# 3장 헬로 TIES

[3장 헬로 TIES](#_uxmpzd5ydcq5)

[3.1 Hello, TIEs](#_3cpsf4q1ddio)

[3.2 프로그램 소스코드 이해하기](#_77ec2ml0ix0w)

[3.3 코드를 한줄씩 실행해 보기](#_skac62etbv48)

[3.4 온라인 C 컴파일러로 실행해 보기](#_p016goiupu6a)

[3.5 요약](#_t2ru9axj9sit)

이번 장에서 배울 내용은

* 2장에서 구성한 개발 환경을 이용하여 간단한 프로그램을 작성해 봅니다.
* 프로젝트 파일의 기본 구조, 소스 코드의 기본 골격을 이해합니다.
* 소스코드를 한 줄씩 실행해 봅니다.

## 3.1 Hello, TIEs

2장에서는 앞으로 계속 사용하게 될 개발 환경을 구성해 보았습니다. 이번 장에서는 개발 환경에서 간단한 프로그램을 만들어 실행해 보면서 프로그램의 구조를 살펴 봅니다. 함께 만들어 볼 프로그램은 콘솔창에 “Hello, TIES” 메시지를 출력하는 프로그램입니다. 콘솔 창은 텍스트 명령어를 통해 컴퓨터와 컴퓨터와 직접 소통할 수 있는 창구입니다. 운영체제에 따라 터미널(Terminal) 또는 커맨드 라인(Command Line), 즉 명령행 창으로도 불립니다.

NOTE\_

C언어가 개발될 1970년도에는 지금 우리가 사용하는 것과 같은 높은 수준의 사용자 인터페이스, 즉 GUI 환경이 없었습니다. 특히 C언어는 애당초 시스템 프로그래밍 언어로 설계되었기 때문에 GUI를 개발하는데에는 부족한 점이 많습니다. GUI 프로그래밍을 하고 싶다면 대상 GUI 운영체제에 맞는 프레임워크에 맞게 코드를 작성해야 합니다. 프레임워크(framework)는 특정 목적에 필요한 프로그램 코드 및 틀을 하나로 묶은 것입니다.

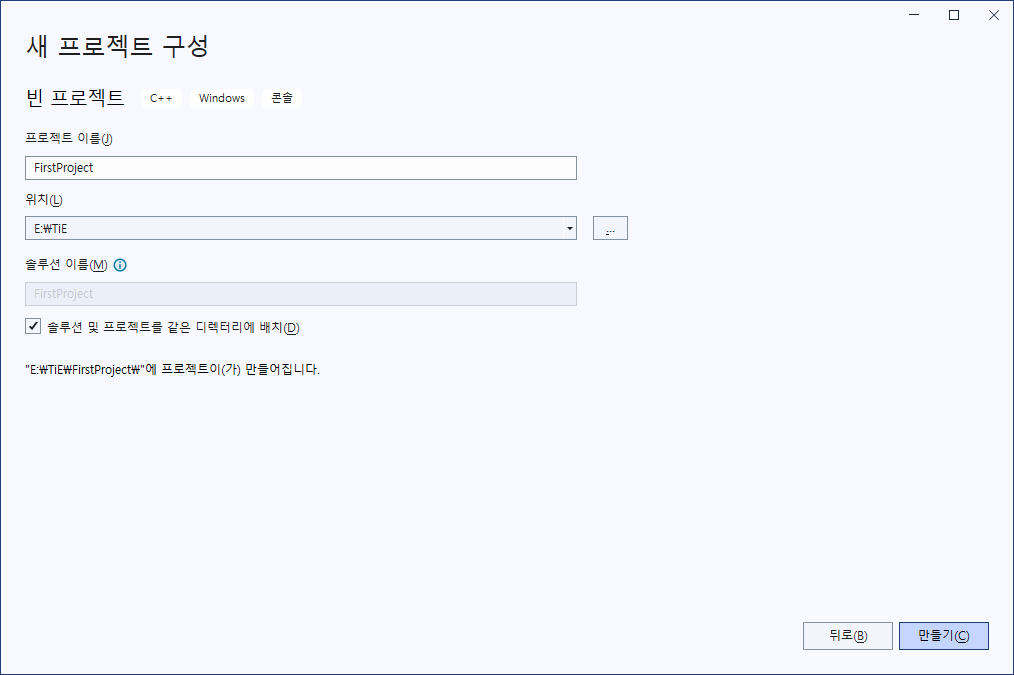
2장에서 C 언어 소스 코드가 실행 파일로 만들어지는 과정을 살펴 보았습니다. 보통 여러 개의 C 소스 코드를 컴파일하여 하나의 프로그램을 생성합니다. 하지만 우리는 당분간 하나의 소스 코드 파일로 하나의 실행 파일을 만듭니다. 프로젝트(Project)는 하나의 프로그램을 만드는데 필요한 소스 파일과 기타 파일을 그룹으로 묶은 것입니다. 비주얼 스튜디오는 프로젝트를 기반으로 동작하므로, 반드시 프로젝트를 만들어야 합니다.

이번 장은 설명에 따라 처음부터 마지막 단계까지 진행하는 것이 중요합니다. 때로는 이해되지 않는 부분이 있더라도 끝까지 단계를 따라해 본 다음, 코드의 세부 항목을 살펴보는 것이 좋습니다.

1 단계: 비주얼스튜디오를 실행하면 다음과 같이 프로젝트 만들기 대화창이 나타납니다. 프로젝트에 사용할 템플릿을 선택합니다. 템플릿(template)이란 무언가를 만들 때 사용되는 기본 틀입니다. C언어 템플릿은 제공되지 않으므로 빈 프로젝트를 선택합니다.

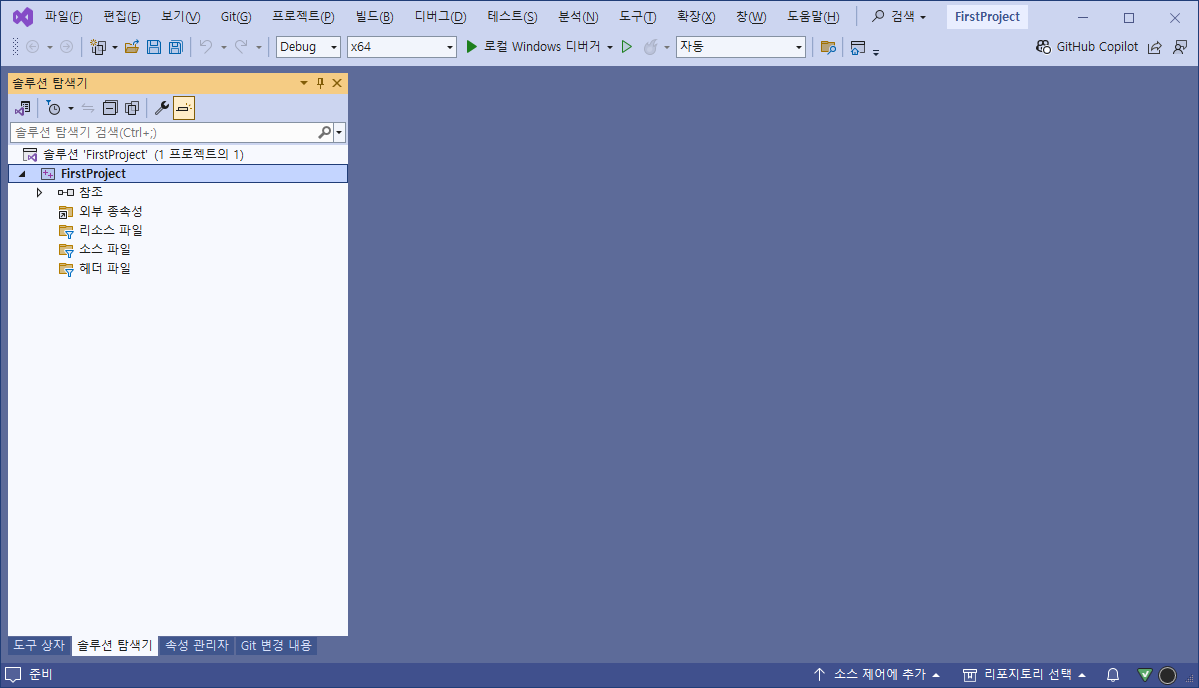


2 단계: 새로운 프로젝트를 구성하기 위하여 프로젝트 이름과 프로젝트에서 사용할 기본 디렉터리를 지정합니다. 이때 솔루션 파일도 프로젝트 위치에 함께 만들어 집니다. 솔루션(Solution) 파일은 큰 프로그램을 만들 때 관련있는 프로젝트를 모아두는 것입니다. 이 책에서 다루는 예제는 프로젝트 마다 솔루션 파일을 만듭니다..

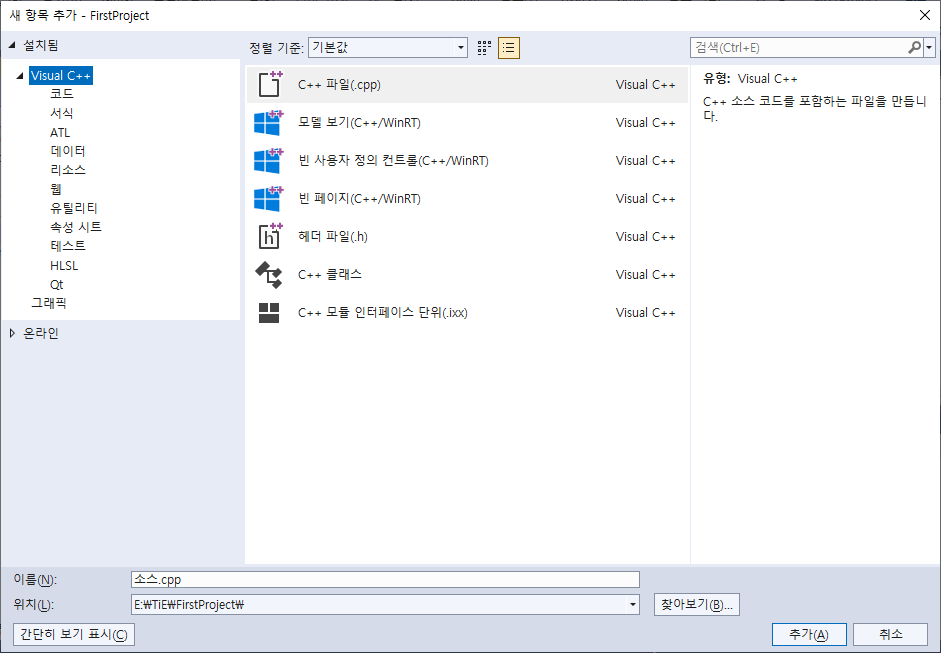


3단계. 드디어 빈 프로젝트가 만들어졌습니다. FirstProject 솔루션에 FirstProject 프로젝트가 들어있습니다. 프로젝트는 아무것도 들어 있지 않은 빈 상태로 구성됩니다. 기본 설정에 따라 솔루션 탐색기가 왼쪽에 표시됩니다. 솔루션 탐색기가 표시되지 않는다면 보기 > 솔루션 탐색기를 선택하거나 Ctrl + Alt + L을 누릅니다. 솔루션 탐색기에서 프로젝트가 어떤 파일로 이루어지는 지를 확인할 수 있습니다.

프로젝트를 구성하는 파일은 폴더를 만들어 관련 파일을 관리할 수 있습니다. 비주얼스튜디오는 기본적으로 소스 파일, 헤더 파일, 리소스 파일 폴더를 생성합니다.



4단계.프로젝트에서 프로그램을 만들기 위한 소스 파일을 추가합니다. 솔루션 탐색기 창의 소스 파일 폴더를 선택한 후 마우스 오른쪽 버튼을 누릅니다. 팝업 메뉴에서 추가 > 새항목을 선택합니다. 추가할 새 항목의 템플릿이 표시됩니다. 템플릿 목록에서 ‘C++ 파일(.cpp)’를 선택한 다음, 아래쪽 이름에 “main.c”를 입력한 다음, 추가 버튼을 누릅니다. 소스 파일의 이름은 원하는 대로 사용할 수 있지만, 파일의 확장자는 반드시 ‘.c’여야 합니다. 컴파일러^^는 파일 확장자에 따라 해당 소스 파일을 어떻게 컴파일해야 하는지를 파악합니다.

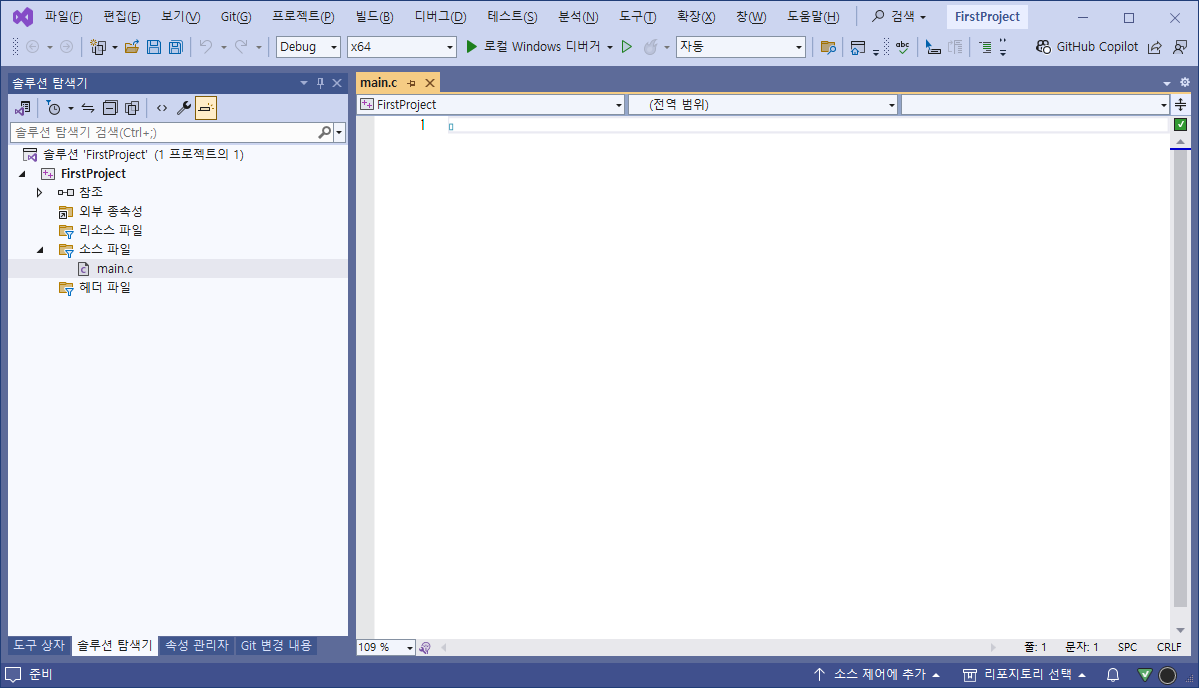


NOTE\_

컴파일러는 소스 파일을 이용하여 최종 결과인 실행 프로그램을 만들기 위한 자신만의 AST 자료구조를 구축합니다. 사용하는 언어에 따라 AST 자료 구조를 구성하는 방식이 달라집니다. 비주얼스튜디오는 기본적으로 .C는 C언어 소스 파일로, .CPP는 C++언어 소스파일로, .H는 헤더 파일로 인식하여 해당 프로그래밍 언어의 구문 규칙에 맞게 소스 코드를 이해합니다.

5단계. 프로젝트의 소스 파일 폴더에 main.c파일이 생성되었습니다. 언제든지 솔루션 탐색기에서 파일을 더블 클릭하면 해당 파일이 오른쪽 편집창 영역에 열립니다. 편집 창에서 소스코드를 살펴보거나 수정할 수 있습니다.

비주얼 스튜디오는 통합 개발환경이므로, 다른 편집기 프로그램을 사용하지 않고도 편집창에서 소스 코드를 작성하거나 수정할 수 있습니다. 편집창은 코드 작성에 도움이 되는 다양한 기능을 제공합니다. 기본적으로 줄마다 왼쪽에 줄 번호가 표시됩니다. 소스 코드를 입력하면 자동으로 코드의 의미 단위별로 다른 색상으로 표시됩니다.



6단계. 이제 소스 코드를 입력해 봅시다. 편집 창에서 코드를 다음과 같이 입력합니다.

/\* TIES-C Programming \*/

#include <stdio.h>

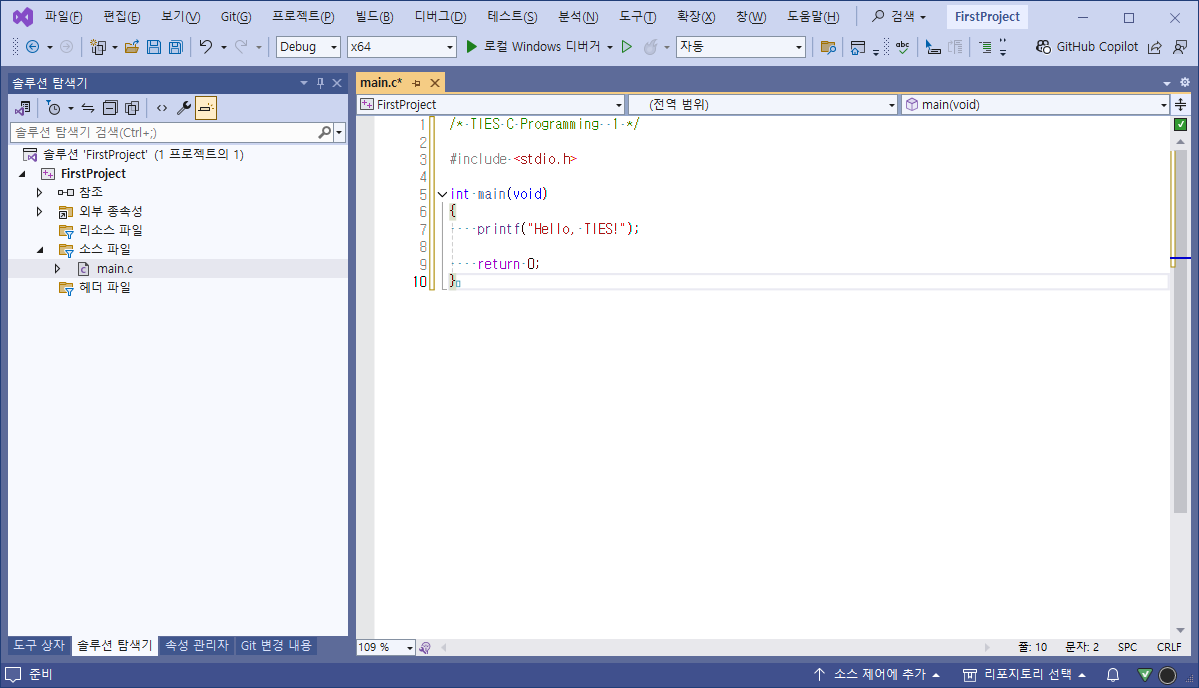
int main()

{

printf("Hello, world!\n");

return 0;

}



코드를 입력하면 위와 같이 표시됩니다.

이 소스 코드는 C 프로그래밍에 대한 최초의 책에 소개된, C 언어로 작성된 가장 오래된 코드를 살짝 바꾼 코드입니다.

편집기 창에서 코드를 입력한 다음, 실행 파일을 만들려면 빌드 과정을 거쳐야 합니다. Ctrl+B 또는 빌드> 솔루션 빌드를 선택합니다. 프로젝트 빌드 설정에 따라 프로젝트 폴더의 Debug\Release폴더에 실행파일이 만들어 집니다.

프로젝트 빌드 설정은 상단 툴바의 콤보 박스에서 변경할 수 있습니다.

정상적으로 빌드가 수행되지 않고 오류가 발생했다면 입력한 소스 코드가 예제 코드와 정확히 일치하는지를 확인합니다.

NOTE\_

빌드 과정을 거쳐 만들어진 실행 프로그램은 다른 컴퓨터로 복사해서 실행해도 동작합니다. 다만 실행파일의 크기를 줄이기 위해 공통적인 실행 코드를 운영체제에 하나만 설치해 두고 함께 사용합니다. 이를 C 런타임 라이브러리라고 부릅니다. 보통은 기본적으로 설치되어 있습니다만, 만약 설치되어 있지 않다면 따로 설치해야 합니다. 디버그용 C런타임 라이브러리는 보통 설치되어 있지 않습니다. 또한 디버그용 빌드에는 개발과정에서 필요한 부가 정보가 함께 포함되어 있습니다. 따라서 개발한 프로그램을 다른 컴퓨터에 배포할 때에는 릴리즈 빌드를 전달합니다.

7단계. 만들어진 프로그램을 실행해봅시다. 메뉴에서 디버그 > 디버깅 시작을 선택하거나 F5를 누르면 우리가 만든 프로그램이 실행된 결과를 볼 수 있습니다. 제대로 동작했다면 화면에 “Hello, TIES!” 문구가 표시됩니다.



## 3.2 프로그램 소스코드 이해하기

프로젝트는 여러 개의 C 언어 코드 파일로 이루어집니다.이 파일은 소스 파일과 헤더 파일로 나누어 집니다. 보통 관련있는 파일들을 같은 위치에 저장합니다.하지만 파일의 사용 목적에 따라 논리적인 그룹으로 묶기도 합니다. 언제든 솔루션 탐색기에서 논리적으로 폴더를 생성하여 관련있는 파일들을 하나의 그룹으로 묶을 수 있습니다.

지금부터 우리가 작성한 C 소스 코드를 상세히 살펴 보겠습니다.

소스 코드는 기본적으로 위에서 아래로 한 줄씩 동작을 수행하면서 주어진 작업을 수행하도록 작성합니다. 관련성 있는 작업을 하나의 기본 실행 단위로 묶은 것을 함수라고 합니다. 함수에 대해서는 나중에 보다 자세히 다룹니다.

C언어로 작성된 프로그램은 main 함수에 정의된 작업을 수행합니다. 즉, 프로그램이 실행되면 main함수 내에 작성된 코드를 시작하며, main함수의 코드를 모두 수행하고 나면 프로그램은 종료됩니다. 따라서 C 프로그램을 작성할 때 반드시 작성해야 하는 단 하나의 함수가 있다면 main함수여야 합니다.main 함수는 두가지 형태를 가질 수 있는데, 우선 int main(void) { } 형태만 살펴봅니다.

함수는 이름을 가지며, 함수가 수행할 동작은 중괄호, 즉 ‘{‘와 ‘}’ 사이에 작성합니다. 함수에 대해 알아야 할 더 많은 내용이 있지만, 이번 장에서는 C프로그램은 main함수에서 시작하여 main함수가 종료되면 프로그램이 종료된다는 정도로만 이해하고 넘어가겠습니다.

[그림 추가 예정 - main 함수 시작과 끝]

C 소스 코드에서는 각 명령 구문을 구분하기 위하여 세미콜론(;)을 사용합니다. 즉, 단위 동작을 수행하는 코드마다 세미콜론을 붙여 주어야 합니다.

NOTE\_ 구문(statement)과 표현식(expression)

구문/표현식에 대한 설명.. 나중에 추가 하겠습니다.

프로그램이 수행할 동작을 코드로 개발자가 작성해야 하지만, 자주 사용하는 기능은 컴파일러에서 미리 구현하여 제공합니다. 이와 같이 개발자들이 공통적으로 자주 사용하는 기능을 미리 구현된 함수들의 모음이 표준 라이브러리(standard library)입니다.

표준 라이브러리에서 제공하는 여러 함수 중 printf 함수는 화면에 문자열을 표시하는 동작을 수행합니다. printf(“출력 문자열");와 같은 코드를 작성하면 콘솔 창에 “출력 문자열”을 표시합니다.

컴파일러는 개별 소스코드 단위로 동작하므로, 표준 라이브러리를 비롯하여 해당 파일에 정의되지 않은 함수에 대해서는 어떤 식으로 사용되는지를 알 수 없습니다. 이 문제를 해결하기 위하여 사용하는 함수의 모양을 선언해 주어야 하는데, 이와 같은 함수 선언을 모아둔 파일이 헤더 파일입니다. 확장자는 .h를 사용하며, 특정 함수를 사용하기 위해서는 include 문을 사용하여 적절한 헤더 파일을 포함시켜주어야 합니다.

#include <stdio.h>

stdio.h 파일에는 printf 함수를 비롯하여 입출력과 관련된 표준 함수들이 선언되어 있습니다.자주 사용하는 몇가지 표준 라이브러리 헤더파일은 다음과 같습니다.

|  |  |
| --- | --- |
| 헤더 파일명 | 설명 |
| stdio.h | 표준 입출력 함수 |
| stdlib.h | 표준 일반 유틸리티 함수. 메모리 관리, 문자열 변환, 알고리즘에 관련된 함수 |
| string.h | 문자열을 다루는 함수 |
| time.h | 시간과 날짜를 다루는 함수 |
| thread.h | 스레드 관련 함수. C11 표준에 추가됨 |

main 함수의 마지막에 return 구문이 사용되었습니다. return 구문은 함수를 호출한 코드에게 함수의 실행결과를 알려주기 위해 사용하는 코드입니다. 보통 미리 정의된 값을 반환합니다. main함수는 다른 함수에서 호출한 함수가 아니므로, 운영체제에 해당 값을 알려줍니다. 일반적으로 아무 문제 없이 동작하였을 경우 0값을 되돌려 줍니다.

지금까지 프로그램의 수행 동작과 관련된 코드를 살펴보았습니다. 하지만 몇가지 살펴보지 않은 코드가 있습니다.

주석(comment)은 사람을 위해 소스 코드에 표시해 둔 추가 설명입니다. 보통 코드의 기능에 대한 추가 설명이나 기존 코드의 일부를 잠시 사용하지 않을 때 주석을 사용합니다. 컴파일러가 코드를 읽고 파악할 때 주석 부분은 별도의 처리 과정을 거치지 않고 그냥 지나가므로, 프로그램의 동작에는 아무런 영향을 미치지 않습니다.

C언어에서 주석은 한줄 주석과 범위 주석, 두가지 형태로 사용합니다.

한 줄 주석은 ‘//’를 사용하여, 해당 위치부터 그 줄의 나머지 부분을 주석으로 표시합니다.

// 화면에 Hello, TIES 문장을 출력한다.

printf(“Hello, TIES”);

printf(“Hello, TIES”); // 화면에 Hello, TIES 문장을 출력한다.

범위 주석은 ‘/\*’와 ‘\*/’를 지정한 영역만 주석으로 표시합니다. ‘/\*’로 시작된 이후 ‘\*/’가 표시될 때까지의 범위가 주석으로 처리됩니다. 물론 예제 코드와 같이 한 줄 주석으로도 사용할 수 있습니다.

/\* 한줄로 된 주석문을 다음과 같이 작성할 수 있다.\*/

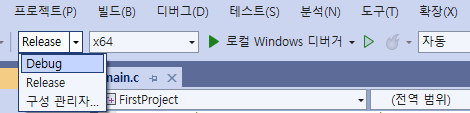
/\* 여러 줄로 된 주석문을

다음과 같이 작성할 수 있다. \*/

## 3.3 코드를 한줄씩 실행해 보기

지금까지는 실행 파일을 만들어 실행해 보았습니다만, 프로그래밍을 하다 보면 한줄씩 코드를 실행해 보면서 코드가 원하는 대로 동작하는지를 확인하고 싶을 수 있습니다. 또는 원하는 대로 동작하지 않은 이유를 분석하기 위해 코드를 한 줄씩 실행해 보고 싶을 수 있습니다. 이 과정을 디버깅(debugging)이라고 부릅니다.

디버깅을 하려면 먼저 대상 빌드를 디버그로 설정해야 합니다. 비주얼 스튜디오의 툴바에서 구성을 Debug로 설정합니다. Release를 선택하면 실행파일의 속도를 빠르게 하고 크기를 줄이기 위하여 실행 파일을 만드는 과정에서 디버깅, 즉 코드를 확인하는 과정에 필요한 부가 정보들을 모두 제거합니다. 개발하는 과정에서는 디버그 빌드를 사용하고, 완성된 프로그램을 사용할 때에는 Release를 선택하여 실행파일을 만듭니다.



먼저 코드에서 디버그 > 디버깅 시작, 또는 F5를 눌러 디버깅을 시작합니다. 하지만 이전과 동일하게 프로그램이 실행되면서 메시지가 출력되고 프로그램은 종료됩니다. 디버깅을 하려면 살펴보려는 코드에 중단점(break point)을 설정해야 합니다.

예제에서는 7번 줄에 중단점을 설정합니다. 코드 편집창에서 7번 줄로 커서를 옮긴 다음, F9 키를 누릅니다. 줄번호 옆에 빨간 원 모양이 표시됩니다. 컴파일러는 디버그 빌드를 실행하다가 중단점을 만나면 해당 줄 코드의 실행을 잠시 멈춥니다.실행이 멈춰진 상태에서 호출스택이나 메모리 검사 창을 통해 프로그램이 어떤 상태인지를 확인할 수 있습니다.



예제에서 아직 데이터를 다루지 않기 때문에, 데이터를 확인하기 보다는 한 줄씩 코드를 실행해 보겠습니다. 다시 F5를 눌러 디버깅을 시작합니다. 이번에는 콘솔창이 만들어 진 다음, 실행이 잠시 멈춰진 상태로 비주얼스튜디오 코드 7번행에 화살표가 표시되면서 현재 멈춘 코드의 위치가 표시됩니다. 멈춰진 상태에서는 현재 메모리의 상태는 어떤지, 어떤 값이 저장되어 있는지를 확인해 볼 수 있습니다.

F10키를 누르면 코드 구문을 하나씩 실행합니다. 따라서 printf 구문이 실행되면서 콘솔에 Hello, TIES! 가 표시되고 9번 행으로 화살표가 옮겨집니다.

7번행 다음에 왜 9번행이 실행될까요? C컴파일러는 코드에 있는 빈 줄은 따로 처리하지 않습니다. 따라서 다음 실행 구문이 있는 9번 행으로 실행 지점이 옮겨집니다. 빈 줄은 컴파일러가 무시하지만, 개발자들은 빈 줄을 사용하여 코드의 맥락을 표현해 줍니다. 코드 실행 블럭의 의도가 변경되는 부분에서 적절히 빈 줄을 추가하여 코드 논리 영역을 시각적으로 나눠 주면 훨씬 쉽게 코드를 이해할 수 있습니다.

다시 한번 F10 키를 누르면 “return 0;” 구문이 실행되면서 main함수의 실행이 끝나고, 프로그램의 실행은 끝나게 됩니다.

디버깅은 프로그램을 만드는 과정에서 어쩔 수 없이 마주하게 되는 과정입니다. 프로그램이 의도한 동작을 제대로 수행할 때까지 디버거 도구를 사용하여 코드를 확인하고 수정하는 과정을 거치게 됩니다. 디버깅 도구는 특정 코드까지 실행하기,함수 단위로 실행하기, 변수값 확인하기 등 훨씬 다양한 기능을 가지고 있습니다. 나중에 함수를 배운 다음 디버거를 사용하는 방법을 다시 한번 살펴보도록 하겠습니다.

TIPS\_

 맥이나 리눅스에서 C언어를 공부하는 경우라면 gdb(GNU debugger)를 사용합니다. 보통 IDE에는 코드 작성 도구, 컴파일러 도구와 디버깅 도구가 모두 통합되어 하나의 도구에서 매끄럽게 원하는 도구를 사용할 수 있습니다.

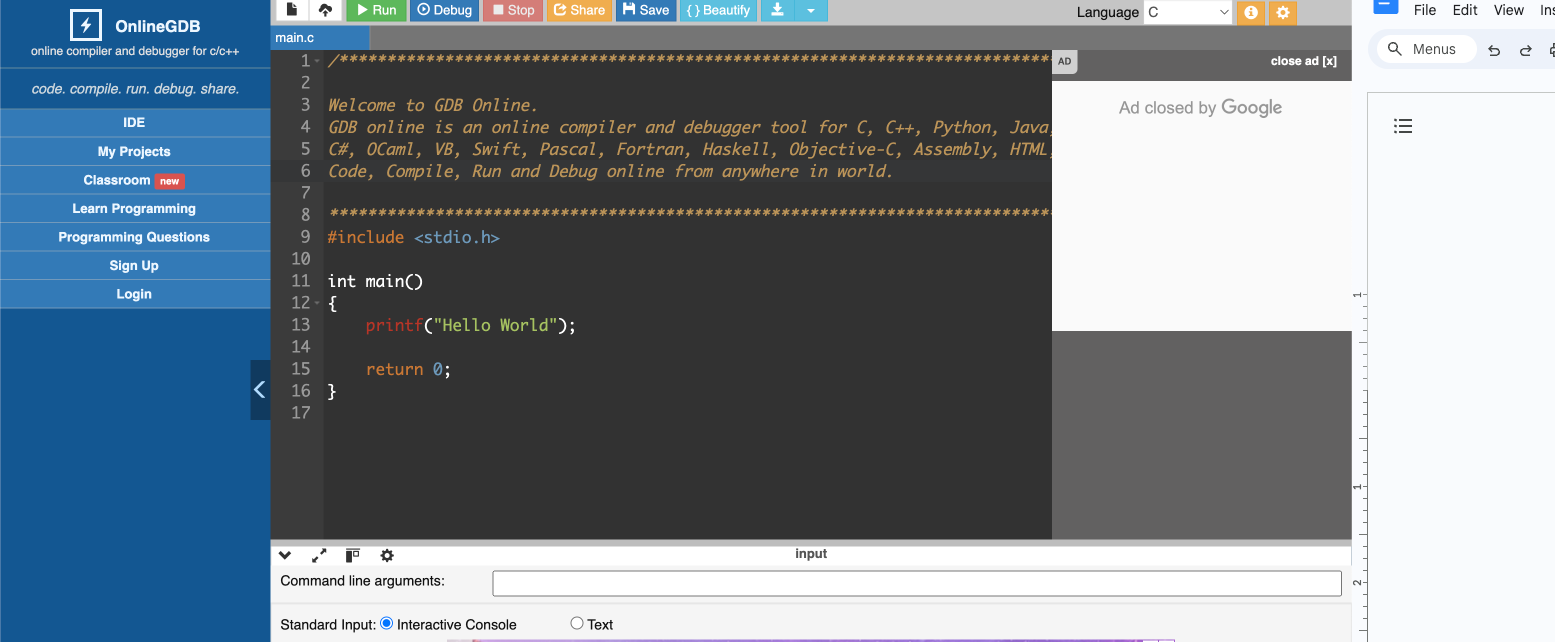
## 3.4 온라인 C 컴파일러로 실행해 보기

지금까지 비주얼스튜디오가 설치된 환경에서 C코드를 작성해 실행해 보았습니다. 이번에는 웹 브라우저에서 온라인 C 컴파일러에 접근하여 코드를 작성한 후 실행해 보겠습니다.

온라인 C 컴파일러는 기본적으로 서버에서 실행된 결과만 웹으로 표시해 주므로 실행파일을 만들 수는 없습니다. 다만 특별한 도구를 설치하지 않아도 웹 브라우저만 있으면 C언어를 배우는 과정에서 손쉽게 코드를 작성, 실행해 볼 수 있습니다.

여러 온라인 컴파일러가 나와 있지만,  온라인 GDB를 사용합니다. 특별한 차이는 없으므로 본인이 마음에 드는 온라인 컴파일러를 사용하면 됩니다.

브라우저를 실행하여 주소 창에 <https://onlinegdb.com> 을 입력합니다. 다음과 같은 편집창이 표시됩니다.



오른쪽 상단에 있는 Language 콤보 리스트에서 C를 선택합니다. 그러면 편집창의 파일명이 자동으로 main.c로 변경됩니다.

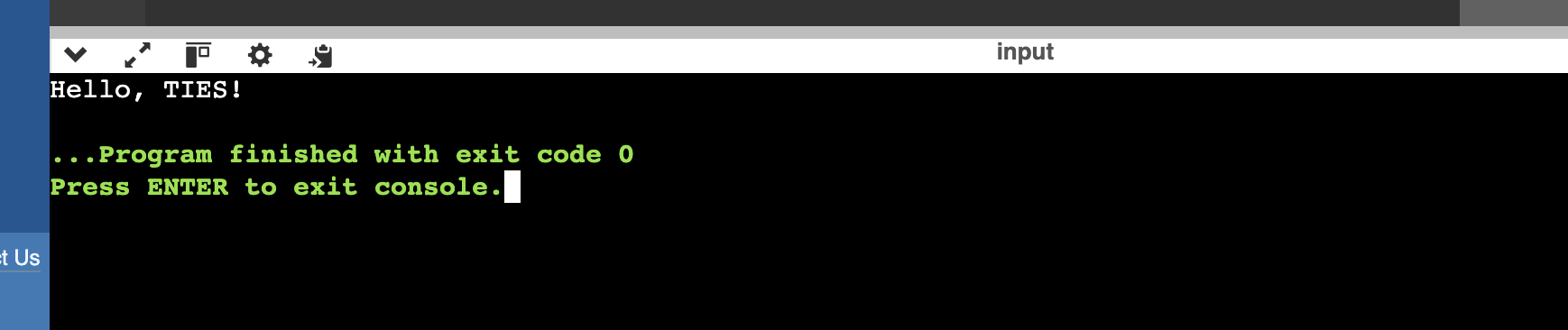
편집창에는 이미 우리가 작성한 예제와 비슷한 코드가 채워져 있습니다. 온라인 컴파일러 역시 편집 창에 줄 번호가 표시되고, 키워드에 따라 다양한 색상으로 표시되어 코드를 쉽게 이해할 수 있습니다. 이번에는 코드를 읽어 봅시다.

기존 예제 코드와 비슷한 모양새를 갖추고 있습니다. 13번째 줄에 출력되는 문자열만 우리가 작성한 예제 코드와 다릅니다. 한번 바꿔서 실행해 봅시다.

기존: printf(“Hello World”);

변경: printf(“Hello, TIES!”);

이제 편집창 윗부분에 있는 버튼 중 Run(실행 버튼)을 눌러 봅니다. 그러면 편집창에 작성된 코드는 서버에서 컴파일된 후, 아무런 문제없이 컴파일되었다면 서버에서 실행되어 실행 결과가 브라우저의 편집창 아래쪽에 표시됩니다.



디버그 기능도 사용할 수 있습니다. 다만 우리가 사용하는 온라인 컴파일러는 gdb를 사용해야 합니다. 상단의 Debug버튼을 누릅니다. 그러면 실행결과가 표시되었던 화면에 gdb 메시지가 표시되고, 편집창 오른쪽에는 디버깅 과정을 돕기 위한 패널이 표시됩니다.



비주얼스튜디오에서는 중단점을 마우스로 설정할 수 있었습니다만, GDB에서는 명령으로 설정해야 합니다. 위 코드에서 main.c 파일의 13번째 줄의 코드에 중단점을 설정해 보겠습니다. 해당 줄에 중단점을 설정하기 위하여 “break main.c:13” 명령을 입력합니다.

Reading symbols from a.out...

(gdb) break main.c:13

Breakpoint 1 at 0x1151: file main.c, line 13.

중단점이 설정되었으므로 디버그 모드로 컴파일된 프로그램을 실행해 봅니다. “run”명령을 입력합니다. 그러면 프로그램이 실행되어 중단점을 만날 때까지 실행되고, 중단점이 설정된 코드가 화면에 표시됩니다.

(gdb) run

Starting program: /home/a.out

[Thread debugging using libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

Breakpoint 1, main () at main.c:13

13 printf("Hello, TIES!");

(gdb)

중단점이 설정된 코드 부분을 실행하려면 “next”명령을 입력합니다. 이제 한줄씩 코드가 실행되면서 프로그램이 종료될 때까지의 상태를 확인할 수 있습니다.

(gdb) next

15 return 0;

(gdb) next

16 }

(gdb) next

\_\_libc\_start\_call\_main (main=main@entry=0x555555555149 <main>, argc=argc@entry=1, argv=argv@entry=0x7fffffffec58) at ../sysdeps/nptl/libc\_start\_call\_main.h:74

74 ../sysdeps/nptl/libc\_start\_call\_main.h: No such file or directory.

(gdb)

GDB는 매우 다양한 명령어를 통해 여러 동작을 수행할 수 있습니다. 하지만 그 사용법을 익히는 것은 이 책의 범위를 벗어나므로 더 이상 다루지는 않겠습니다.

이 책의 대부분의 예제를 익힐 때에는 온라인 컴파일러를 사용해도 괜찮습니다. 다만 회원 가입을 하지 않으면 브라우저를 종료하면 작성한 코드가 사라져 버리고, 실행 결과는 확인할 수 있지만 실행 파일을 만들 수 없습니다. 또한 디버깅 과정도 자신의 컴퓨터에 설치된 개발환경에 비해 불편합니다. 따라서 이 책에서는 온라인 컴파일러로도 코드를 익힐 수 있지만, 비주얼 스튜디오를 설치한 환경을 기준으로 설명합니다.

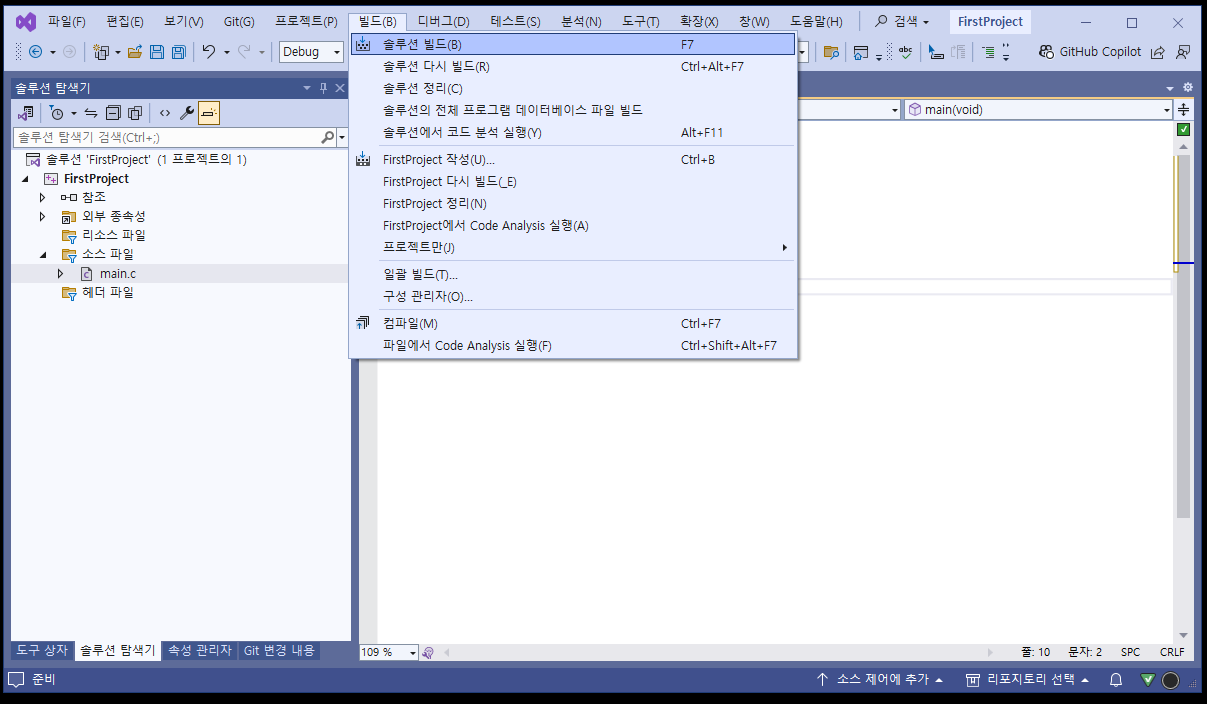
## 3.5 요약

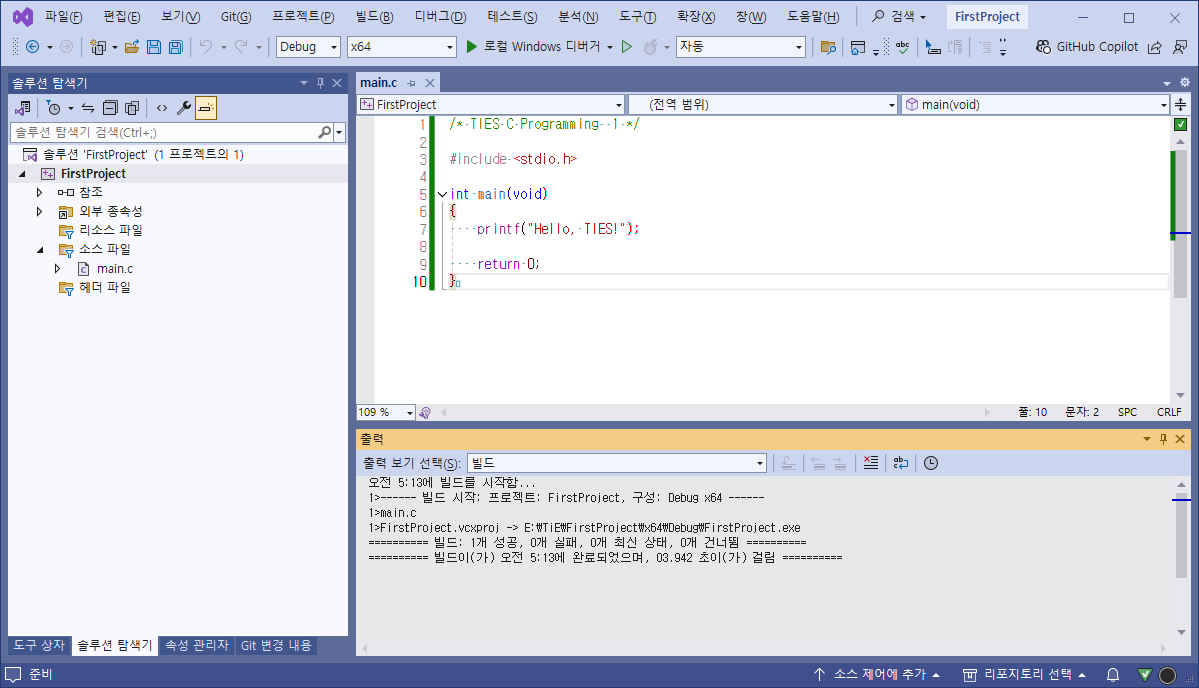
이번 장에서는 매우 간단한 콘솔 프로그램을 만들어 보면서 프로그램의 구조와 개발환경의 사용법에 대해 살펴보았습니다.

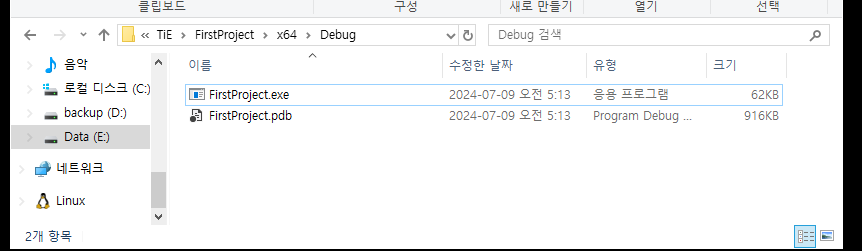
* C 프로그램에는 반드시 main함수가 있어야 합니다.
* 프로그램은 main함수에 작성된 코드를 순서대로 실행하며, main함수의 실행을 마치면 프로그램은 종료됩니다.
* C 코드에서 구문은 세미콜론(;)으로 끝납니다.
* 주석은 코드의 이해를 돕기 위해 코드에 남겨두는 문장입니다. 적절한 주석은 코드의 이해를 돕습니다.
* 디버그는 개발자가 의도한 대로 코드가 동작하는 지를 확인하기 위한 도구입니다. 중단점을 설정하여 특정 코드가 실행된 순간에 프로그램의 실행 상태를 확인해 볼 수 있습니다.

—-------------------------------

혹시나 나중을 위해 남겨놓은 초안







윈도우 탐색기에서 exe 파일을 더블 클릭하여 실행할 수도 있지만, 이 경우 출력 콘솔 창이 실행 완료와 함께 사라지므로, 가급적 명령행 도구를 실행한 다음 프로그램을 실행하는 것이 좋습니다.

다음과 같은 결과 가 표시됩니다.

옵션 >

도구 >

솔루션 디렉터리 설명.

디버그/릴리즈 빌드 - 런타임으로 설명하기.

이번에는 온라인 컴파일러에서 실행해 본다. (한번만 설명)

이제 코드를 한줄씩 이해해 봅시다.

Main함수. C언어 프로그램의 시작과 끝.

main함수는 두가지 버전이 있지만, 당분간은 입력이 없는

지시문은 한번에 하나씩. 끝날때마다 세미콜론을 찍어 준다.

printf는 화면에 출력을 하기 위한 함수. stdio.h라는 파일에 정의되어 있음.

왜 헤더 파일을 따로 정의했을까.. 레고 블럭도 연결할 수있는 부분을 미리 알아야 연결할 수 있으니까. 그래서 함수의 모양새를 미리 알려줘야 한다. 그래서 매번 소스코드를 컴파일할 때 알려줄 수 없으니 따로 빼서 #include 로 포함시킨 것.

Retrun

주석. 코드를 작성하는 것은 컴퓨터에 시킬 작업 지시문. 그 작업을 변경하는 것은 결국 사람. 코드를 읽고 바로 의도를 이해할 수 있으면 가장 좋지만 그렇지 않을 경우에는 추가 설명이 필요합니다.

/\* \*/

//

“설명이 필요한 개그는 실패한 개그이다"

하지만 모호할 때는 설명이 필요합니다. -> 클린 코드

빈줄은 실행 파일을 만드는 데는 의미가 없지만, 시각적으로 코드를 분리시켜 주므로 적절히 빈줄로 영역을 나눠 주자.

함수는 작은 함수들로 이루어 진다.

디버깅.

이제 잠시 책을 기존 코드에 “안녕하세요"를 출력해 보자…. 한글도 출력되는 것만 확인. 문자열 인코딩 이런건 나중에 이야기 하자.

한줄씩 실행 -> 로컬 변수 살펴 보기

한줄씩 실행하다간 원하는 부분까지 가다가 죽을꺼야,. 중단점 설정해서 원하는 부분까지 가서 좀 더 살펴보자.

Step over같은 것은 용어만 설명하는 수준에서 넘어갈 것.

—-

드디어 첫번쨰 프로그램을 실행해 보았다. 이제 이 프로그램 골격을 바탕으로 계속 확장시켜 보자.