변수 타입

형식 이름	바이트	기타 이름	값의 범위
int	4	signed	-2,147,483,648 to 2,147,483,647
unsigned int	4	unsigned	0 to 4,294,967,295
__int8	1	char	-128 to 127
unsigned __int8	1	unsigned char	0 to 255
__int16	2	short, short int, signed short int	-32,768 to 32,767
unsigned __int16	2	unsigned short, unsigned short int	0 to 65,535
__int32	4	signed, signed int, int	-2,147,483,648 to 2,147,483,647
unsigned __int32	4	unsigned, unsigned int	0 to 4,294,967,295
__int64	8	long long, signed long long	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned __int64	8	unsigned long long	0 to 18,446,744,073,709,551,615
bool	1	none	false or true
char	1	none	-128 to 127 by default 0 to 255 when compiled by using /J 옵션
signed char	1	none	-128 to 127
unsigned char	1	none	0 to 255
short	2	short int, signed short int	-32,768 to 32,767
unsigned short	2	unsigned short int	0 to 65,535
long	4	long int, signed long int	-2,147,483,648 to 2,147,483,647
unsigned long	4	unsigned long int	0 to 4,294,967,295
long long	8	none (but equivalent to __int64)	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long long	8	none (but equivalent to unsigned __int64)	0 to 18,446,744,073,709,551,615
enum	varies	none	
float	4	none	3.4E +/- 38 (7 digits)
double	8	none	1.7E +/- 308 (15 digits)
long double	same as double	none	Same as double
wchar_t	2	__wchar_t	0 to 65,535

기본 자료형의 종류

구분	자료형	크기(byte)	범위
문자형	char	1 byte	-128 ~ 127
	unsigned char	1 byte	0 ~ 255
정수형	__int8	1 byte	-128 ~ 127
	__int16	2 byte	-32,768 to 32,767
	unsigned int	2 byte	-32,768 to 32,767
	(signed) short (int)	2 byte	-32,768 to 32,767
	(unsigned short (int)	2 byte	0 ~ 65,535
	__int32	4 byte	-2,147,483,648 ~ 2,147,483,647
	(signed) int	4 byte	-2,147,483,648 ~ 2,147,483,647
	unsigned int	4 byte	0 ~ 4,294,967,295
	(signed) long (int)	4 byte	-2,147,483,648 ~ 2,147,483,647
	unsigned long (int)	4 byte	-2,147,483,648 ~ 2,147,483,647
	__int64	8 byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
실수형	float	4 byte	3.4E +/- 38 (7 digits)
	double	8 byte	1.7E +/- 308 (15 digits)
	long double	8 byte	1.2E +/- 4932 (19 digits)

자료형의 종류

DWORD	unsigned long	4 byte
bool	char	1 byte
BOOL	int	4 byte
BYTE	unsigned char	1 byte
WORD	unsigned short	2 byte
UINT	unsigned int	4 byte

WORD 또는 DWORD

win32 API를 다루다보면 종종 WORD나 DWORD 자료형을 볼 수 있다.

WORD는 CPU가 처리할 수 있는 하나의 단위이다. CPU는 어떤 연산을 진행하기 위해 레지스터라는 공간에 데이터를 가져오게 되고 연산을 진행하게 되는데, 한번에 처리할 수 있는 데이터의 크기는 CPU마다 다르다. 32 비트의 CPU, 64 비트의 CPU의 ~~ 비트는 바로 한번에 처리할 수 있는 양을 나타낸다.

즉, CPU가 한번에 처리할 수 있는 데이터의 크기 단위를 WORD 라고 한다.

WORD / DWORD 의 의미에 대한 고찰

Development/C/C++

WIN32 API / MFC 환경에서 개발하다 보면 종종 볼수 있는게 WORD / DWORD 인데요.

이건 제 개인적인 버릇인데 VC++환경에서 코드를 짜면 DWORD를 많이 씁니다.. (이유는 없고, 처음 배울 때 부터 그렇게 써서;;)

그런데 WORD나 DWORD를 그냥 생각없이 쓰기에는 좀 찝찝한 구석이 있네요. 그래서 짬을 내서 알아봤습니다.

WinDef.h에는 다음과 같이 DWORD / WORD가 정의되어 있습니다.

```
typedef unsigned long          DWORD;
typedef unsigned  short        WORD;
```

따라서 일반적인 WIN32 기반하에서는 DWORD는 4Byte(32-bit) 가 되고, WORD는 2Byte(16-bit)이 됩니다.

근데 왜 32bit Windows에서 WORD의 Size가 16-bit이 되었을까요?

그 이유는 **하위 호환성**에 있습니다.

32-bit machine은 제 기억에 386dx부터 인 것으로 기억이 납니다. 그 전에는 16-bit machine 이었고요.

결국 Intel 8086 Processor부터 이어지는 Intel CPU Architecture의 하위 호환 문제 때문에 Compiler에서도 그냥 WORD를 16-bit로

표현 한 듯 합니다. Intel이 새삼스레 대단하다고 느껴지네요 ㅎㅎ

Visual Studio에서는 이 외에도 DWORD32 / DWORD64 형을 지원합니다.

```
typedef unsigned int DWORD32, *PDWORD32;
typedef unsigned __int64 DWORD64, *PDWORD64;
```

DWORD32를 살펴보니 long대신에 int가 쓰였습니다.

그런데 DWORD64라는 이름은 선뜻 이해가 가지 않네요.. 위의 하위 호환이라는 규칙에 어긋나게 되니까요..-_-;;

주의할 점은 위의 내용이 64-bit 기반의 gcc에서는 VC++과는 크기가 다르기 때문에 멀티 플랫폼 개발시에는 고려를 좀 해야 합니다.