부록 5 Visual Studio 사용법

1. Visual Studio에서 자주 사용되는 메뉴들

Visual Studio를 실행하면 맨 위에 바(bar)형태의 다양한 메뉴들(File, Edit, View, Insert 등)을 볼 수 있다. 이 메뉴바(Menu Bar)에는 프로그램을 작성하기위해 자주 사용되는 메뉴들로 구성되어 있다. 각 기능들에 대해 살펴보도록 하자.

1-1. File 메뉴

메뉴명(단축키)	설 명
New(Ctrl+ N)	새로운 파일을 또는 Project를 생성하기
Open(Ctrl+ O)	기존의 파일을 또는 Project를 불러오기
Close	현재 작업 중인 Project를 닫기
Close Solution	현재 작업 중인 Solution을 닫기
Save(Ctrl+ S)	파일을 저장하기
Save As	다른 이름으로 저장하기
Save All(Ctrl+ Shift+ S)	현재 작업 중인 모든 파일들을 저장하기
Page Setup	여백을 지정하기
Print(Ctrl+ P)	프린트하기
Recent Files	최근에 작업했던 파일의 리스트
Exit(Alt+ F4)	종료하기

1-2. Edit 메뉴

메뉴명(단축키)	설 명
Undo(Ctrl+ Z)	현재 수행했던 바로 이전 상태로 되돌리기
Redo(Ctrl+ Y)	Undo를 수행했던 바로 이전 상태로 되돌리기
Cut(Ctrl+ X)	잘라내기
Copy(Ctrl+ C)	복사하기
Paste(Ctrl+ V)	붙이기
Delete(Del)	삭제하기
Select all(Ctrl+ A)	모두 선택하기
Find and Replace	찾아 바꾸기

1-3. View 메뉴

메뉴명(단축키)	설 명
Solution Exploler(Ctrl+ Alt+ L)	Project에 속한 solution목록을 열람하는 창을 열기
Output(Alt+ 2)	에러 등 컴파일 결과를 확인하는 창을 열기
Full Screen(Shift+ Alt+ Enter)	에디트 창을 Full Screen으로 보기

1-4. Build 메뉴

메뉴명(단축키)	설 명
Build Solution(F7)	컴파일한 후 실행파일을 만들기
Clean Solution	컴파일로 생성된 파일들을 프로젝트에서 삭제

1-5. Debug 메뉴

메뉴명	설 명
Start Debugging(F5)	디버깅하면서 프로그램 실행하기
Start Without Debugging(Ctrl+ F5)	디버깅 없이 프로그램 실행하기

1-6. Window 메뉴

메뉴명	설 명
New Window	새로운 에디트 창을 만들기
Split	현재 에디트 창을 4등분하여 같은 화면을 표시하기
Auto Hide	사용하지 않는 창을 자동으로 숨기기
Hide	현재 작업하던 창을 숨기기
Auto Hide All	모든 창을 자동으로 숨기기

2. Visual Studio에서의 Debug 명령

프로그램을 작성하고 컴파일을 할 때 프로그램에 문법상의 오류가 있으면 컴파일을 할 때 컴파일러가 오류가 발생한 위치를 찾아내 주는 것은 이미 알고 있을 것이다. 그러나 프로그램을 작성하고 컴파일을 해보면 정상적으로 컴파일이 되지만 작성자가 의도하지 않은 행동을 보여주는 경우가 있을 수 있다. 이런 오류를 찾아서 수정하는 일을 디버깅(Debuging)이라고 하는데, Visual Studio에서는 디버깅 작업을 돕기 위한 몇 가지 기능들을 제공하고 있다. 이것들을 Debug 명령이라고 부르고, 다음과 같은 것들이 있다.

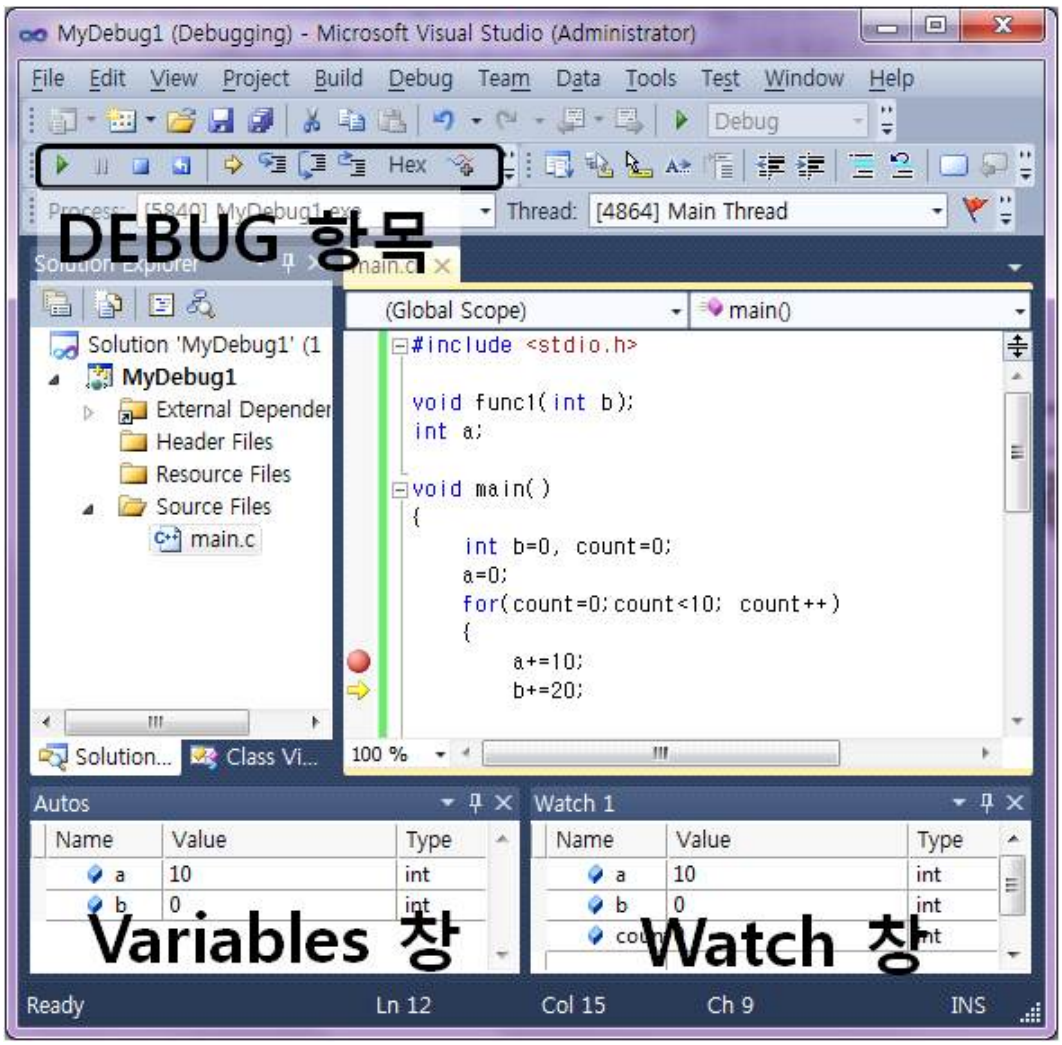
2-1. Debug 모드로 실행/종료

디버깅을 하기 위해서는 프로그램을 Debug 모드로 실행시켜야만 한다. 디버그 모드로 실행하거나 실행을 멈추기 위해서 다음과 같은 몇 가지 기능들이 제공된다.

기 능	실행방법	설 명
Start Debug(Go)	F5, Build→Start Debug→Go	디버그 모드로 실행
Start Debug(Step Into)	F11, Build→Start Debug→Step Into	디버그 모드로 main()함수 시작 직후 까지 실행
Start Debug(Run to cursor)	Ctrl-F10, Build→Start Debug→Run to cursor	디버그 모드로 현재 커서 위치까지 실행
Stop Debugging	Shift-F5, Debug→Stop Debugging	디버그 모드 종료

2-2. Debug 모드 화면 설명

Debug 모드로 프로그램을 실행하면 상단 메뉴의 Debug 항목이 생기고, 하단의 Output 영역에는 Variables 창과 Watch 창이 나타난다(본 화면 구성은 사용자 지정을 통해 달라질 수 있다).



Debug 항목과 Debug창에는 Debug 명령들이 들어있고, Variables 창에는 사용되고 있는 변수들의 이름과 값이 나타난다. Watch 창은 각 변수들을 추적할 때 사용되는 창으로, 값의 변화를 추적하고 싶은 변수가 있다면 Variables 창에서 끌어서 Watch 창에 끌어 놓고 다음에 설명할 Step Into, Step Over, Step Cut 명령으로 프로그램을 한 단계씩 실행시키면서 살펴보면 된다.

항 목	설 명
Debug 메뉴	Debug 명령들이 수록되어 있음
Debug 창	
Variables 창	사용되고 있는 변수들의 목록을 표시
Watch 창	프로그래머가 추적하고자 하는 변수들을 나열해놓고 추적

2-3. Step Into / Step Over / Step Cut

이 세 가지 기능은 프로그램을 한 단계씩 실행시킬 때 사용하는 것이다. 이 기능들을 사용하면 현재 실행할 차례인 라인을 실행 한 후에 프로그램의 실행이 일시정지 된다. 변수나 배열의 값의 변화를 살펴볼 때 사용하면 유용한 기능이다.

항 목	실행방법	설 명
Step Into	F11	한 단계씩 실행하는 중에 함수를 호출하는 부분을 만나면 호출된 함수의 내부로 들어가서 한 단계씩 실행하는 것을 계속
Step Over	F10	한 단계씩 실행하는 중에 함수를 호출하는 부분을 만나면 호출된 함수의 내부에서는 멈추지 않고 실행
Step Cut	Shift-F11	현재 한 단계씩 실행중인 함수의 나머지 부분을 멈추지 않고 실행

2-4. Breakpoint

프로그램상의 특정 지점에서 실행이 일시정지 되도록 하는 기능이다. Debug 모드로 실행하기 전이나 Debug 모드에서 프로그램의 실행이 일시정지 되었을 때 일시정지 시키고자 하는 부분에서 Breakpoint를 설정해주면 프로그램이 실행되다가 Breakpoint가 설정된 라인을 만나면 실행이 일시정지 된다.

Breakpoint를 설정하는 방법은 다음과 같은 두 가지 방법이 있다.

기 능 명	실행방법	설 명
Breakpoint	F9	현재 커서가 위치한 라인에서 일시 정지
	Alt-F9	여러 가지 조건을 설정하여 조건에 맞으면 일시 정지

2-5. Quickwatch

Quickwatch는 Watch창과 마찬가지로 현재 시점의 각종 변수나 배열 등의 값을 보고 편집할 수 있는 기능을 가지고 있지만, Watch창과는 달리 변수나 배열의 이름을 가지고 검색할 수 있는 기능이 있다. Quickwatch 기능을 사용하기 위해서는 Debug 메뉴나 Debug 창에서 Quickwatch를 찾아서 눌러주거나 Shift-F9키를 누르면 된다.

3. 헝가리안 표기법

변수명만 보고도 그 변수의 타입이 무엇인지를 바로 알 수 있도록 변수명에 그 변수의 타입을 나타내는 접두어를 붙이는 방식을 헝가리안 표기법이라고 한다. 표 4에 주로 사용되는 헝가리안 표기법의 접두어를 정리하였다.

접두어	설 명
a	배열
b 또는 f	BOOL형 변수(b는 “bool”, f는 “flag”)
by	BYTE(unsigned char)형 변수
c	카운터로 사용되는 변수
ch	char형 변수
cx, cy	x, y 길이를 나타내기 위해 사용되는 변수
d	날짜형 변수
dbl	double형 변수
h	핸들(HANDLE)형 변수
n 또는 i	int형 변수
l	long형 변수
p	포인터 변수
lp	long(far)포인터 변수(32비트 프로그래밍에서는 일반 포인터와 같음)
s	문자열
sz	널(NULL) 문자로 끝나는 문자열
u	unsigned int형 변수
w	WORD(unsigned short)형 변수
dw	DWORD(unsigned long)형 변수
str	CString형 변수

표 4. 형가리안 표기법.

표 4를 가지고 m_lpszFilename라고 표기된 변수의 의미를 분석해보면, ‘m_’는 클래스 멤버 변수임을 표시하는 것이고, ‘lp’는 long 포인터 변수, ‘sz’는 널 문자로 끝나는 문자열을 나타낸다. 따라서, m_lpszFilename는 어떤 클래스의 멤버 변수인데, 파일의 이름을 널 문자로 끝나는 문자열의 형태로 저장하고, 그것의 주소를 가리키기 위한 포인터임을 알 수 있다.

4. 윈도우 프로그래밍에서 정의된 데이터 형

윈도우 프로그래밍에서는 표 5와 같은 데이터 형들이 정의되어 사용된다.

데이터 형	의 미
BOOL	논리형. TRUE 또는 FALSE 값만 가질 수 있음
BYTE	unsigned char (8bit)
DWORD	unsigned long (32bit)
DWORDLONG	unsigned double (64bit)
FLOAT	float
LONG	signed long (32bit)
LONGLONG	signed double (64bit)
LPARAM	32bit 메시지 파라미터
LPCSTR	널 문자로 끝나는 윈도우 문자열 상수의 포인터
LPCTSTR	널 문자로 끝나는 유니코드 또는 윈도우 문자열 상수의 포인터
LPSTR	널 문자로 끝나는 윈도우 문자열의 포인터
LPTSTR	널 문자로 끝나는 유니코드 또는 윈도우 문자열의 포인터
TCHAR	유니코드 또는 윈도우 문자
UINT	unsigned int (32bit)
WORD	unsigned short (16bit)
WPARAM	16bit 메시지 파라미터

표 5. 윈도우 프로그래밍에서 정의된 데이터 형.