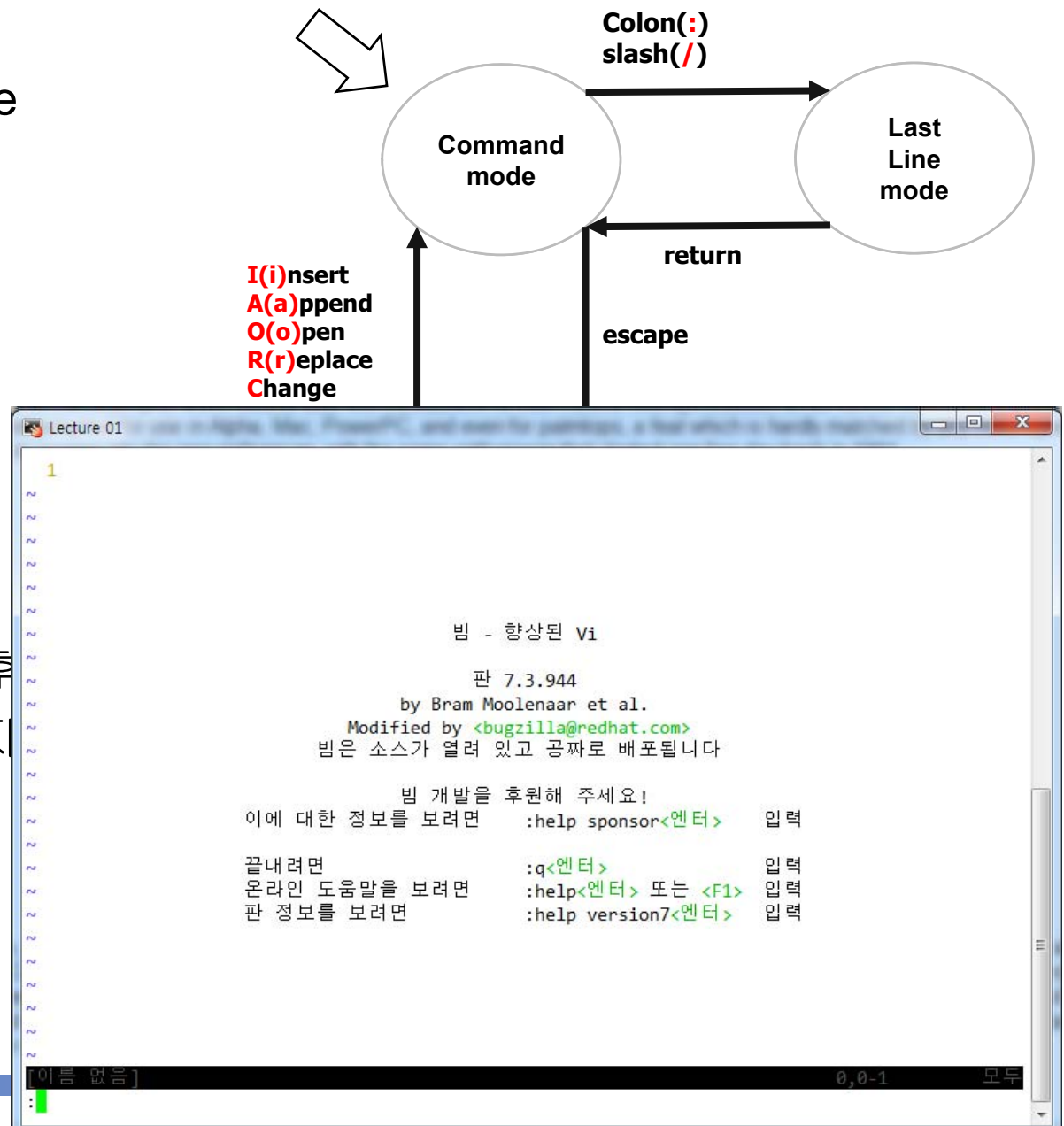


## ■ 파일 속성 제어

- ✓ 파일마다 permission과 owner 존재

```
root@localhost:~/sp/cmd/programs — ssh — 80x26
[root@localhost programs]# ls -l
total 8
-rw-r--r-- 1 root root 78 Sep 21 21:29 hello.c
-rw-r--r-- 1 root root 7 Sep 21 21:27 test_new.txt
[root@localhost programs]#
[root@localhost programs]# gcc -o hello hello.c
[root@localhost programs]#
[root@localhost programs]# ls -l
total 16
-rwxr-xr-x 1 root root 6425 Sep 21 21:30 hello
-rw-r--r-- 1 root root 78 Sep 21 21:29 hello.c
-rw-r--r-- 1 root root 7 Sep 21 21:27 test_new.txt
[root@localhost programs]# ./hello
Hello world~
[root@localhost programs]#
[root@localhost programs]# chmod -x hello
[root@localhost programs]# ./hello
-bash: ./hello: Permission denied
[root@localhost programs]# mkdir test
[root@localhost programs]# ls -l
total 20
-rw-r--r-- 1 root root 6425 Sep 21 21:30 hello
-rw-r--r-- 1 root root 78 Sep 21 21:29 hello.c
drwxr-xr-x 2 root root 4096 Sep 21 21:30 test
-rw-r--r-- 1 root root 7 Sep 21 21:27 test_new.txt
[root@localhost programs]#
```

- 실행
  - ✓ \$ vi [option] filename
- vi 수행
  - ✓ 명령 모드
  - ✓ 입력 모드
  - ✓ last line 모드
- 종료
  - ✓ :wq, :x – 문서 저장 후
  - ✓ :q! – 문서를 저장하지



## ■ 커서 이동 명령

h, j, k, l	좌, 하, 상, 우	{	한 문단 위로 이동
(	현재 문장의 처음으로	}	한 문단 아래로 이동
)	현재 문장의 끝으로	^ or 0	행의 시작으로 이동
H	화면 맨 윗줄로 이동	\$	행의 끝으로 이동
M	화면 중간 줄로 이동	gg	문서의 시작으로 이동
L	화면 맨 아랫줄로 이동	G	문서의 끝으로 이동
w	다음 단어로 이동	:n	n 행으로 이동
b	이전 단어로 이동	nG	n 행으로 이동
CTRL+f	한화면 아래로	z enter	현재 줄을 화면의 처음으로
CTRL+d	반화면 아래로	Nz enter	N 번째 줄을 화면의 처음으로
CTRL+b	한화면 위로	z.	커서 line을 화면 중앙으로
CTRL+u	반화면 위로	Nz.	N 번째 줄을 화면 중앙으로
CTRL+e	화면 한줄 아래로	z-	현재 줄을 화면 맨 아래로
CTRL+y	화면 한줄 위로	Nz-	N 번째 줄을 화면 맨 아래로

## ■ 입력 모드 전환 명령

i	커서 위치부터 입력	o	커서의 다음 줄에 입력
I	커서 행 맨 앞부터 입력	O	커서의 이전 줄에 입력
a	커서 위치 다음부터 입력	s	커서 위치 한 글자 지우고 입력
A	커서 행 맨 뒤부터 입력	S	커서 위치 한 줄 지우고 입력
r	현재 커서 위치 문자를 다른 문자로 변경	C	현재 커서 위치에서 뒤쪽 줄을 삭제하고 입력
R	현재 커서 위치부터 <b>replace</b>		

## ■ 실행 취소

- ✓ **u**: 바로 전에 수행한 **vi** 명령 모드 명령어 취소
- ✓ **U**: 현재 줄에서 수행한 모든 **vi** 명령 모드 명령어 취소 줄을 변경하면 취소 불가

## ■ 복사 및 삭제

x or dl	커서 위치 한 글자 삭제	yy, Y	커서위치 줄을 버퍼로 복사
X or dh	커서 앞 글자 삭제	Nyy or NY	N 줄을 버퍼로 복사
nX	커서 앞 N개 글자 삭제	u	되돌리기
dw	커서 위치 단어 삭제	p	버퍼 내용을 커서 아래 줄에 추가
d\$ or D	커서 위치부터 행 끝까지 삭제	P	버퍼 내용을 커서 위 줄에 추가
d0	커서 위치 앞부터 행 처음까지 삭제	dj	커서가 있는 행과 그 다음 행 삭제
dd	커서가 있는 행 삭제	dk	커서가 있는 행과 그 이전 행 삭제

## ■ 저장 및 종료

:w	저장	:q	종료
:w!	강제 저장	:q!	강제 종료
:w filename	현재 파일 filename으로 저장	:e	현재 파일 불러옴
:w >> filename	filename에 덧붙여서 저장	:e filename	filename 파일 불러옴
:wq or ZZ or :x	저장 후 종료	:wq!	강제 저장 후 종료

- 그 밖의 유용한 기능
  - ✓ v : block 지정
  - ✓ ctrl + w n: 가로 창 분할
  - ✓ :%s/old/new/g : old문자열을 new문자열로 치환
    - :2,6s/old/new/g : 2행과 4행 사이의 old문자열을 new문자열로 치환
  - ✓ /: 문자열 검색
  - ✓ n은 다음 문자, N인 이전 문자



- 컴파일러의 구분
  - ✓ IDE (Integrated Development Environments): 통합 개발 환경
  - ✓ Command line Compiler
- 본 강의에서 사용할 컴파일러
  - ✓ GNU gcc,
  - ✓ 그 외 Microsoft Visual C++ (command line 명령은 cl), Turbo C/C++, Borland C/C++,

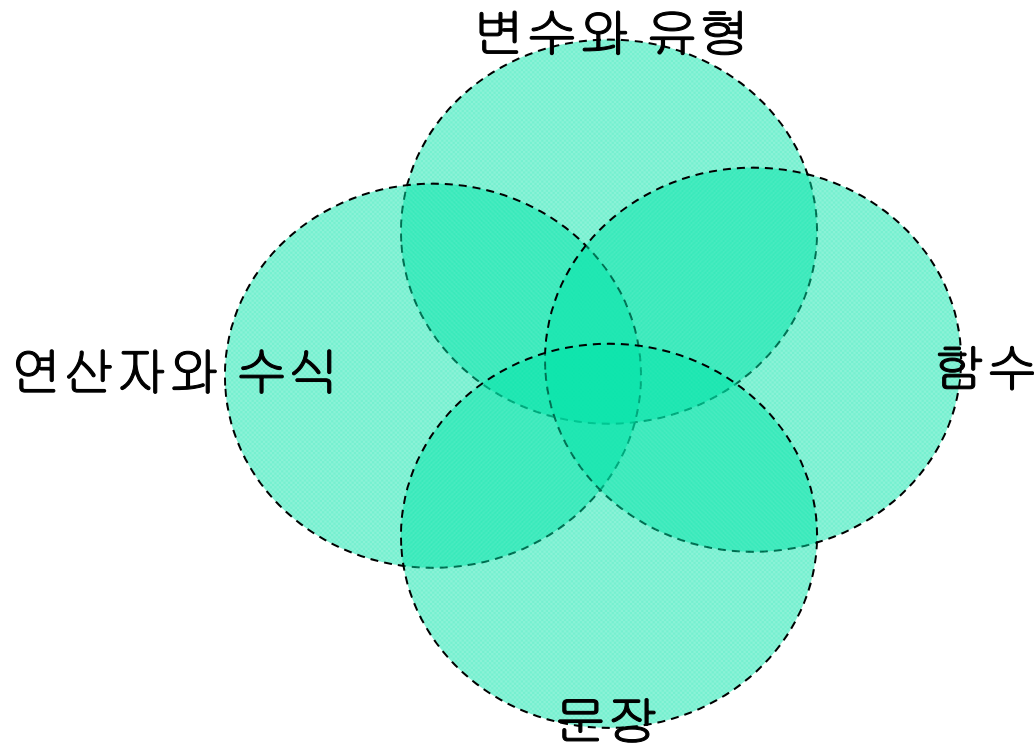
## ■ Overall

```
root@localhost:/home/Lecture/C — -ssh -2 root@root.ees.guru — 80×24
[[root@localhost C]# ls
[[root@localhost C]# vim hello.c
[[root@localhost C]#
[[root@localhost C]# more hello.c
#include <stdio.h>

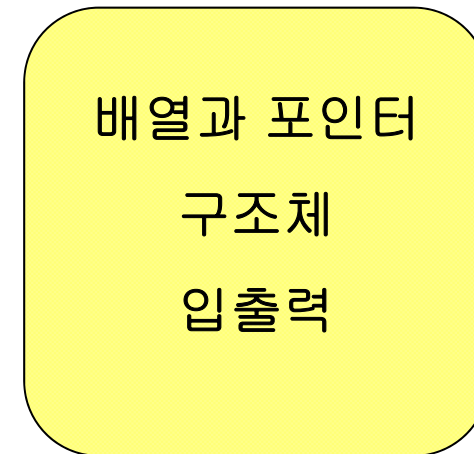
int main(void)
{
    printf("Hello world~\n");
    return 0;
}
[[root@localhost C]# ls -l
total 4
-rw-r--r--. 1 root root 77 Mar 13 14:11 hello.c
[[root@localhost C]# gcc -o myhello hello.c
[[root@localhost C]# ls -lh
total 16K
-rw-r--r--. 1 root root 77 Mar 13 14:11 hello.c
-rwxr-xr-x. 1 root root 8.4K Mar 13 14:12 myhello
[[root@localhost C]# ./myhello
Hello world~
[[root@localhost C]# █
```

- C 프로그램의 특징
  - ✓ System Software 개발 도중 프로그래머에 의해 만들어짐
  - ✓ High-level과 Low-level 언어의 장점 포함 (middle-level lang.)
  - ✓ Brevity (간결성)
  - ✓ Generality (범용성)
  - ✓ C++, Java의 기반
  - ✓ 알고리즘 기술 언어. 프로그래머간의 대화 도구

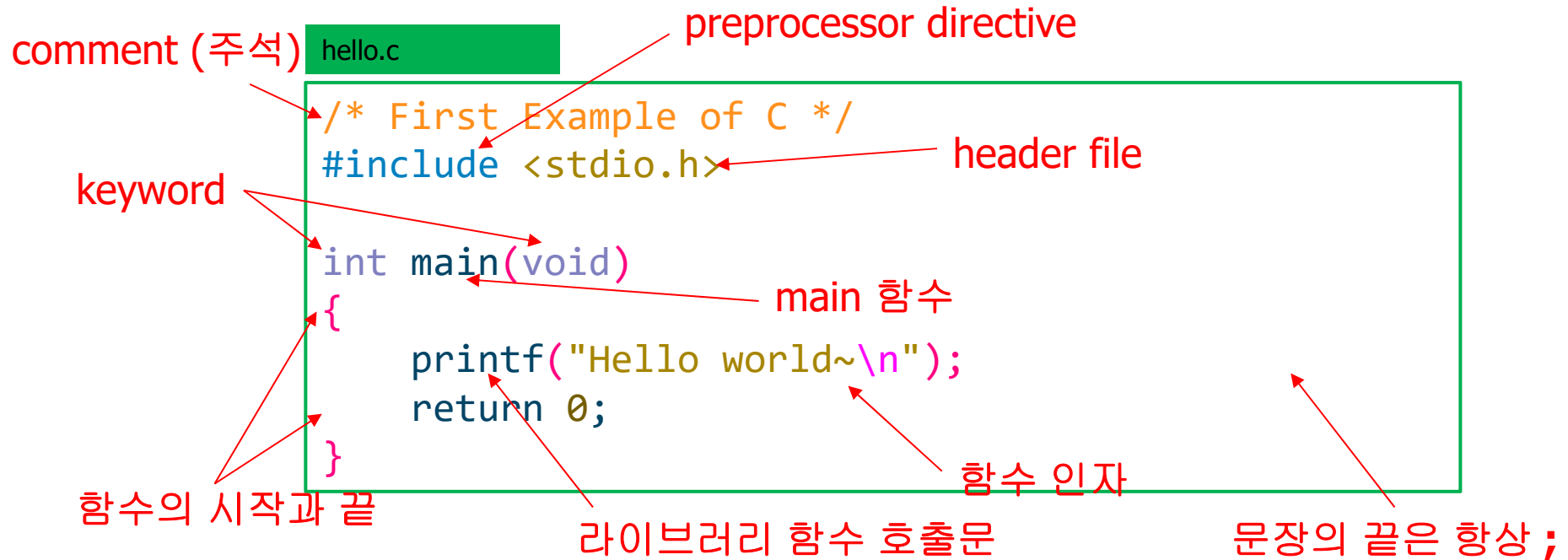
## 기본 요소



## 확장 요소



# C 프로그램 구성 요소(3/6)



☞ 선행처리 지시자 (Preprocessor Directive) 문장은 ;으로 끝나지 않는다.

## C 프로그램 구성 요소(4/6)

### ■ 프로그램

- ✓ 모든 C 프로그램은 함수들로 구성 (최소한 main 함수 하나는 포함)

### ■ 함수

- ✓ 함수는 문장 또는 연관된 문장의 집합(block)으로 구성
- ✓ {}
- ✓ 함수 이름, 반환 자료형, 매개 변수(인수, 인자)
- ✓ 함수 이름에는 알파벳, 숫자, 특수 문자로는 \_ 만 사용 가능, 숫자가 첫 문자일수는 없음 (실제 이 부분은 compiler dependent)
- ✓ caller, callee

```
/* 함수 프로토타입 */  
  
ret-type function-name(param-lists)  
{  
    statement sequence  
}
```

## ■ 문장

- ✓ 프로그램의 수행 동작(**operation**)을 의미
- ✓ 모든 문장은 ;(세미콜론)으로 끝남
- ✓ 문장은 수식과 연산자, 그리고 C 키워드로 구성
- ✓ 문장의 종류에는 제어문, 반복문, 치환문, 선택문, 선언문 등이 존재

## ■ 수식과 연산자

- ✓ 수식은 변수(또는 상수)와 연산자로 구성
- ✓ 연산자에는 산술, 논리, 관계, 치환, 비트 연산자 등이 존재

## ■ 변수

- ✓ 메모리의 특정 주소를 이름으로 접근 가능하도록 함
- ✓ 유형(**type**)을 갖는다.

## ■ 라이브러리 함수

- ✓ 다양한 부가 기능을 제공하는 함수 집합
  - 입출력
  - 메모리 할당
  - 스트링(문자열) 조작
  - 수학 함수
  - 통신

## ■ C는 전처리기 사용

- ✓ 선행처리 지시자 (preprocessor directive)
  - #include
  - #define
  - 조건 컴파일
  - 인라인 처리

☞ 위 두 가지의 사용이 C를 매우 간결하면서 매력적인 언어로 만들었다



## 프로그램에 주석 달기

- 주석: /\* \*/, 일부 컴파일러에서는 //도 지원
  - 주석의 중요성
    - ✓ 여러 개발자가 각자가 맡은 부분을 프로그램 한 이후 통합 하였다. 다음 문제가 발생 했을 때 이 중 가장 벌금을 많이 내야 할 개발자는?
      - 컴파일 시에 오류를 야기하는 프로그램 개발자
      - 수행 중에 버그를 야기시킨 개발자
      - 프로그램에 주석을 달지 않는 개발자 (프로그램은 잘 동작함)
      - 기한 내에 완료하지 못한 개발자
- ☞ 주석인 없는 프로그램 보고서는 제출하지 않은 것과 동일!!!  
주석에는 날짜, 작성자, 프로그램 명세(설계서 수준), 필요할 경우 각 문장별 주석 기술  
**Indentation**도 반드시 지킬 것

## ■ printf 사용 예

PrintfOne.c (p. 43)

```
#include <stdio.h>
int main(void)
{
    printf("Hello Everybody\n");
    printf("%d\n", 1234);
    printf("%d %d\n", 10, 20);
    return 0;
}
```

Results

```
Hello Everybody
1234
10 20
```

## ■ printf(...) 함수

- ✓ 첫 번째 인자 내용을 화면에 출력
- ✓ \n : 줄 바꿈 (escape sequence)
- ✓ %d : 부호가 있는 정수 출력 (서식 문자)

## ■ printf 사용 예

- ✓ printf 내에서 인자는 , 로 구분

```
printf(인자1, 인자2);
```

```
printf("%d\n", 1234);
```

```
printf("1234\n");
```

```
printf(인자1, 인자2, 인자3);
```

```
printf("%d %d\n", 10, 20);
```

```
printf("10 20\n");
```

Results

```
1234
10 20
```

- 컴퓨터 구조
- 리눅스 정의, 접근 방법 이해
- 파일, 프로세스 관련 명령어 이해
- 컴파일 관련 시스템 프로그램 이해
- C 언어의 기본 요소를 이해
  - ✓ 함수, 헤더 파일, 문장 구분
  - ✓ 표준 라이브러리
- printf 함수 이해
  - ✓ 기본 사용법

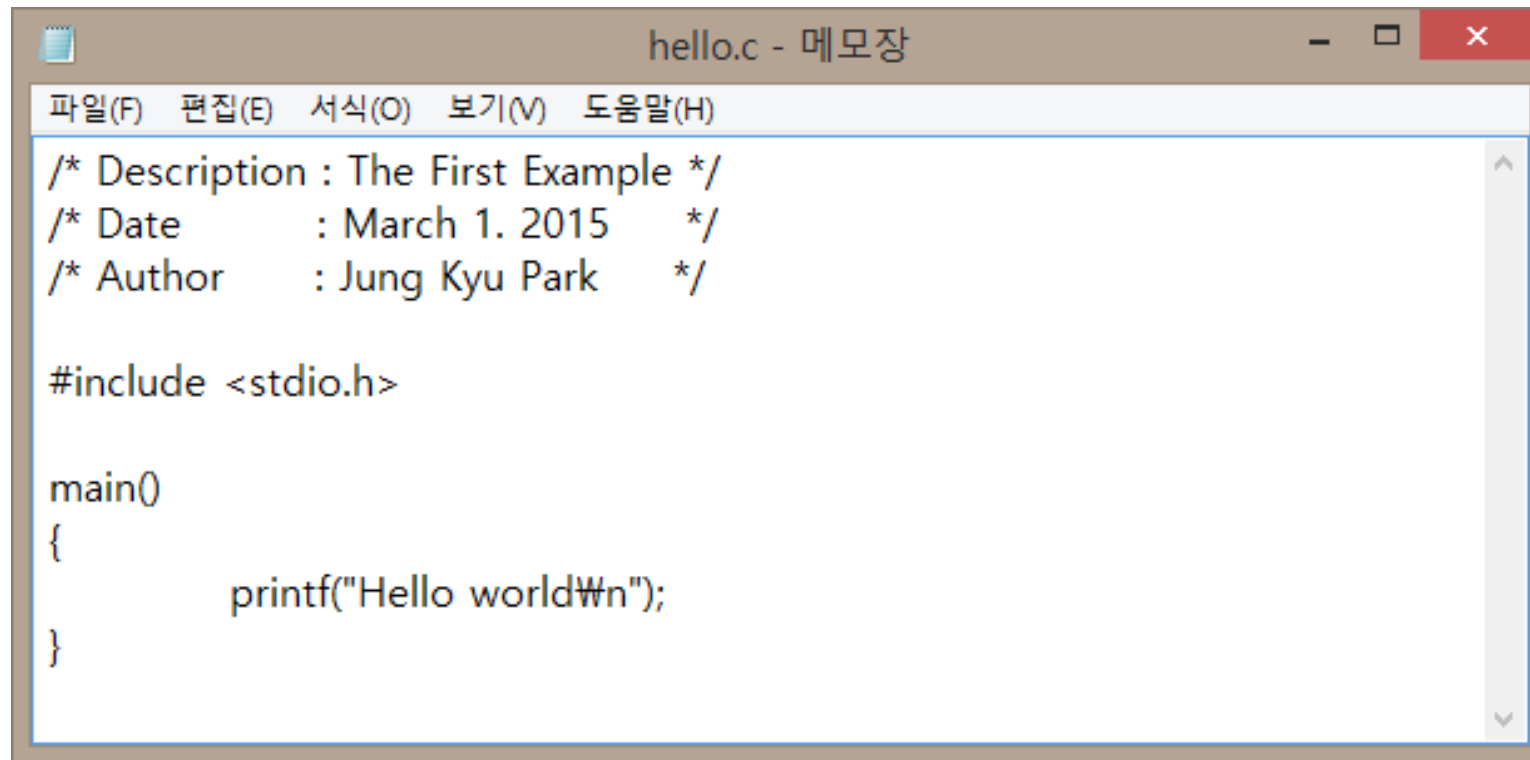
☞ 과제 #1.

- **printf** 함수와 **%d** 서식 문자를 사용하여 구구단 2단을 출력 하시오.
- 10개 이상 리눅스 명령어 사용하기 (**vim**과 **gcc** 포함).
- 보고서에는 반드시 본인 ID가 있어야 함 (eg. **whoami**, **date** 사용)

```

root@localhost:~/home/Lecture/C -- ssh -2 root@root.ees.guru -- 80x48
[root@localhost C]# vim report1.c
[root@localhost C]#
[root@localhost C]# gcc -o report1 report1.c
[root@localhost C]#
[root@localhost C]# ./report1
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
[root@localhost C]#
[root@localhost C]# whoami
root
[root@localhost C]# who am i
root pts/0 2016-03-13 14:09 (220.149.236.36)
[root@localhost C]#
[root@localhost C]# who
(unknown) :0 2016-02-18 16:57 (:0)
root pts/0 2016-03-13 14:09 (220.149.236.36)
[root@localhost C]#
[root@localhost C]# ls -l
total 32
-rw-r--r--. 1 root root 77 Mar 13 14:11 hello.c
-rwxr-xr-x. 1 root root 8503 Mar 13 14:12 myhello
-rwxr-xr-x. 1 root root 8507 Mar 13 14:25 report1
-rw-r--r--. 1 root root 379 Mar 13 14:24 report1.c
[root@localhost C]#
[root@localhost C]# cp report1 newreport
[root@localhost C]# ./newreport
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
[root@localhost C]#
    
```

## ■ 과정 1: 프로그램 작성



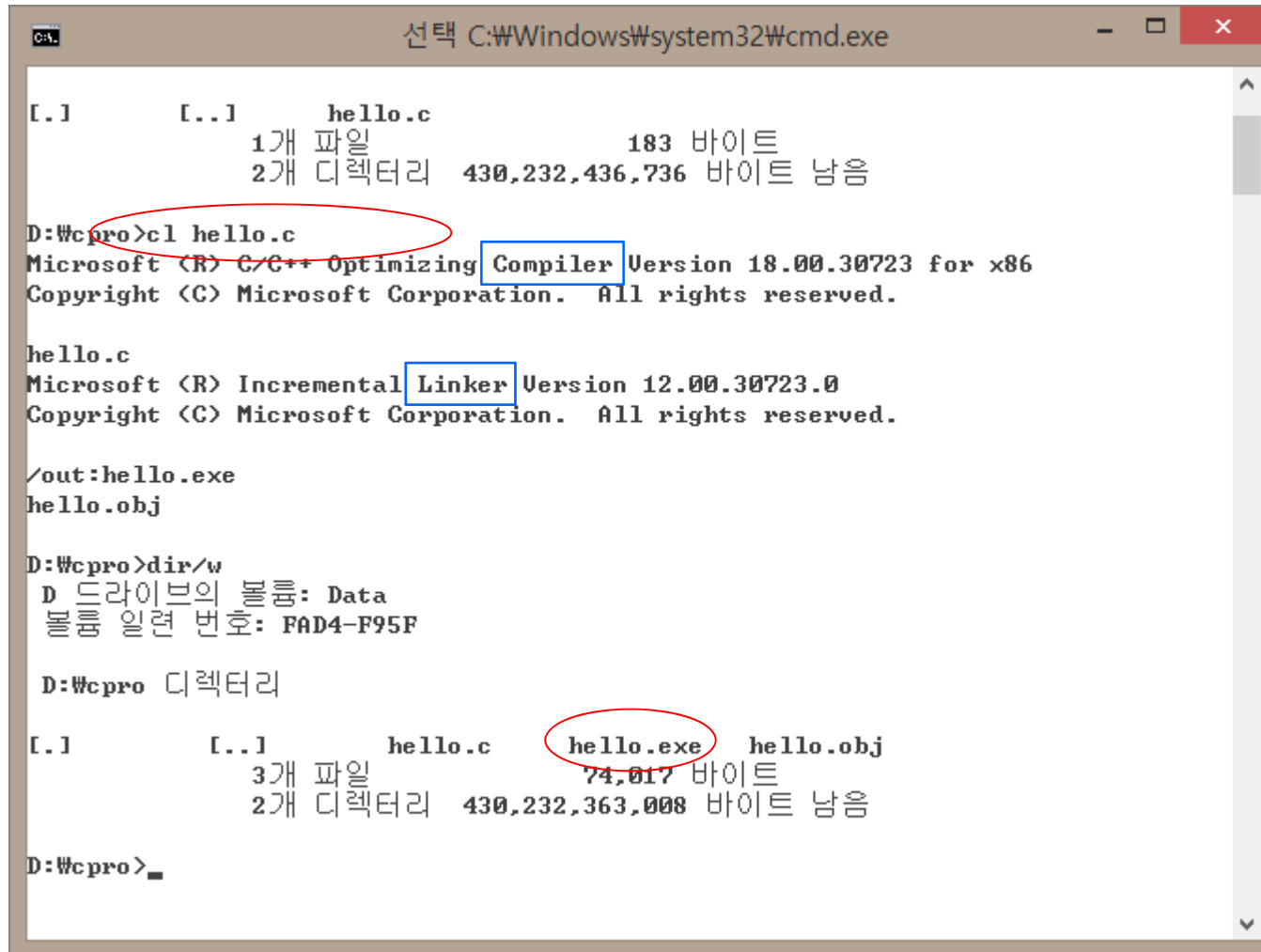
```
hello.c - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
/* Description : The First Example */
/* Date       : March 1, 2015   */
/* Author     : Jung Kyu Park   */

#include <stdio.h>

main()
{
    printf("Hello world\n");
}
```

## ■ 과정 2: 컴파일 (compile)

- ✓ cl (MS Visual C++), bcc (Borland C++), gcc (Linux), tc (Turbo C)



```
선택 C:\Windows\system32\cmd.exe

[.]      [...]      hello.c
          1개 파일          183 바이트
          2개 디렉터리 430,232,436,736 바이트 남음

D:\wcp>cl hello.c
Microsoft (R) C/C++ Optimizing Compiler Version 18.00.30723 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

hello.c
Microsoft (R) Incremental Linker Version 12.00.30723.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:hello.exe
hello.obj

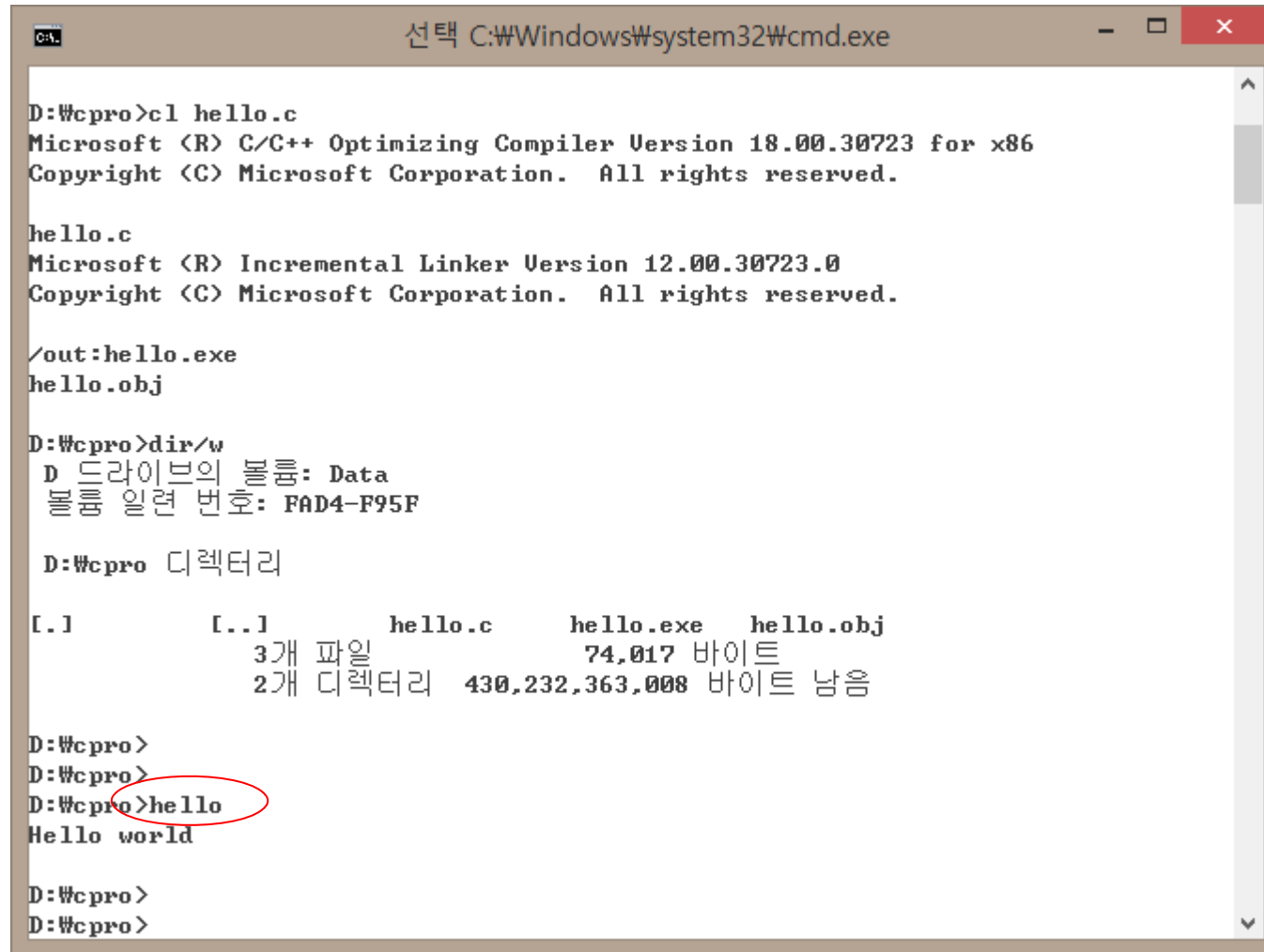
D:\wcp>dir/w
D 드라이브의 볼륨: Data
볼륨 일련 번호: FAD4-F95F

D:\wcp 디렉터리

[.]      [...]      hello.c      hello.exe      hello.obj
          3개 파일          74,017 바이트
          2개 디렉터리 430,232,363,008 바이트 남음

D:\wcp>
```

## ■ 과정 3: 수행



```
선택 C:\Windows\system32\cmd.exe

D:\wcprow>cl hello.c
Microsoft (R) C/C++ Optimizing Compiler Version 18.00.30723 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

hello.c
Microsoft (R) Incremental Linker Version 12.00.30723.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:hello.exe
hello.obj

D:\wcprow>dir/w
D 드라이브의 볼륨: Data
볼륨 일련 번호: FAD4-F95F

D:\wcprow 디렉터리

[.]          [...]      hello.c      hello.exe    hello.obj
              3개 파일          74,017 바이트
              2개 디렉터리 430,232,363,008 바이트 남음

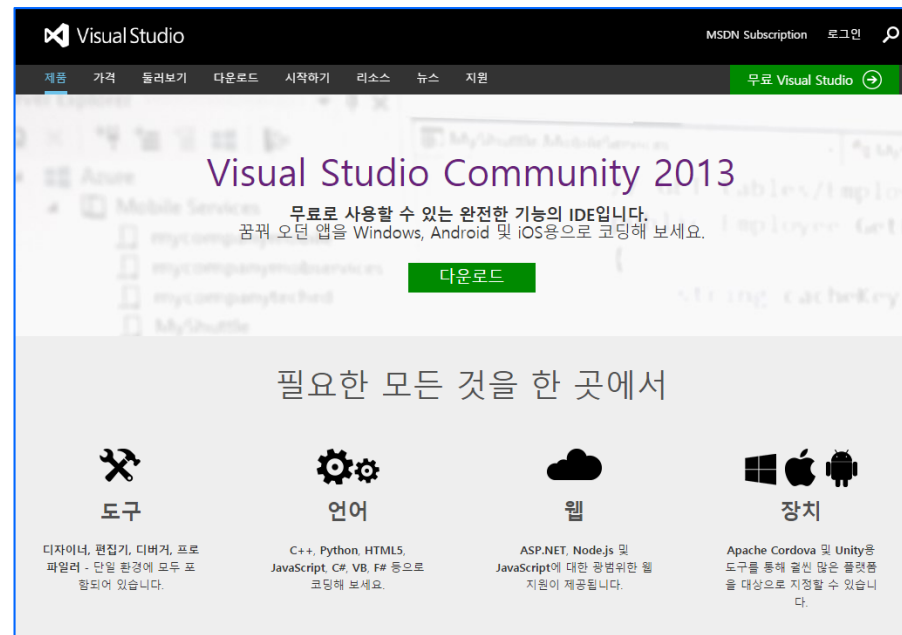
D:\wcprow>
D:\wcprow>
D:\wcprow>hello
Hello world

D:\wcprow>
D:\wcprow>
```

👉 **prompt> cl /help** 사용해 볼 것. [C:\Program Files \(x86\)\Microsoft Visual Studio 12.0\VC\bin](C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin)

## ■ What is Visual Studio?

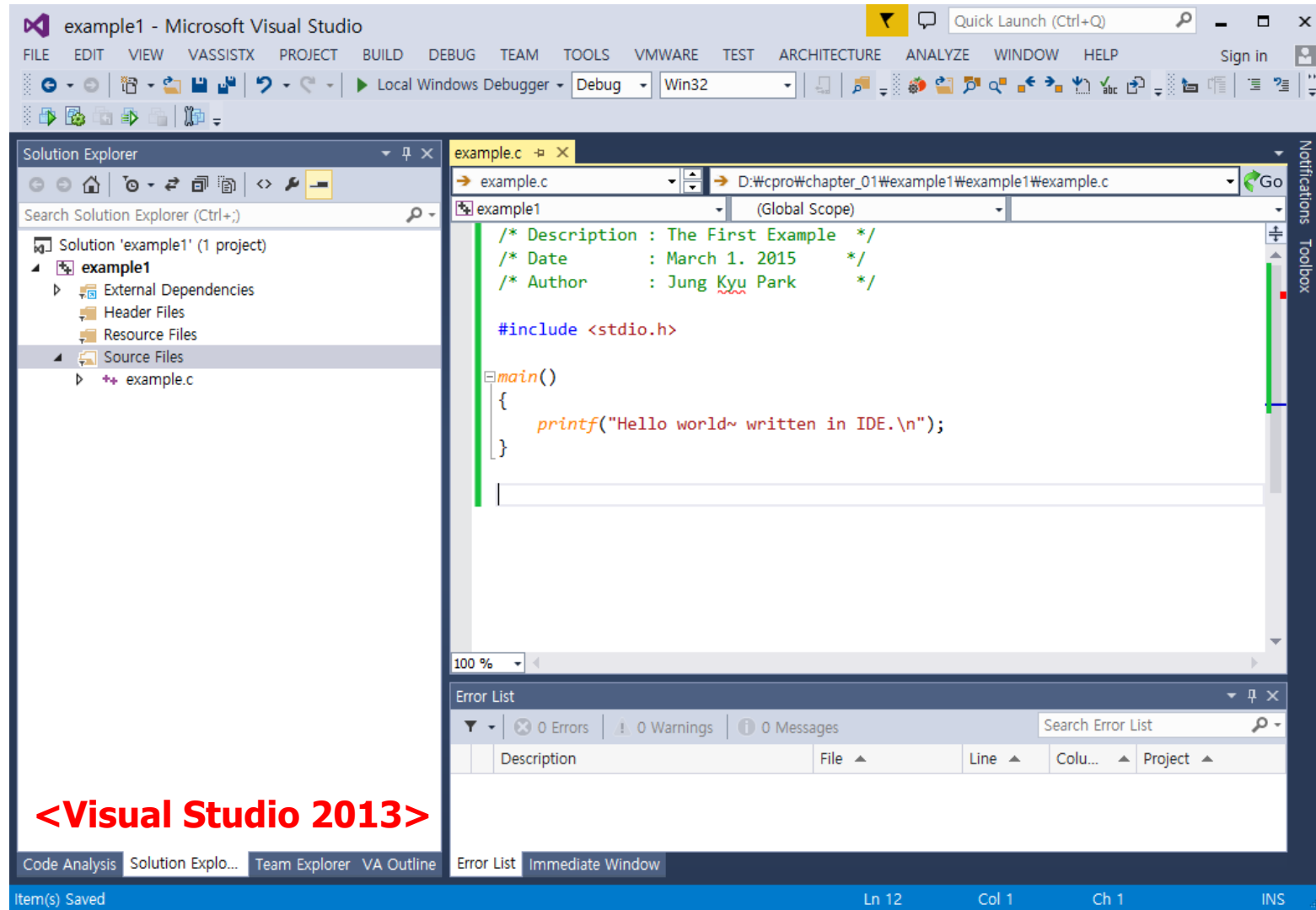
- ✓ An integrated development environment (IDE) from Microsoft
- ✓ Microsoft Windows, as well as web sites, web applications and web services
- ✓ C, C++ and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F#



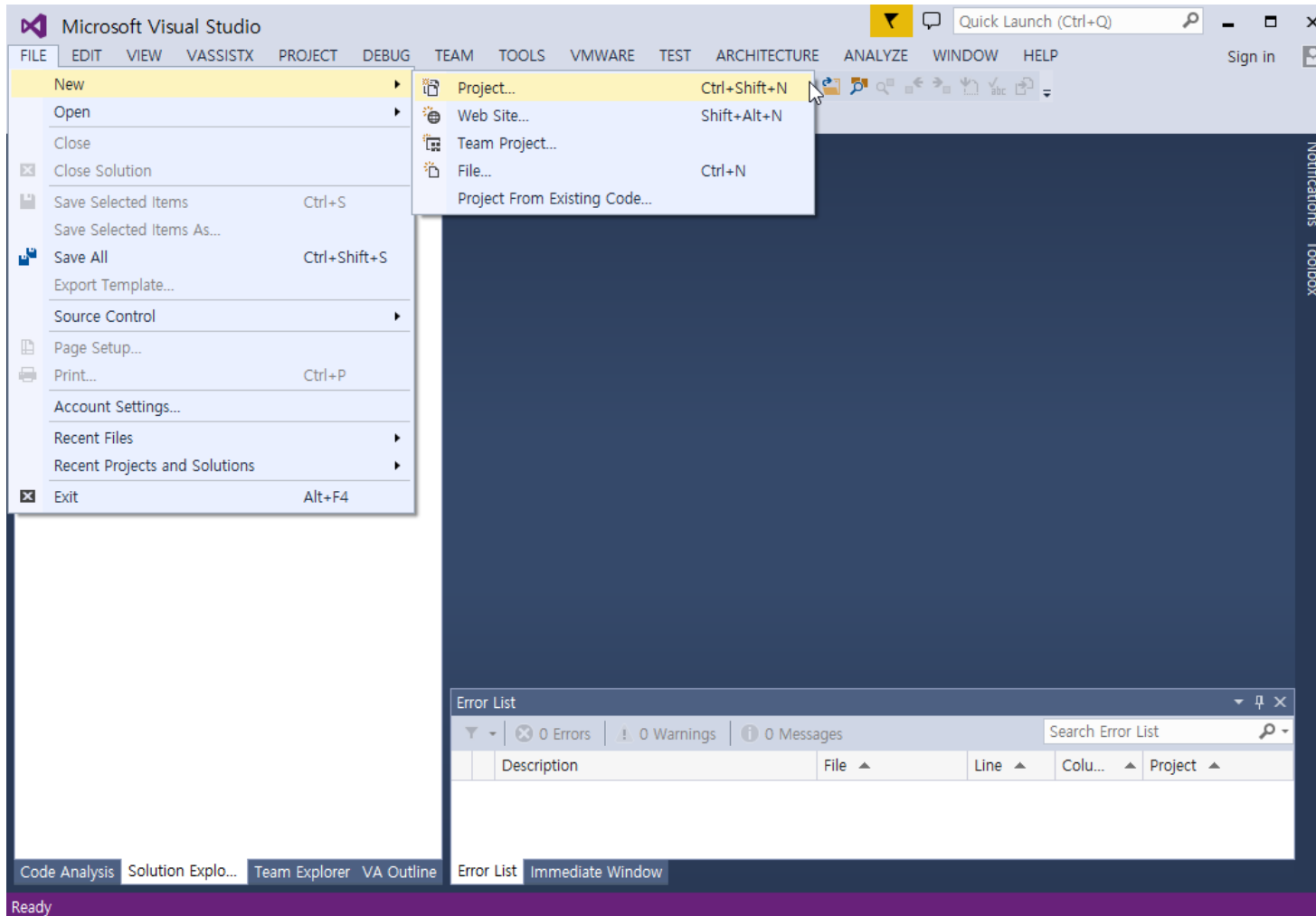
<<https://www.visualstudio.com/ko-kr/products/visual-studio-community-vs>>



## ■ IDE (Integrated Development Environment) 사용

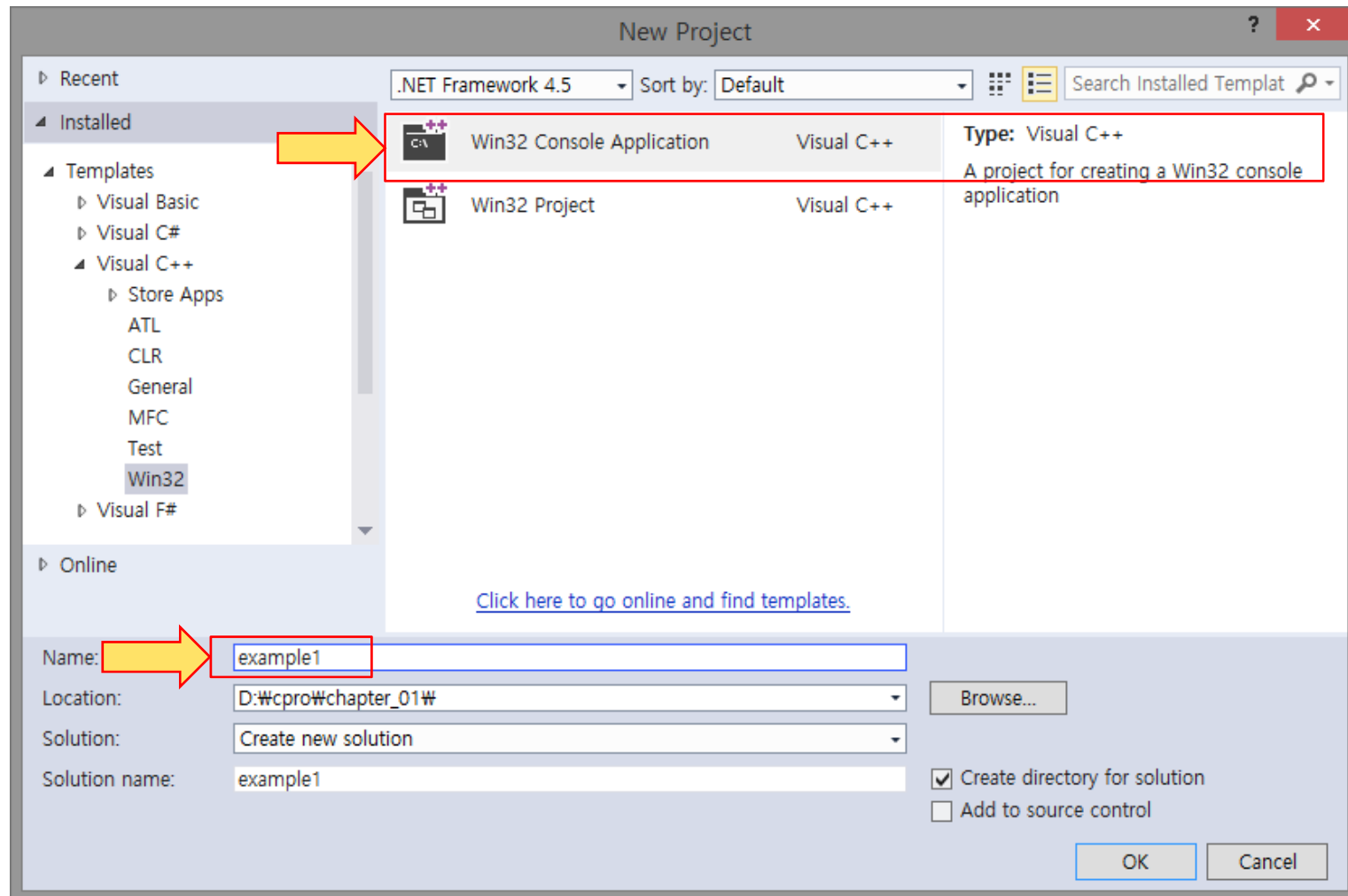


- IDE 사용 (project 생성)
  - ✓ New → Project...

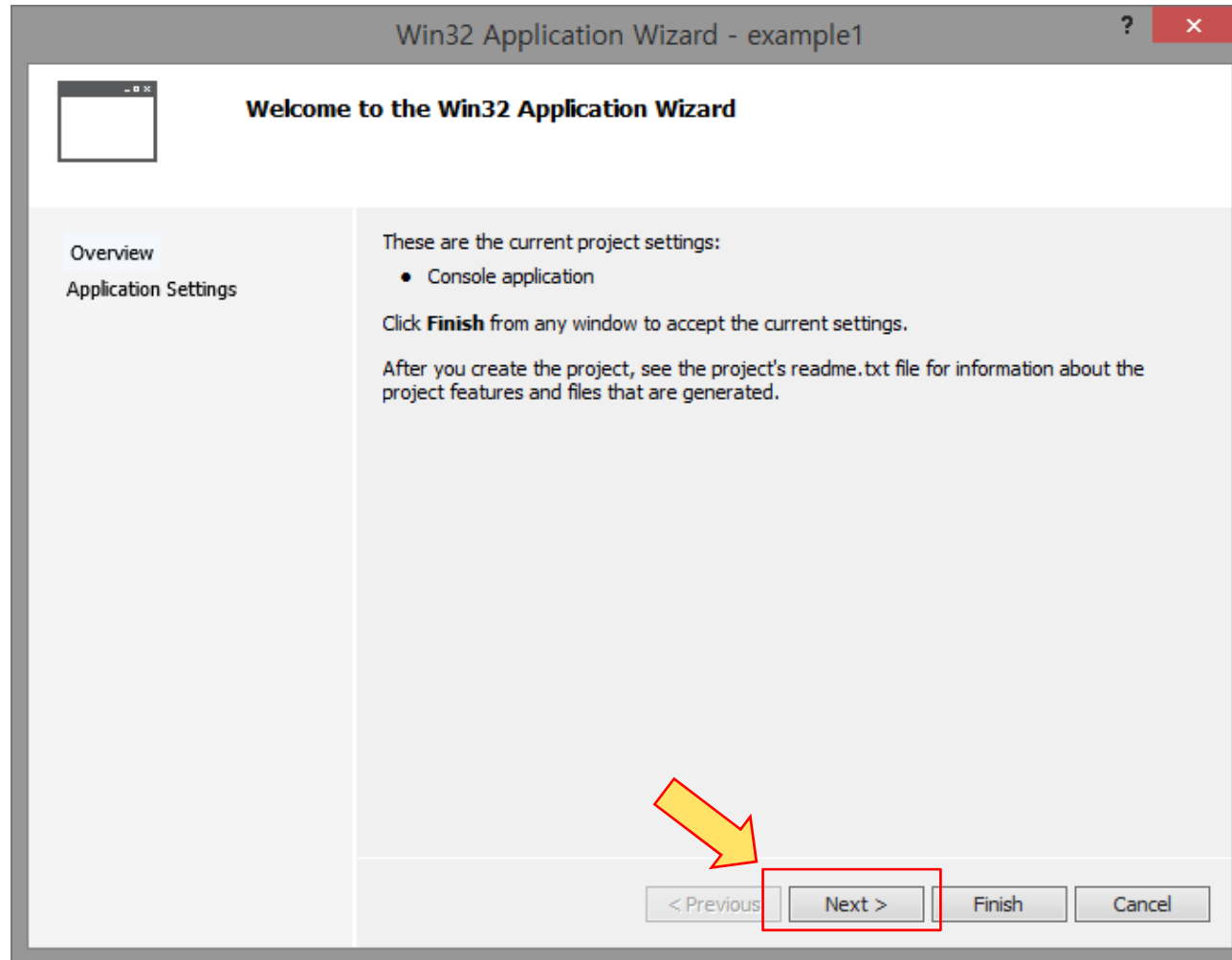


## ■ IDE 사용 (project 생성)

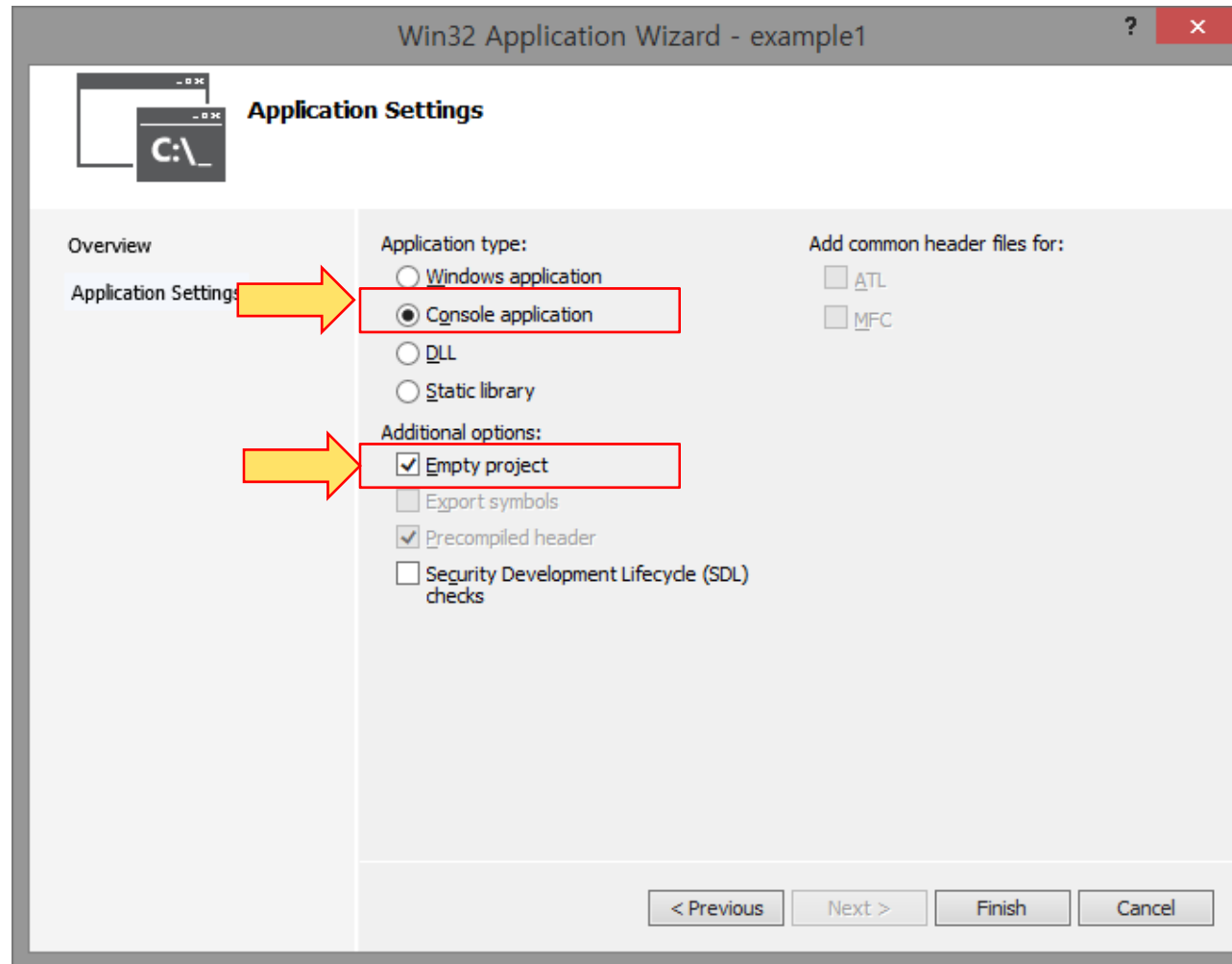
✓ Visual C++ → Win32 → Win32 Console Application



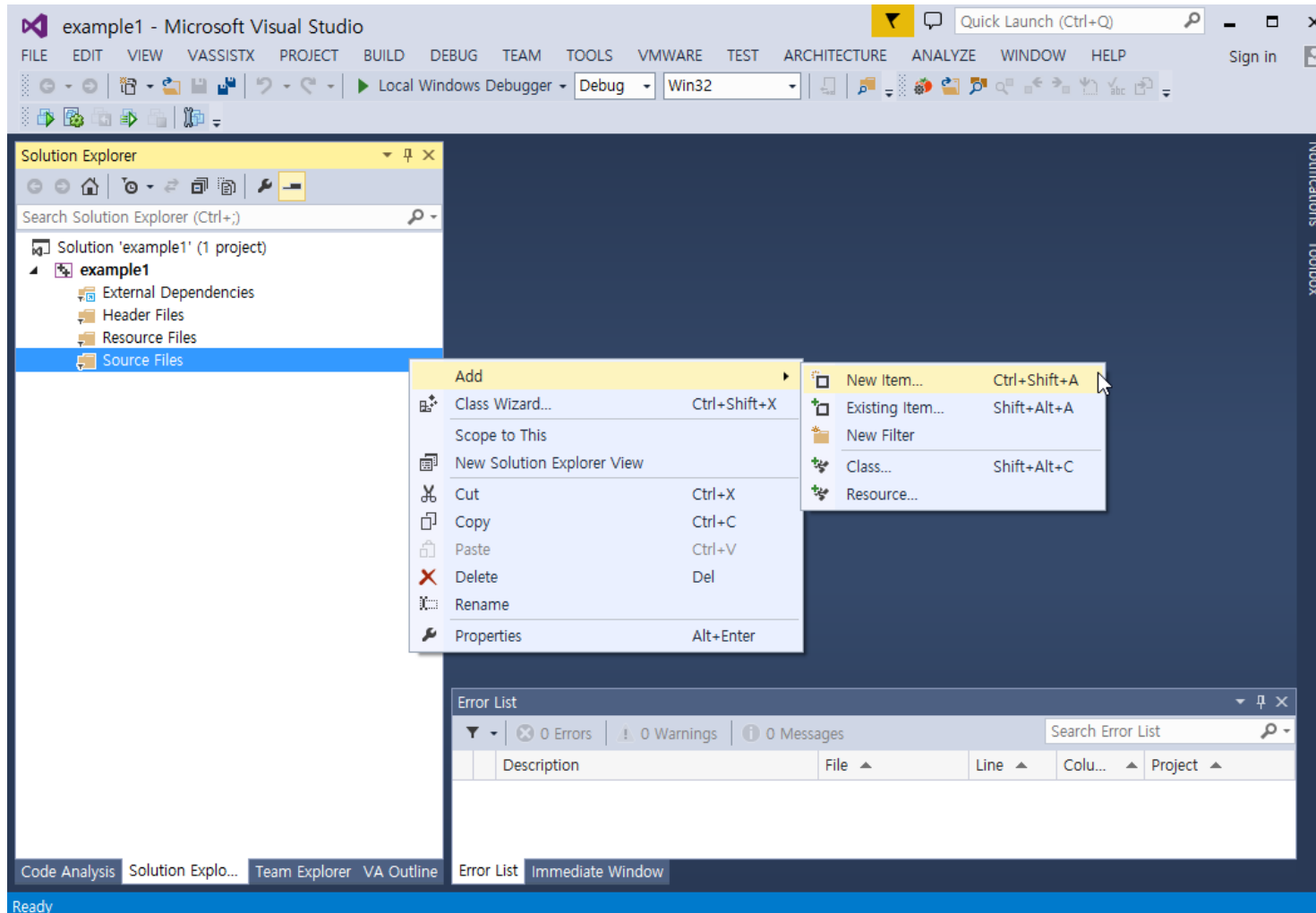
- IDE 사용 (project 생성)
  - ✓ Select Next



- IDE 사용 (project 생성)
  - ✓ Select Next

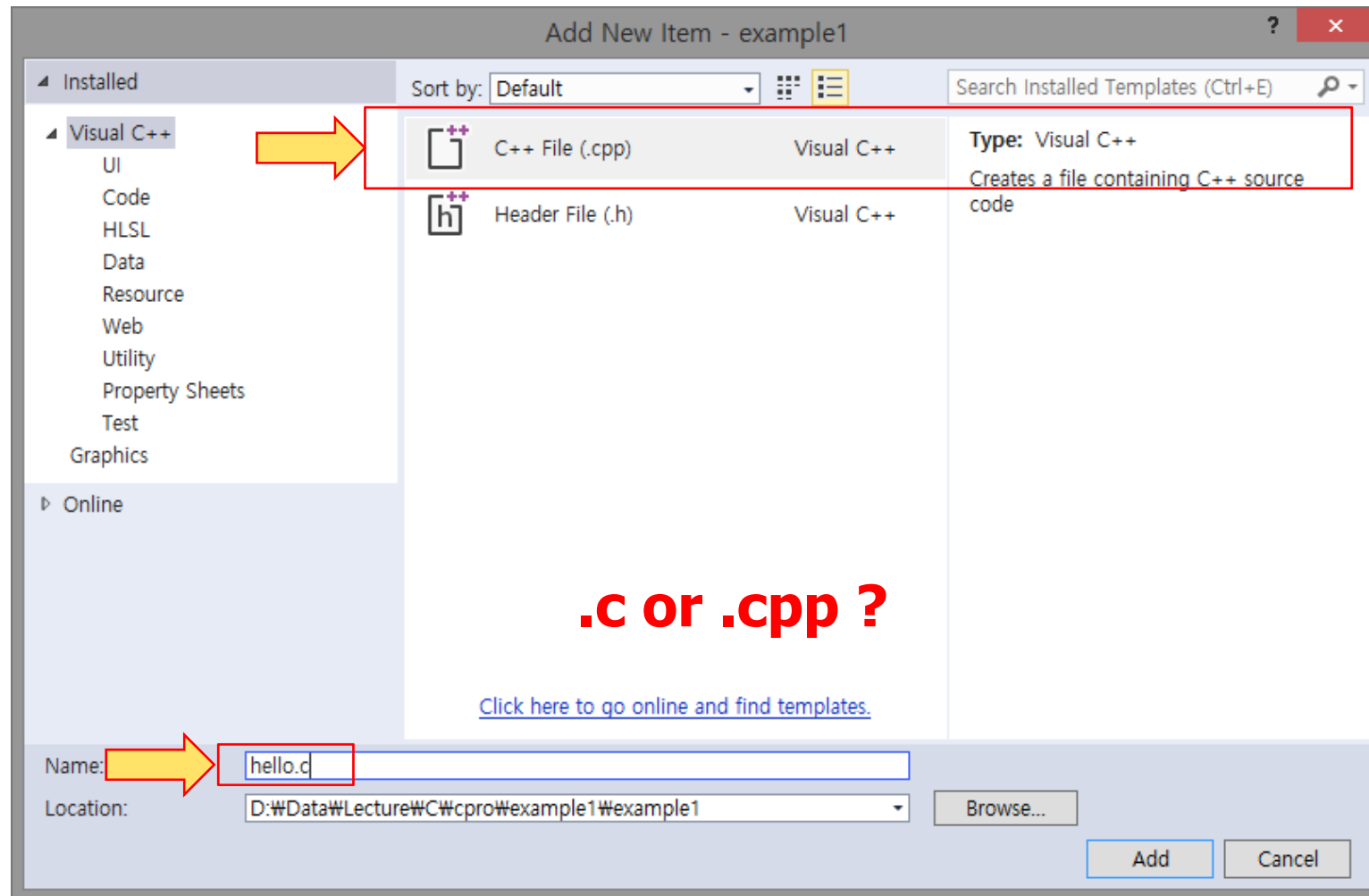


- IDE 사용 (project 생성)
  - ✓ Source Files → Add → New Item...

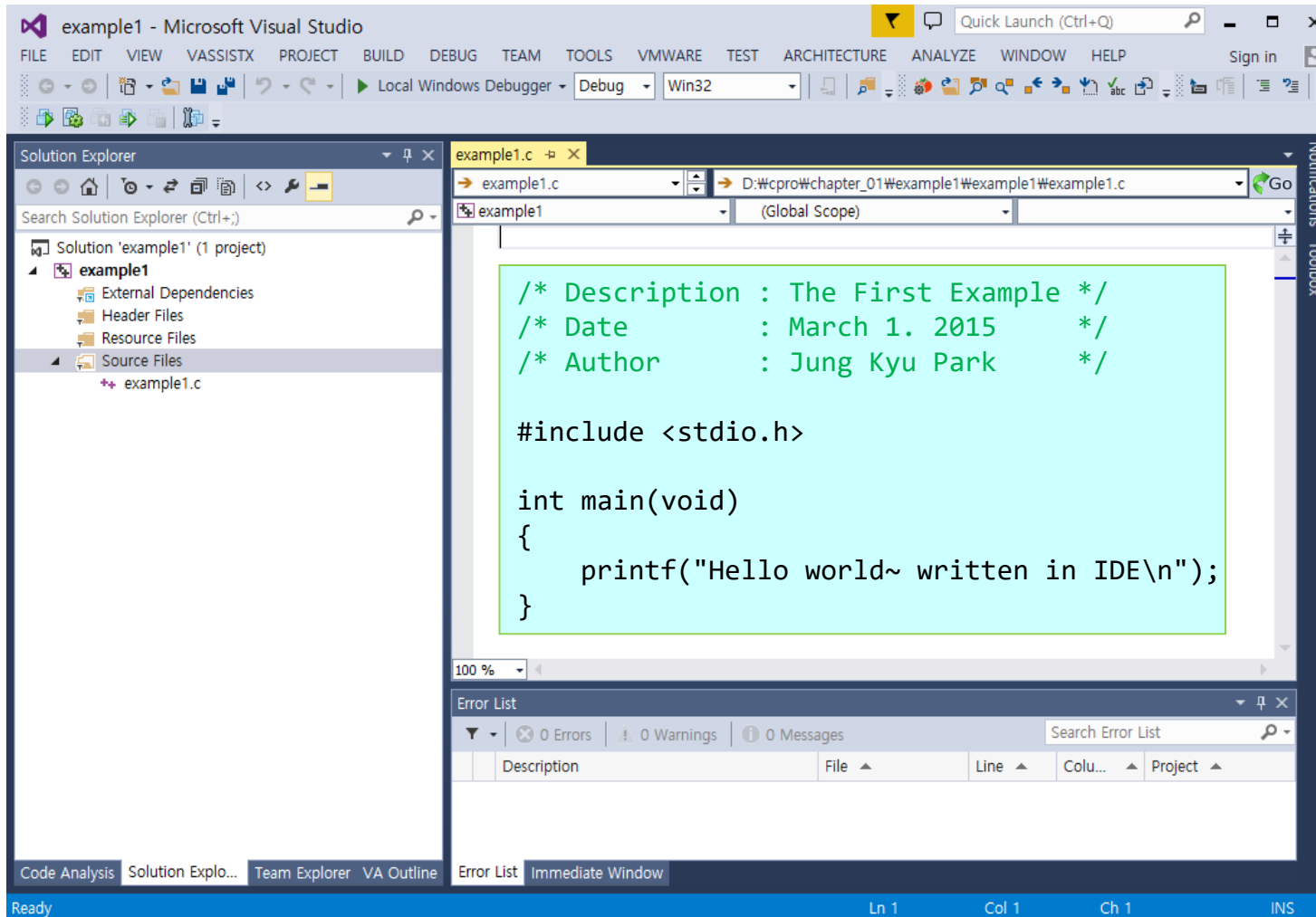


## ■ IDE 사용 (project 생성)

- ✓ Select C++ File (.cpp) → Enter *new file name* (.c)



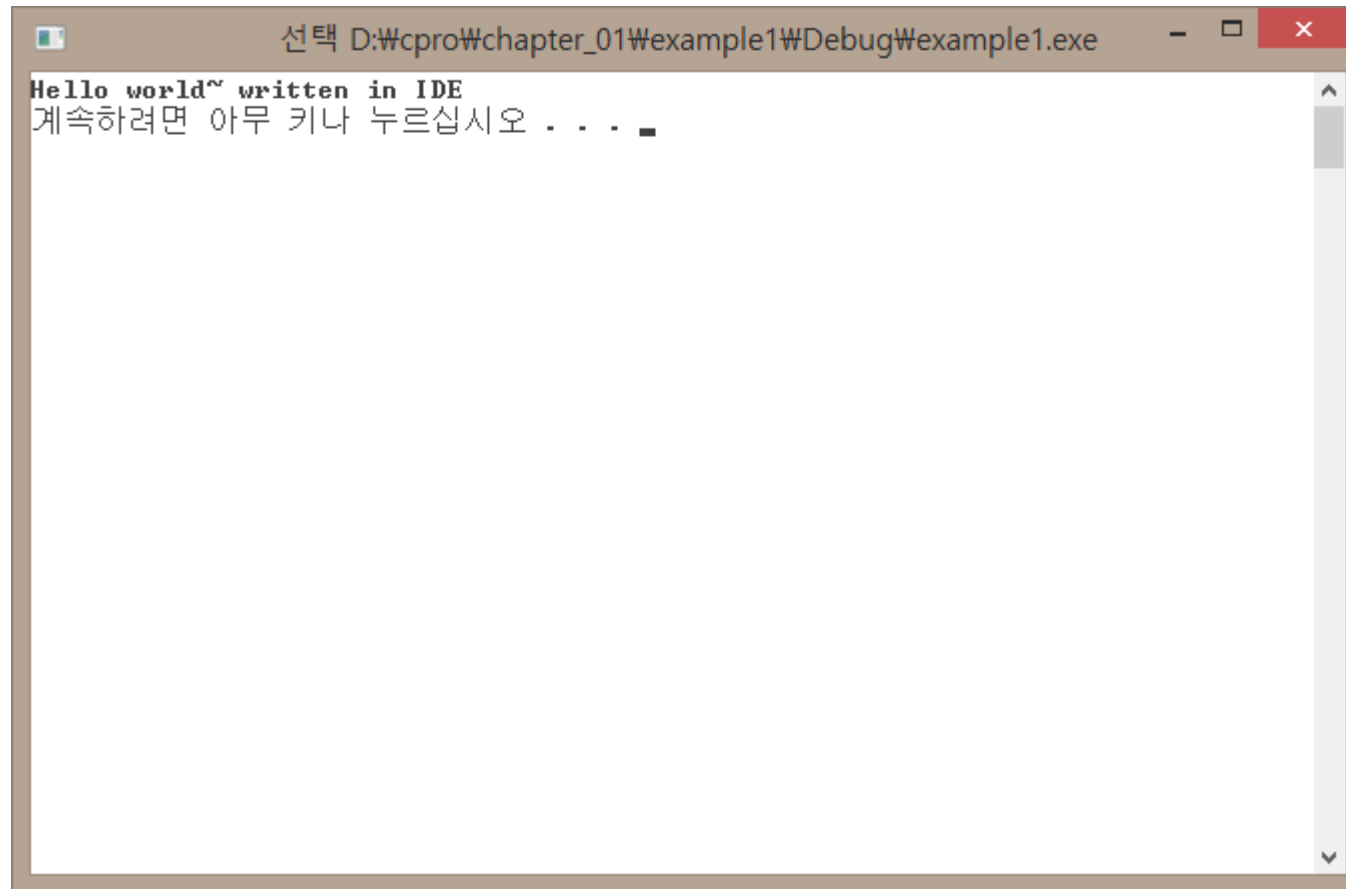
- IDE 사용 (project 생성)
  - ✓ Enter source code





## (부록) Windows 프로그램 컴파일 과정

- ✓ 프로젝트 컴파일 : F7 (파일 컴파일: Ctrl+F7)
- ✓ 수행 : Ctrl+F5 or 명령 프롬프트

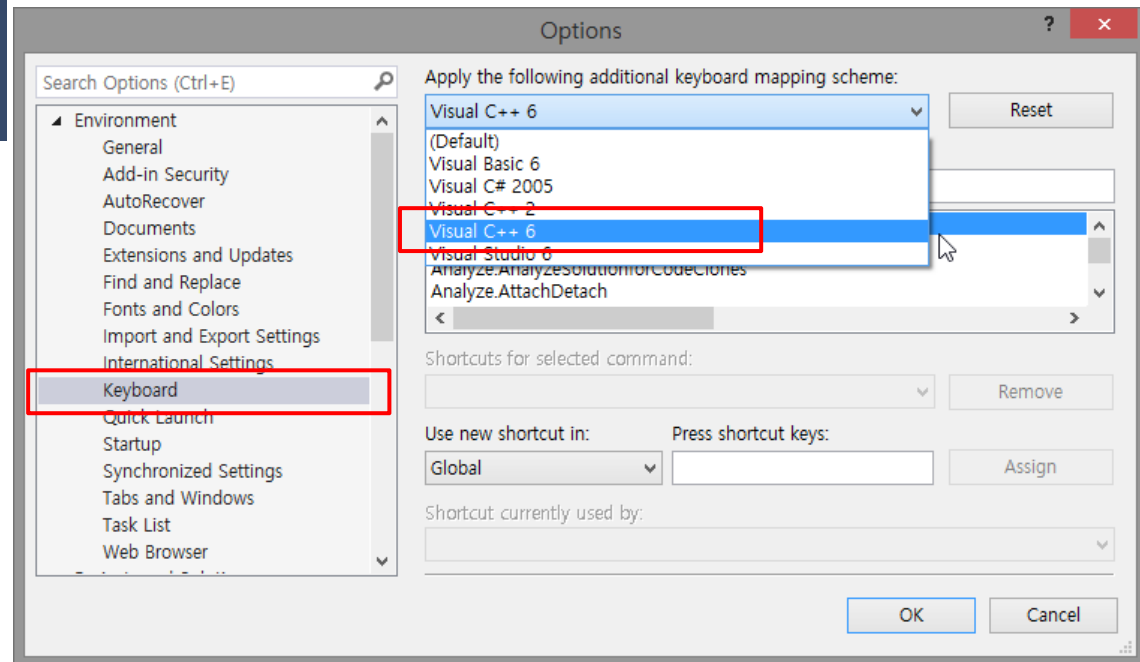
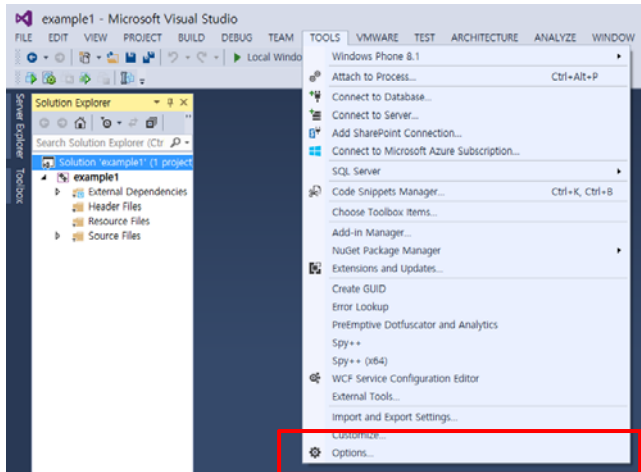


The screenshot shows a Windows command prompt window with the following text:

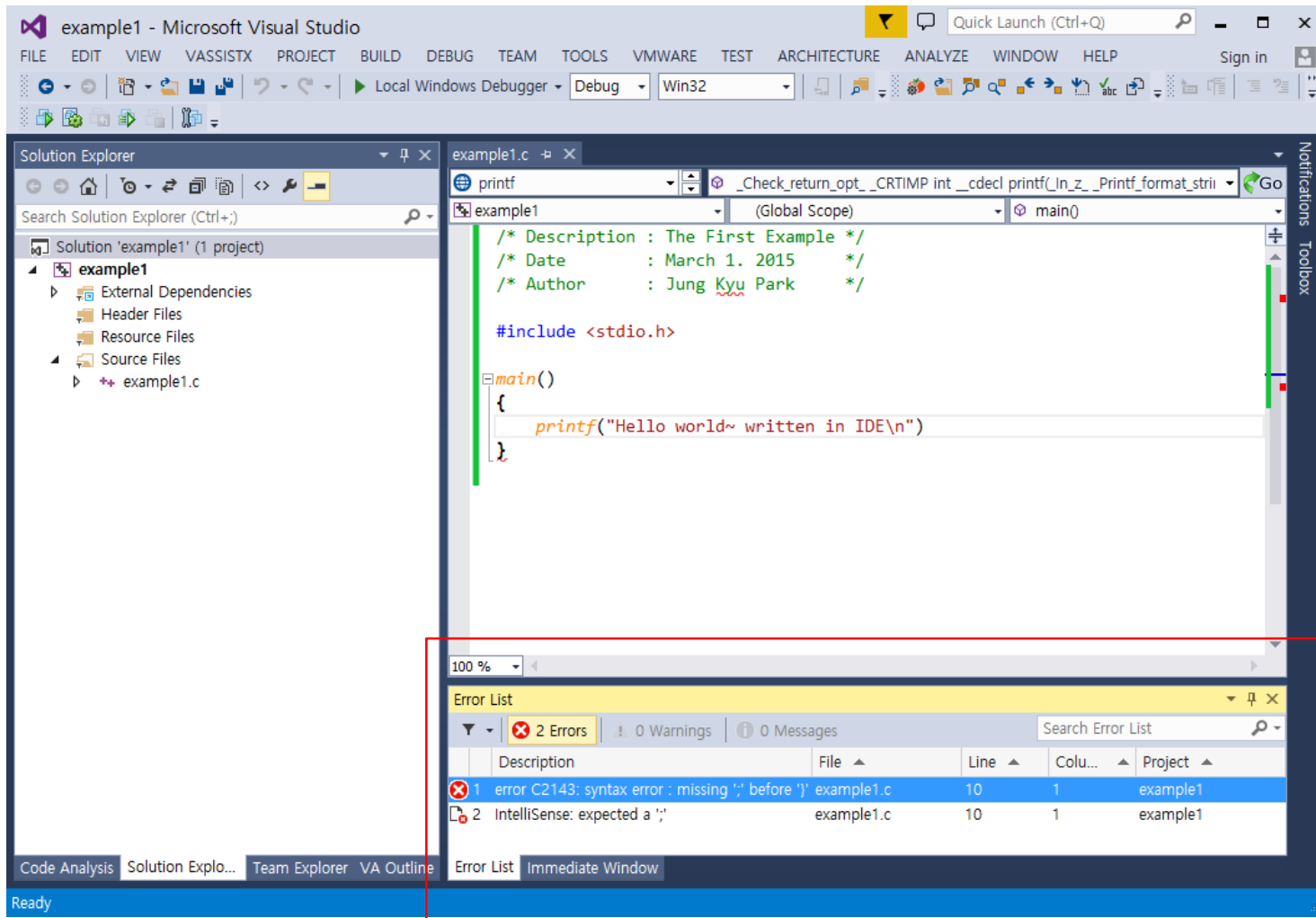
```
선택 D:\wcprow\chapter_01\example1\Debug\example1.exe
Hello world~ written in IDE
계속하려면 아무 키나 누르십시오 . . .
```

## (부록) Windows 프로그램 컴파일 과정

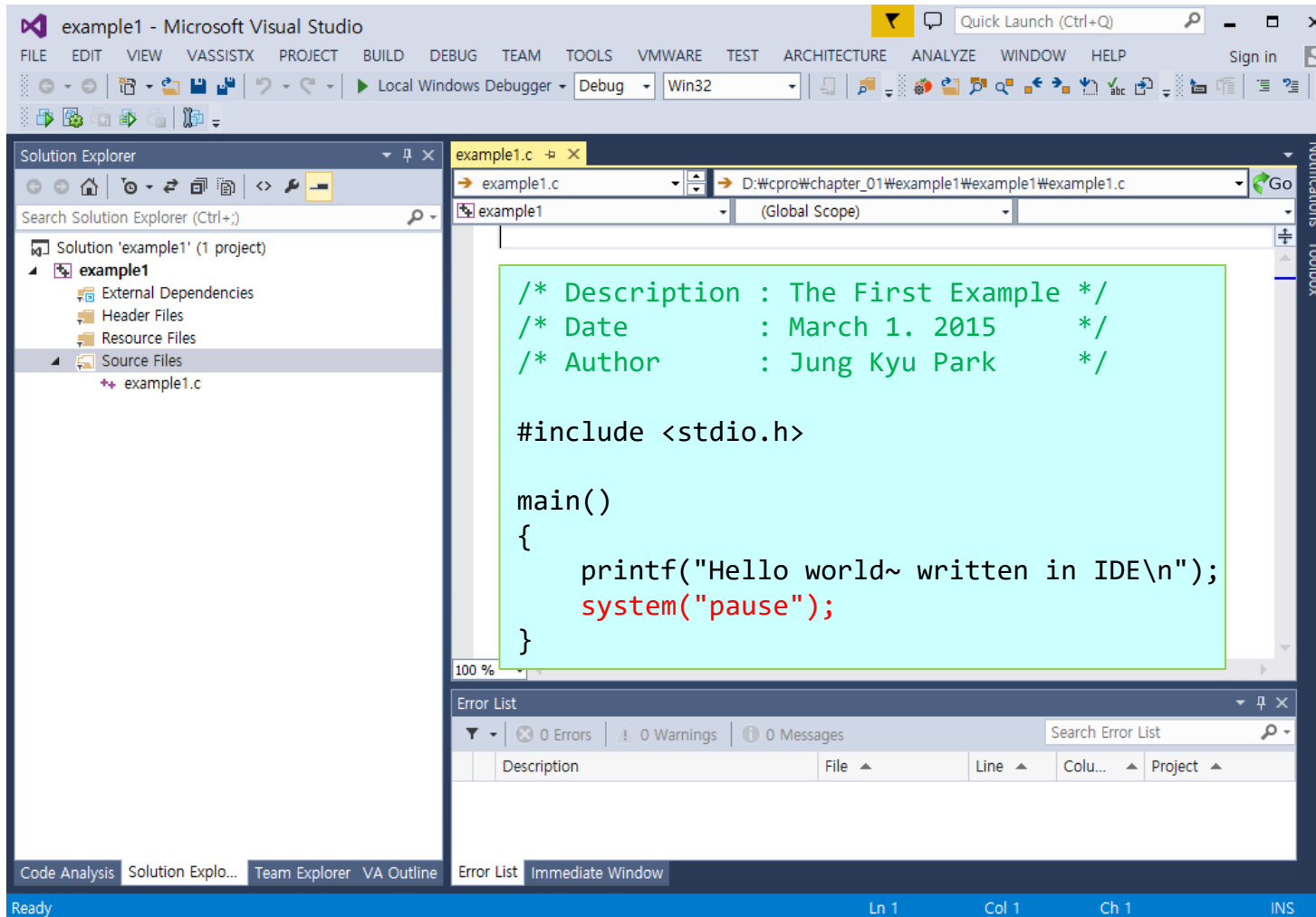
- F7 로 컴파일이 안될 때
  - ✓ Tool – Option (Environment – Keyboard – Visual C++ 6)



- IDE 사용 (project 생성)
  - ✓ Check error



- 결과 보기 : Method 1
  - ✓ system("pause") 넣기



The screenshot shows the Microsoft Visual Studio IDE. The Solution Explorer on the left shows a project named 'example1' with a source file 'example1.c'. The main editor window displays the following C code:

```
/* Description : The First Example */
/* Date       : March 1. 2015      */
/* Author     : Jung Kyu Park     */

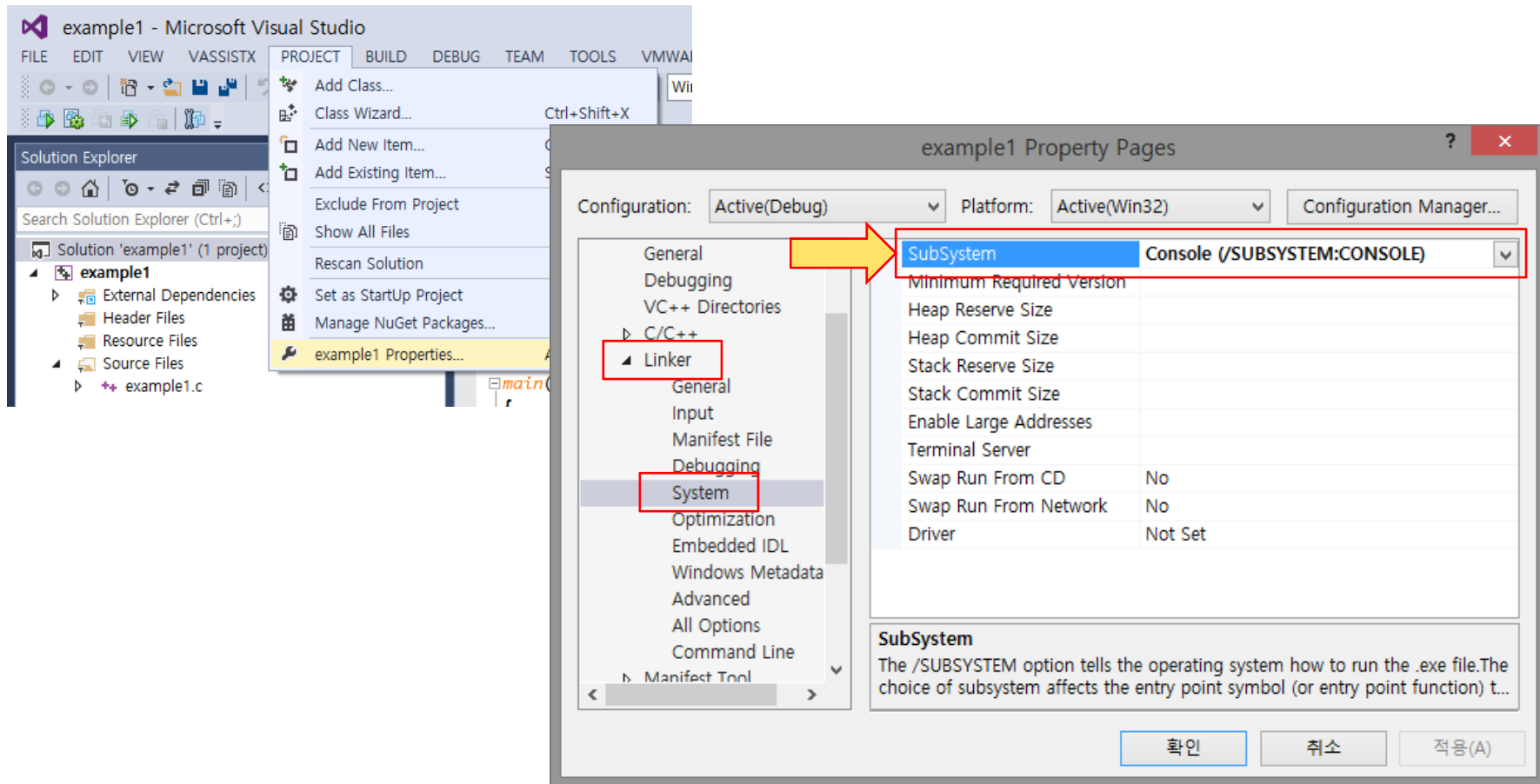
#include <stdio.h>

main()
{
    printf("Hello world~ written in IDE\n");
    system("pause");
}
```

The code is displayed in a light blue background. The 'system("pause");' line is highlighted in red. Below the code editor, the Error List window is visible, showing 0 Errors, 0 Warnings, and 0 Messages. The status bar at the bottom indicates 'Ready' and 'Ln 1 Col 1 Ch 1 INS'.

## ■ 결과 보기 : Method 2

✓ Project → Properties (ALT+F7)



Configuration Properties → Linker → System → Subsystem  
→ Console (SUBSYSTEM:CONSOLE)

- 시작
  - ✓ step 1: MS Visual C++ 실행
- 프로젝트 생성
  - ✓ step 2: [File]-[New] 선택
  - ✓ step 3: project 생성
    - Visual C++, Win32, Win32 Console Application
  - ✓ step 4: Console Application, An empty project
  - ✓ step 5: project 생성 완료
    - \*.sln 프로젝트 파일 생성됨
- 파일 생성
  - ✓ step 6: [Project]-[Add New Items...] or Right Click on Source Files
  - ✓ step 7: C++ File (.cpp)
- 프로그램 작성
  - ✓ step 8: 프로그램 작성. [File]-[Save]
- 컴파일
  - ✓ step 9: [Build]-[Build Solution] (F7)
  - ✓ step 10: Output windows 상에서 error, warning 확인
- 수행
  - ✓ step 11: [Debug]-[Start without Debugging] (Ctrl+F5)
  - ✓ step 12: 실행 결과 확인