

학습 목표

- C 프로그램을 작성하기 전에 프로그램을 실행시키는데 필요한 하드웨어와 소프트웨어에 대해서 알아본다.
- 프로그래밍 과정을 생각해 본다.
- 첫 번째 C 언어 프로그램을 작성하고 분석한다.
- C 언어 프로그래밍 도구를 설치하고 사용법을 익힌다.
- 터미널 환경에서 컴파일하는 방법을 알아본다.

목차

- 01 컴퓨터 하드웨어
- 02 운영체제
- 03 프로그래밍이란?
- 04 프로그래밍 과정
- 05 프로그래밍 언어
- 06 C 언어
- 07 첫 번째 C 언어 프로그램 작성
- 08 프로그래밍 개발 환경 구축

01

컴퓨터 하드웨어

1. 컴퓨터 하드웨어

- C 프로그램을 실행하려면 컴퓨터 하드웨어와 그 위에서 실행을 도와주는 운영체제가 필요
- 흔히 볼 수 있는 데스크톱 컴퓨터(Desktop Computer)
- 디스플레이, 키보드, 마우스 같은 입력 장치들이 연결되어 프로그램을 실행



그림 1-1 데스크톱 컴퓨터

1. 컴퓨터 하드웨어

- 컴퓨터의 내부 구조
- 프로세서, 메모리, 보조 기억 장치 등을 시스템 버스로 연결

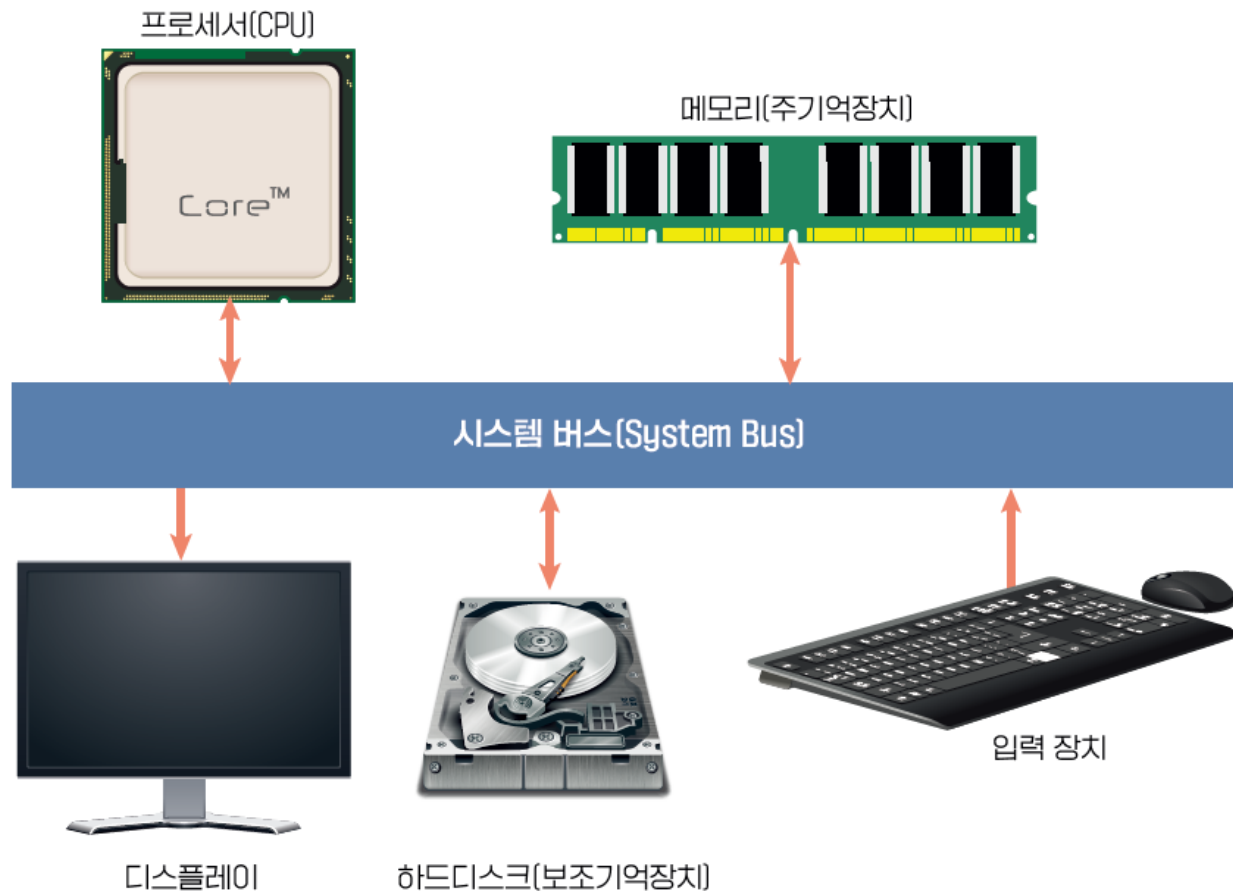


그림 1-2 컴퓨터 하드웨어 시스템 구조

1. 컴퓨터 하드웨어

- 프로그램 실행 과정

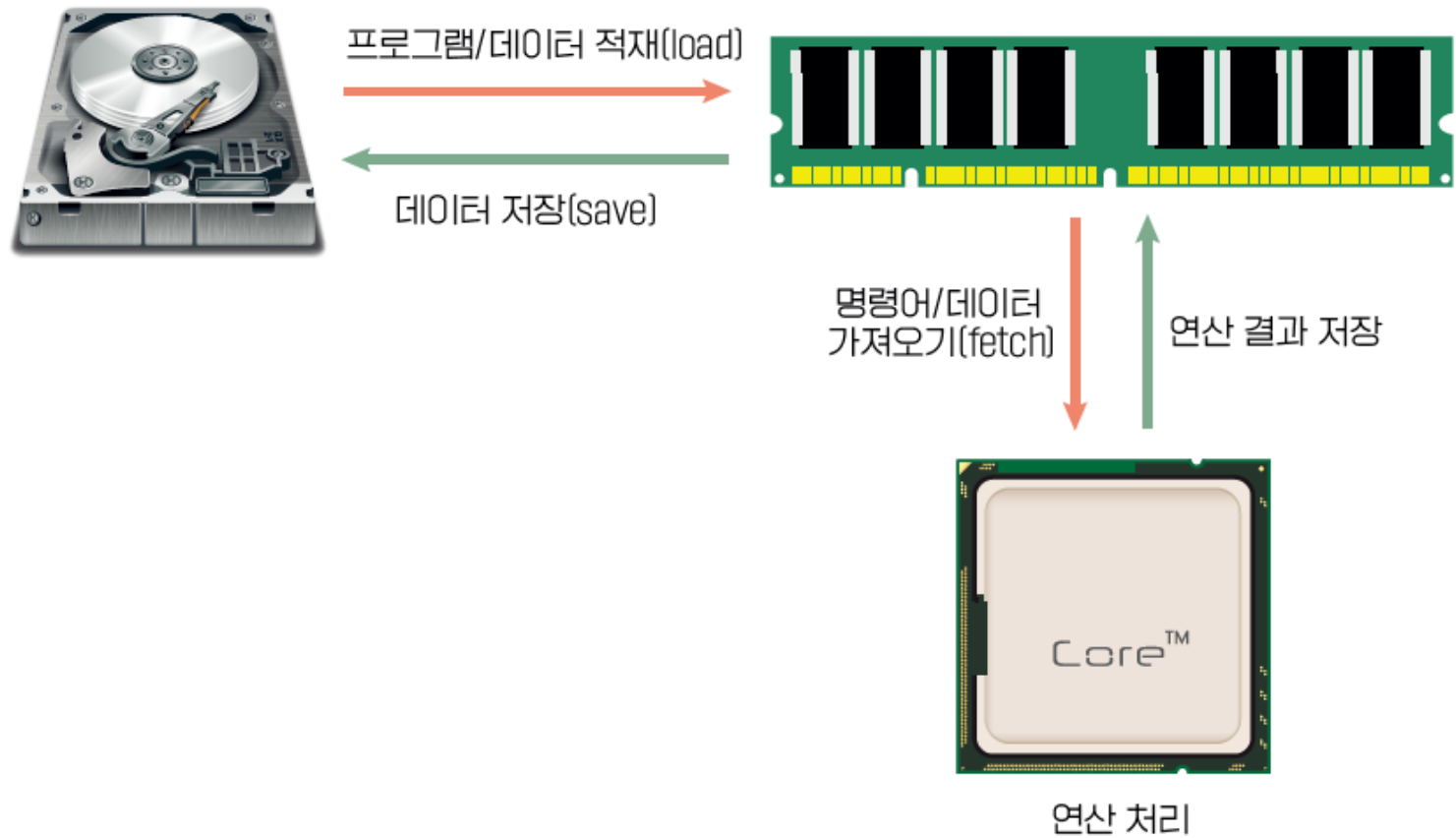


그림 1-3 전처리기에서 프로그램 실행

1. 컴퓨터 하드웨어

■ 프로세서(CPU)

- 프로그램을 실행
- 컴퓨터에 연결된 모든 장치들에게 일을 시키고 결과를 확인하는 등 전반적인 컴퓨팅 작업을 제어
- 프로세서에는 레지스터(Register)라는 아주 작은 양의 메모리들이 있음
- 레지스터는 산술 연산 등에서 임시값이나 결괏값, 현재 실행하는 코드나 메모리 주소 등을 저장

CPU[전처리기]



1. 컴퓨터 하드웨어

■ 메모리(주기억장치)

- 일반적으로 컴퓨터 전원이 종료되면 내용이 사라지는 휘발성 기억 장치
- 컴퓨터는 디지털 시스템으로 0 또는 1의 이진수만 이해 가능
- 메모리나 보조기억 장치에 저장되는 모든 코드와 데이터는 0과 1 형태로 저장
- 0과 1을 표현하는 최소 단위는 '비트(bit)'
- 비트 8개를 묶은 것은 '바이트(byte)'
- 메모리는 바이트 단위로 접근할 수 있도록 주소(address)가 부여됨
- 프로그램의 코드나 데이터는 모두 메모리에 기록

메모리(주기억장치)

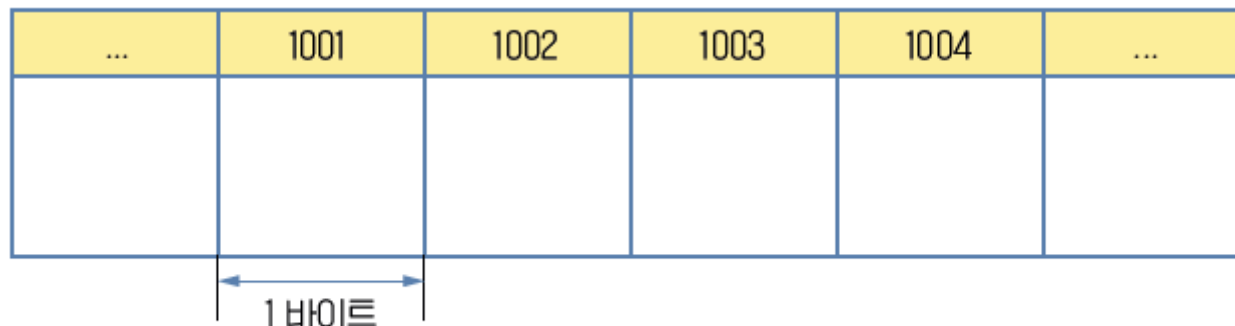


그림 1-4 메모리

1. 컴퓨터 하드웨어

■ 하드디스크(보조기억장치)

- 프로그램들은 컴퓨터가 종료되어도 기록이 보존되는 보조 기억 장치에 저장되어 있음
- 프로그램 실행 시 메모리에 적재
- 대표적인 보조기억장치 : SSD(SolidState Drive), CD-ROM과 DVD-ROM 같은 ODD(Optical Disk Drive), USB 메모리 등 포함



하드디스크(보조기억장치)

1. 컴퓨터 하드웨어

■ 시스템 버스

- 프로세서나 메모리라는 장소를 연결하는 도로
- 프로세서와 메모리, 메모리와 다른 보조기억장치 또는 입력 장치들을 연결해서 데이터를 주고받는 이동 통로
- 모든 데이터는 시스템 버스를 통해서 이동

02

운영체제

2. 운영체제

- 하드웨어와 사용자 프로그램 사이에 위치한 소프트웨어
- 프로세서, 메모리, 입출력 장치 같은 하드웨어를 관리
- 사용자가 컴퓨터를 쉽게 사용하도록 도움
- 프로그램을 메모리에 적재하고 실행
- 보조기억장치에 파일이나 디렉토리 등을 생성하고 관리

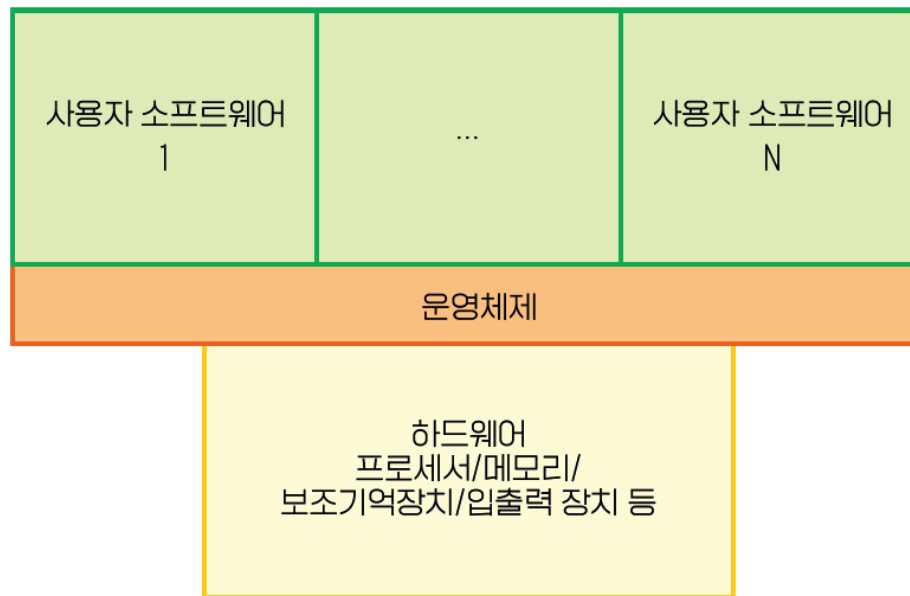


그림 1-5 운영체제와 하드웨어, 사용자 프로그램 간의 관계

03

프로그래밍이란?

3. 프로그래밍이란?

- 프로그래밍이란 '문제를 해결'하는 것
- 포괄적으로 '문제 해결을 위해 계획을 세우고, 해결 방안을 마련하고 진행하는 과정'을 의미
- 여기서 말하는 프로그래밍이란 '컴퓨터에 의한 문제 해결'
- 컴퓨터 프로그래밍은 컴퓨터에서 동작하는 소프트웨어를 만든다는 의미로 '소프트웨어 개발'이라고도 함

04

프로그래밍 과정

4. 프로그래밍 과정

- 문제 해석과 요구사항 분석
 - 문제를 분석해서 무엇을 해야 하는지 파악
 - 사용자로부터 받아야 하는 입력 내용이 무엇이고, 무엇을 처리할 것이며, 결과가 무엇이고 어떻게 출력할지 결정
- 프로그램 설계
 - 문제 해결 방법을 계획하고 구성
 - 요구사항에 맞춰 어떤 방법을 선택할 것인지 결정
 - 문제를 해결하는 방법을 알고리즘(Algorithm)이라고 함
- 코딩
 - 정해진 문제 해결 방법을 컴퓨터가 이해할 수 있는 언어로 표현
- 검수(테스트)
 - 제대로 동작하는지 확인하는 과정
 - 생각했던 결과가 제대로 나오는지 확인
 - 검수-설계-코딩 또는 검수-코딩 과정을 반복하면서 프로그램 완성

05

프로그래밍 언어

5. 프로그래밍 언어

■ 프로그래밍 언어

- 사람이 컴퓨터와 의사소통을 하기 위한 수단
- 컴퓨터의 프로세서는 자기만의 독자적인 기계를 사용
- 프로세서가 달라지면 기계어가 달라져 의사소통 불가능한 경우 다수
- 사람이 이진수 형태의 기계를 기억하고 프로그래밍 하는 것은 아주 어려움
- 인텔 32 비트 CPU의 기계어로 작성된 $1+2+3+\dots+100$ 의 합을 구해 반환하는 코드의 일부를 16진수 형태로 표현한 것

코드 1-1 합을 구해서 반환하는 코드의 기계어

```
55 48 89 e5 48 83 ec 10 c7 45 fc 00 00 00 00 c7 45 f8 00 00 00 00 eb 0a 8b 45 f8  
01 45 fc 83 45 f8 01 83 7d f8 64 7e f0 8b 45 fc 48 83 c4 10 5d c3
```

5. 프로그래밍 언어

■ 어셈블리 언어

- 기계어의 대안으로 만들어진 언어
- 이진수로 구성된 기계어 명령 코드를 사람이 좀 더 기억하기 쉬운 니모닉 (Mnemonic) 형태로 대응

코드 1-2 어셈블리 코드

```
movl    $0x0,-0x4(%rbp)
movl    $0x0,-0x8(%rbp)
jmp     22 <sum+0x22>
mov     -0x8(%rbp),%eax
add     %eax,-0x4(%rbp)
addl    $0x1,-0x8(%rbp)
cmpl    $0x64,-0x8(%rbp)
jle     18 <sum+0x18>
ret
```

5. 프로그래밍 언어

■ 고급 프로그래밍 언어

- 사람들이 쉽게 배우고 사용할 수 있도록 만든 언어
- 어셈블리 언어의 니모닉보다 사람이 이해하기 쉬운 단어들과 문법으로 구성
- 사람이 사용하는 일반 언어에 비해 문법 체계가 단순해서 쉽게 학습 가능
- 프로세서가 달라도 동일한 사용법을 제공
- 다양한 컴퓨터에서 사용하는 프로그램 작성 가능

코드 1-3 C 언어로 작성한 합을 구하는 코드

```
int sum(void)
{
    int sum = 0;
    for (int i = 0; i <= 100; i++) {
        sum += i;
    }
    return sum;
}
```

5. 프로그래밍 언어

■ 컴파일 언어

- 일반적으로 C 언어는 컴파일(compile) 과정을 통해 기계어로 직접 번역되고 실행
- 컴파일러(Compiler)는 한 언어를 다른 언어로 변환하는 소프트웨어

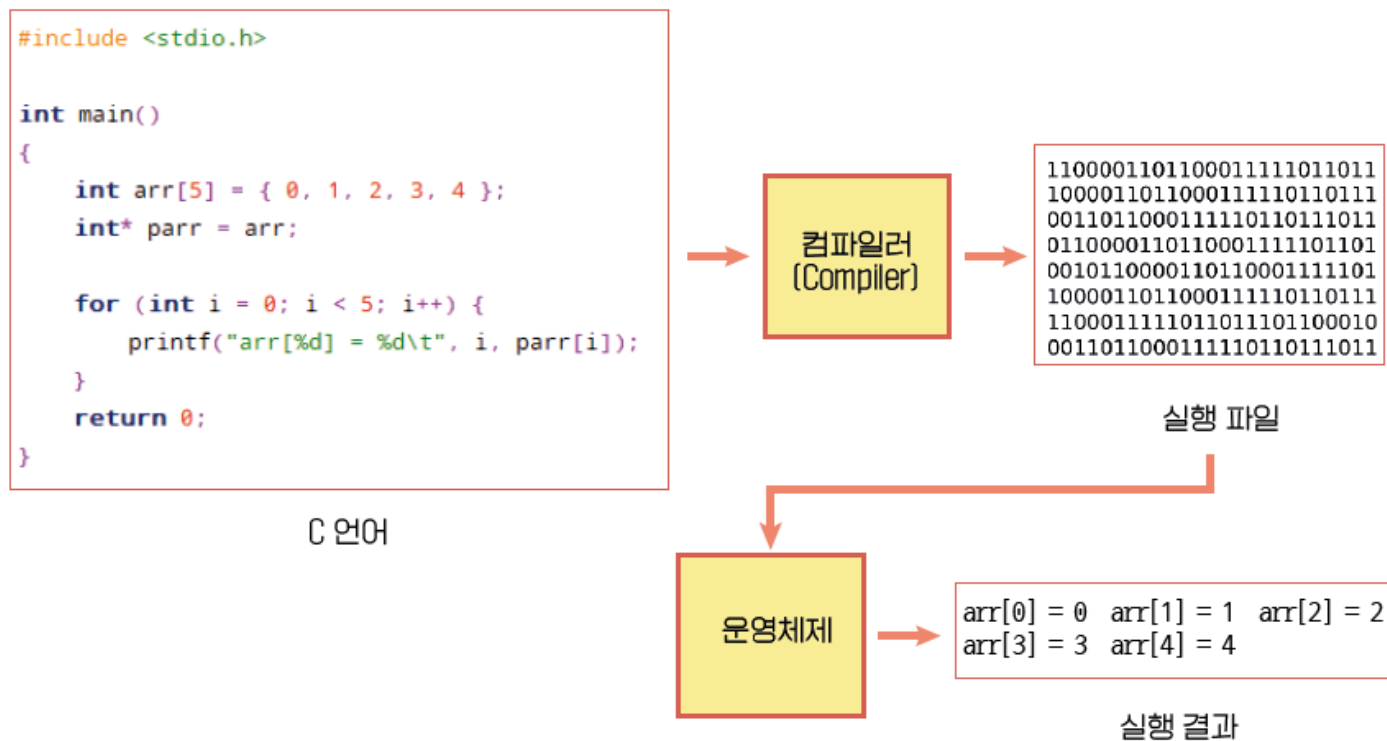


그림 1-6 컴파일 언어가 실행되는 과정

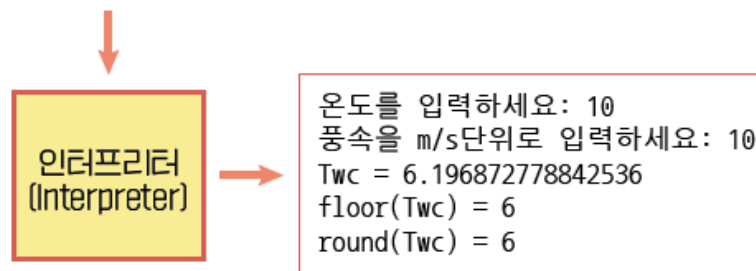
5. 프로그래밍 언어

■ 인터프리터 언어

- 실시간으로 한 문장씩 통역(Interpret)해서 실행
- 파이썬(Python) 언어가 대표적인 인터프리터 언어

```
import math

temp = float(input("온도를 입력하세요: "))
wind = float(input("풍속을 m/s단위로 입력하세요: "))
v = 3.6 * wind
Twc = 13.12 + 0.6215 * temp - 11.37 * (v ** 0.16) + 0.3965 * temp * (v ** 0.16)
print("Twc =", Twc)
print("floor(Twc) =", math.floor(Twc))
print("round(Twc) =", round(Twc))
```



실행 결과

그림 1-7 인터프리터 언어가 실행되는 과정

06

C 언어

6. C 언어

- 시스템 프로그램에서부터 일반 응용 프로그램까지 개발할 수 있는 범용 프로그래밍 언어
- 작고, 배우기 쉽고, 이식성(portability)이 높은 언어
- 70년대 초반에 처음 만들어진 이후 약 50여 년간 사용됨
- 축적된 정보와 코드도 많고 사용하는 사람도 많음

6. C 언어

■ ANSI C

- 1989년에 미국에서 표준화
- 1990년에 ANSI C를 국제 표준화 기구 ISO 표준으로 채택
- 그래서 C89/C90으로 불리기도 함
- C 언어의 문법을 좀 더 명확하고 시스템에 영향을 받지 않게 변경
- 표준 라이브러리가 정의되어 라이브러리에 있는 것들만 사용한다면 시스템이 달라져도 동일한 C 코드 사용 가능

6. C 언어

■ C99와 이후 버전

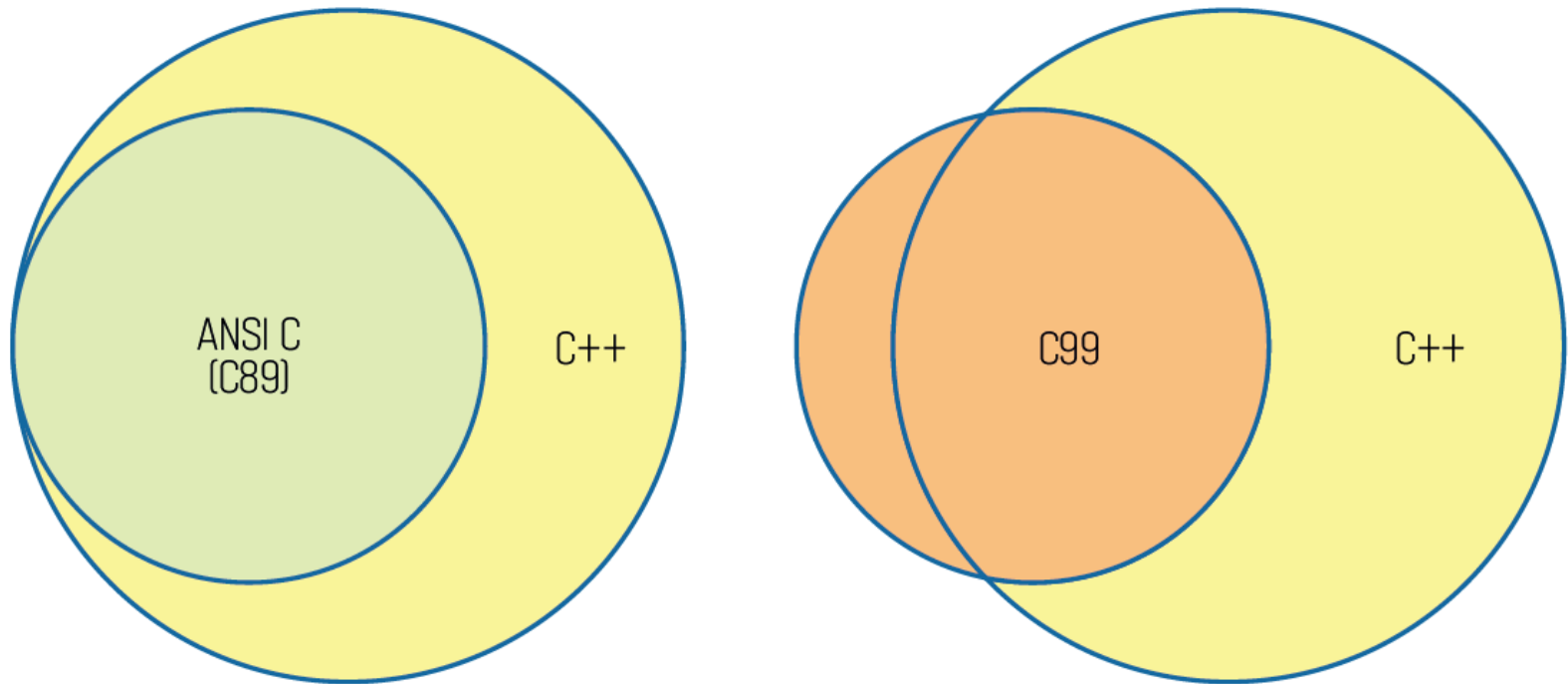


그림 1-8 C89, C99, C++의 상관관계

6. C 언어

■ C 언어의 영향을 받은 언어

■ C++

- 벨 연구소에서 반 스트라우스트럽이 C 언어에 객체 지향 언어의 특성을 붙여 만듦
- ANSI C 언어를 부분 집합으로 여길 정도
- C 언어에 증가 연산자인 ++를 붙였다고 알려져 있어 더 나은 C 언어라고 불리기도 함

■ 자바/C#

- 객체 지향 언어들이라서 C보다는 C++에서 영향을 받았다고 하는 편이 정확
- 기본적인 변수, 조건문 반복문, 함수 개념 등이 거의 C 언어와 유사
- C++, Java, C# 등을 C 언어 계열 언어라고 부르기도 함

6. C 언어

■ 앞으로 다룰 C 언어

- C89와 C99 사이에 있는 'C 언어'를 설명
- C99 또는 이후 버전을 제대로 지원하는 것은 오픈 소스 C 컴파일러인 GCC 계열
- 마이크로소프트사의 윈도우 운영체제에서 많이 사용하는 비주얼 스튜디오는 C99를 제대로 지원하지 않음
 - 하지만 C99의 일부 기능을 확장 형태로 사용할 수 있도록 함(주로 이 내용을 다룰 예정)
 - 일부 내용은 비주얼 스튜디오에서 사용할 수 없다고 명시함

6. C 언어

■ C 언어의 특징과 장단점

■ 장점

- 저수준 언어
- 간결한 문법
- 이식성
- 성능
- 강력한 메모리 조작 기능
- 다양한 라이브러리

■ 단점

- 낮은 추상화
- 메모리 관리 오류
- 보안 취약성
- 코드 가독성

07

첫 번째 C 언어
프로그램 작성

7. 첫 번째 C 언어 프로그램 작성

■ 명령문

- 구문(statement)이라고 부르기도 하며 프로그램의 실행 단위
- C 언어 프로그램을 실행한다는 것은 명령문들을 순차적으로 실행하는 것
- 대입문, 조건문, 반복문처럼 ~문 형태로 끝나는 용어들이 명령문의 종류를 나타냄
- 빈 명령문도 작성 가능

코드 1-4

```
1  int n = 0;
2  if (n > 0)
3      n -= 1;
4  printf("n = %d\n", n);
5  ;
```


7. 첫 번째 C 언어 프로그램 작성

■ 코드 블록

- 중괄호({ }) 사이에 0줄 이상의 명령문을 넣은 것
- 코드 블록에 있는 명령문들은 한 개씩 순차적으로 실행
- 빈 명령문처럼 명령문이 없는 빈 코드 블록(empty code block)을 구현하는 것도 가능

코드 1-5

```
1  int func()  
2  {  
3      int n = 0;  
4      if (n > 0) {  
5          n -= 1;  
6      }  
7      printf("n = %d\n", n);  
8  
9      {  
10         int n = 0;  
11         {  
12             printf("n = %d\n", n);  
13         }  
14     }  
15     if (n > 0) {}  
16 }
```

7. 첫 번째 C 언어 프로그램 작성

■ 코드 블록

- 일반 명령문과는 다르게 코드 블록은 세미콜론을 붙이지 않음
- 닫는 중괄호(`}`)만으로 충분

코드 1-6

```
1  if (n > 0) {  
2      printf("n = %d\n", n);  
3  };
```

1줄부터 3줄의 }까지 한 개의 명령문으로 처리되고, ;는 빈 명령문으로 따로 취급

7. 첫 번째 C 언어 프로그램 작성

■ 띄어쓰기와 들여쓰기

- 식별자
 - 프로그래머가 코드에서 사용하기 위해 붙이는 이름
- 키워드
 - C 언어에서 사용하려고 미리 지정한 단어들

7. 첫 번째 C 언어 프로그램 작성

■ 띄어쓰기

- int는 키워드
- a는 식별자
- ;(세미콜론)은 명령문의 끝

코드 1-7

```
int a;
```

- 띄어쓰기는 한 칸 이상만 있으면 됨
- 줄을 바꾸어 작성해도 동일

코드 1-8

```
int  
a   ;
```

7. 첫 번째 C 언어 프로그램 작성

■ 띄어쓰기

- 코드 1-6을 띄어쓰기 없이 한 줄에 붙여도 동일하게 실행됨

코드 1-6

```
1  if (n > 0) {  
2      printf("n = %d\n", n);  
3  };
```

```
if(n>0){printf("n = %d\n",n);};
```

7. 첫 번째 C 언어 프로그램 작성

■ 들여쓰기

- 앞줄에서 일정한 칸의 수만큼 안쪽으로 댄 것
- 일부 명령문들과 코드 블록에서 사용
- 들여쓰기도 미리 정해진 규칙은 없고, 코드 실행에 영향을 미치지 않음
- 코드를 읽기 쉽게 공백 네 칸 사용(본 책에서 권장하는 들여쓰기)

코드 1-9

```
1  if (n > 0)
2      n -= 1;
3
4  if (n > 0) {
5      n -= 1;
6  }
7
8  int f()
9  {
10     int n;
11     n = 2 * 3;
12 }
```

들여쓰기 된 부분

코드 1-10

```
1  if (n > 0) n -= 1;
2  if (n > 0) { n -= 1; }
3  int f() { int n; n = 2 * 3; }
```

짧은 코드는 줄을 바꾸지 않고 붙여쓰는 경우도 있음

오류는 발생하지 않지만, 코드의 가독성이 떨어짐
줄을 바꿔 사용하는 편이 좋음

7. 첫 번째 C 언어 프로그램 작성

■ 주석

■ 주석문을 사용하는 예시

- 작성자 정보나 버전 정보를 보임. 버전 정보는 프로그램 개발 단계를 나타내는 기호로 주로 숫자 조합을 사용
- 문제 해결 방법(알고리즘)을 설명
- 프로그래머가 코드를 작성하면서 해야 할 일이나 수정해야 할 내용을 작성
- 다른 사람과 협업하는 경우 다른 사람에게 코드를 어떻게 사용해야 하는지 설명을 붙이기도 함

7. 첫 번째 C 언어 프로그램 작성

■ 주석

■ 주석을 붙이는 방법

- //로 시작 (C99에서 지원하는 내용. 비주얼 스튜디오에서 사용 가능)

코드 1-11

```
int a = 3;    // 주석 시작. 변수 a를 생성하고 3을 저장
// int b = 4;
```

- /*와 */ 사이에 주석 내용 추가(기존 C 언어 방법. 최신 버전에서도 사용 가능)

코드 1-12

```
/* 한 줄로 만드는 주석 */
int a = /*5*/ 3;
/* 여러 줄로 만들어진
주석 */
```


7. 첫 번째 C 언어 프로그램 작성

■ 최소한의 C 코드

- C 언어로 작성하는 프로그램의 최소 코드

코드 1-13

```
1  int main(void)
2  {
3      /* 실행할 코드 */
4      return 0;
5  }
```

- 프로그램이 실행되는 과정

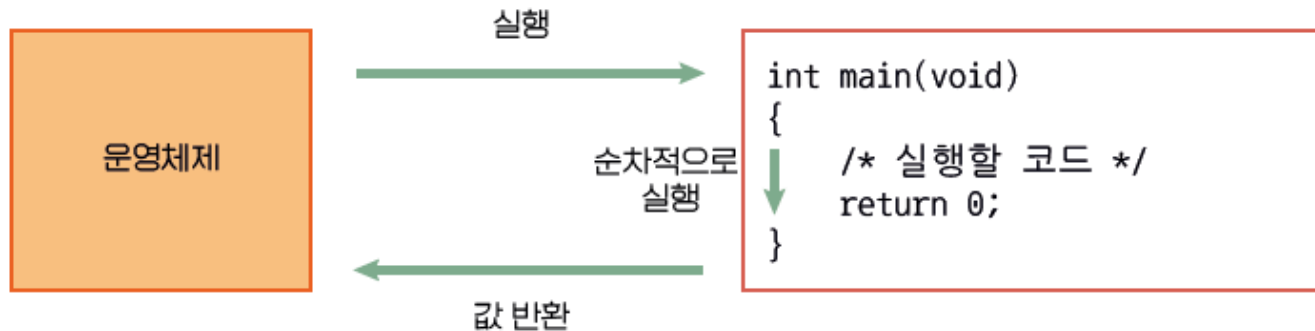


그림 1-10 운영체제에서 C 프로그램 실행

7. 첫 번째 C 언어 프로그램 작성

■ 첫 번째 C 코드 작성

- 화면에 "My first C program"이라는 문자열을 출력하는 C 코드

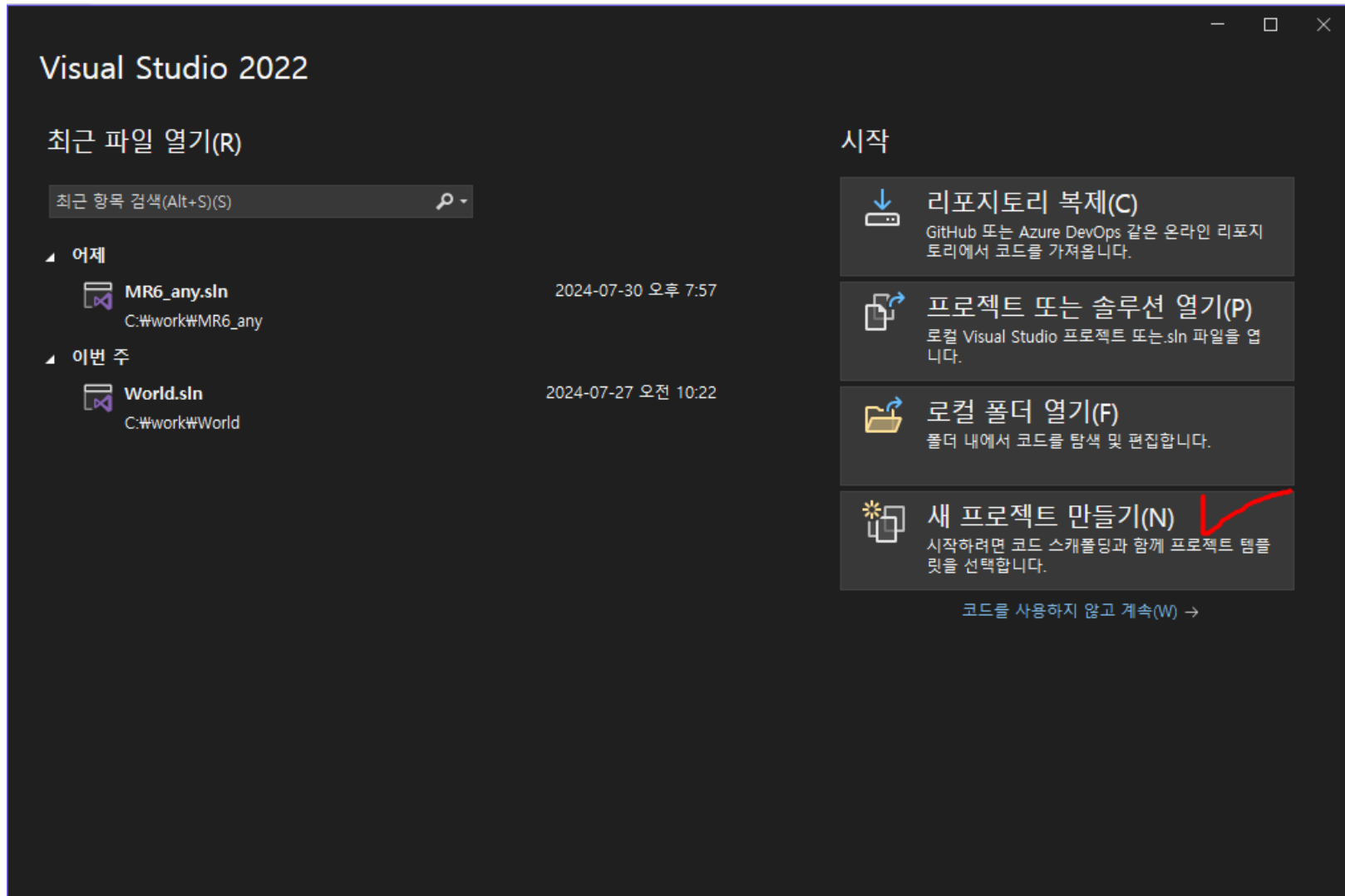
코드 1-14 First.c

```
1  /* 첫 번째 C 언어 프로그램 */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("My first C program"); // 문자열 출력하는 printf() 함수 호출
7      return 0;
8  }
```

VS community 설치

- <https://visualstudio.microsoft.com/ko/vs/community/>
- 다운로드 설치 진행
- c& c++ 관련 컴포넌트 확인 및 설치 완료

VS community 실행



VS community 실행

새 프로젝트 만들기

최근 프로젝트 템플릿(R)

빈 프로젝트

C++

콘솔 앱

Visual Basic

템플릿 검색(Alt+S)(S)



모두 지우기(C)

C++

모든 플랫폼(P)

모든 프로젝트 형식(T)



빈 프로젝트

Windows용 C++를 사용하여 처음부터 시작합니다. 시작 파일을 제공하지 않습니다.

C++

Windows

콘솔



콘솔 앱

Windows 터미널에서 코드를 실행합니다. 기본적으로 "Hello World"를 출력합니다.

C++

Windows

콘솔



CMake 프로젝트

.sln 또는 .vcxproj 파일에 종속되지 않은 최신 플랫폼 간 C++ 앱을 빌드하세요.

C++

Windows

Linux

콘솔



Windows 데스크톱 마법사

마법사를 사용하여 고유한 Windows 앱을 만드세요.

C++

Windows

데스크톱

콘솔

라이브러리



Windows 데스크톱 애플리케이션

Windows에서 실행되는 그래픽 사용자 인터페이스를 사용하는 애플리케이션용 프로젝트입니다.

C++

Windows

데스크톱



Cocos

다중 플랫폼 게임 작성에 사용되는 Cocos 엔진입니다.

설치 필요

뒤로(B)

다음(N)

VS community 실행

새 프로젝트 구성

빈 프로젝트

C++

Windows

콘솔

프로젝트 이름(I)

Project2

위치(L)

C:\Users\mymy\source\repos

솔루션 이름(M) ⓘ

Project2

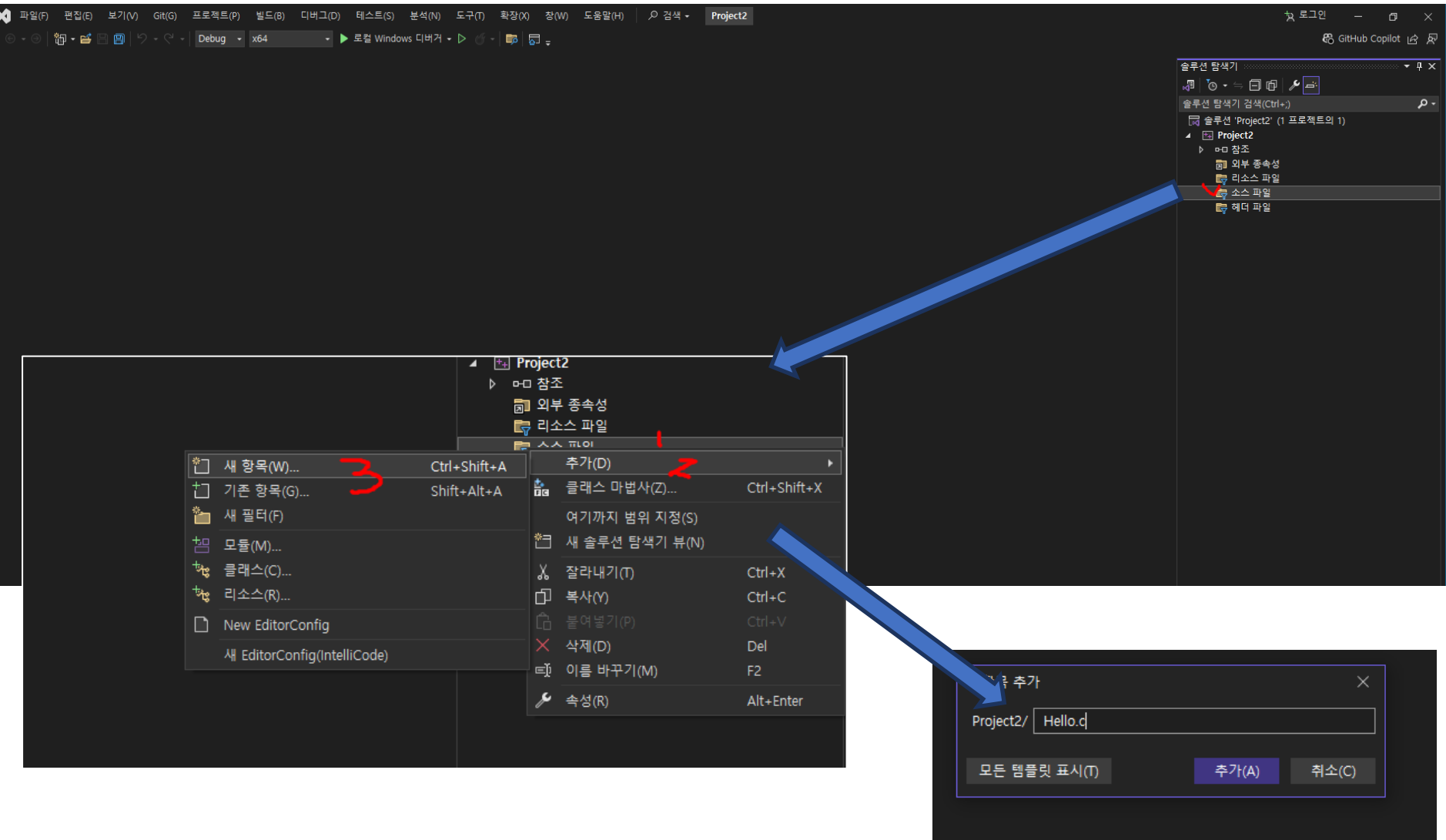
☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

"C:\Users\mymy\source\repos\Project2\Project2"에 프로젝트이(가) 만들어집니다.

뒤로(B)

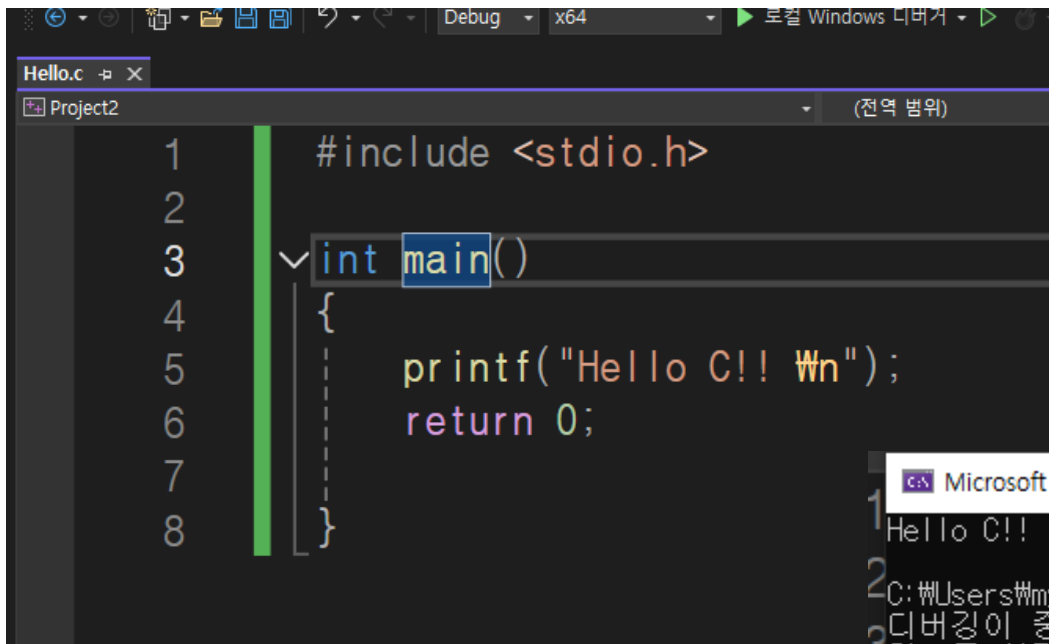
만들기(C)

VS community 실행

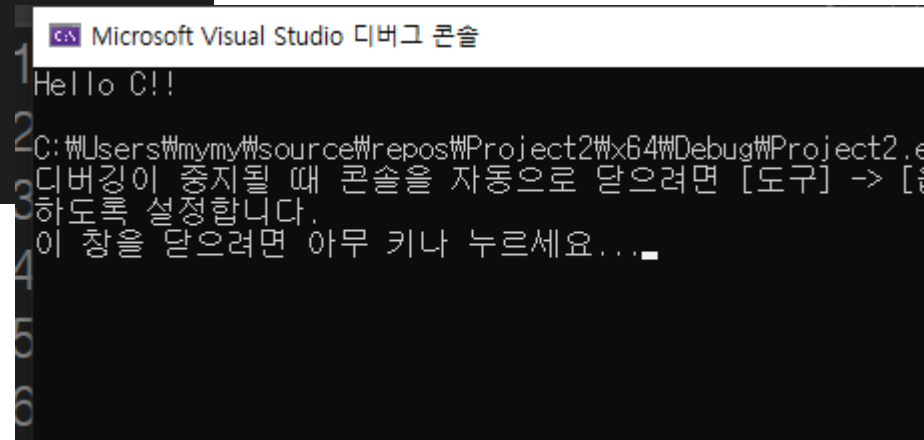


VS community 실행

- ctrl_+shift +b → build
- F5 → run



```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello C!! \n");
6      return 0;
7  }
8
```



```
Microsoft Visual Studio 디버그 콘솔
1 Hello C!!
2 C:\Users\mymy\source\repos\Project2\x64\Debug\Project2.exe
3 디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [디버그 콘솔]
4 하도록 설정합니다.
5 이 창을 닫으려면 아무 키나 누르세요....
```

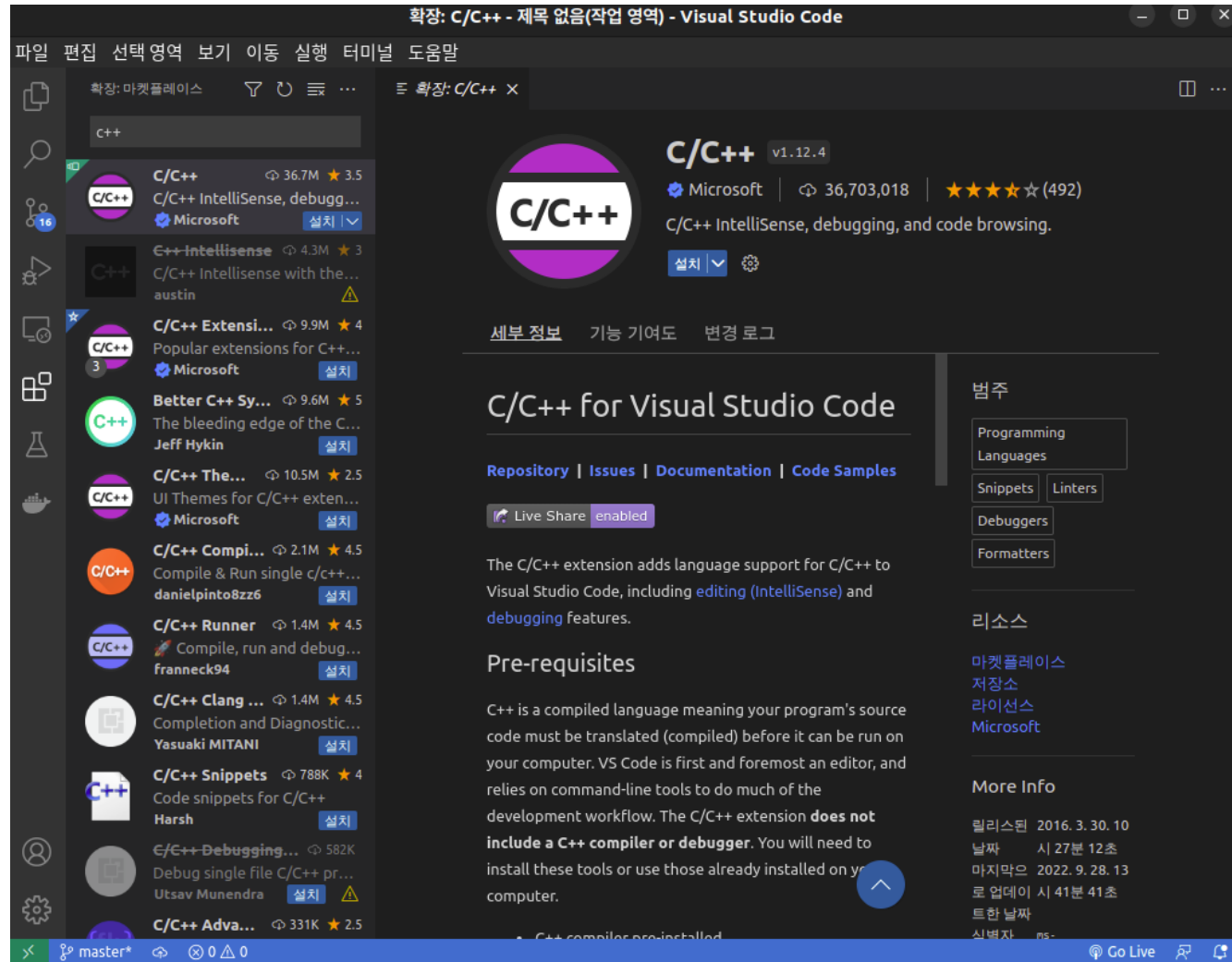

UBUNTU 환경

Gnu c 컴파일러 설치

```
sudo apt update && sudo apt upgrade -y  
sudo apt install gcc g++ clang gdb -y
```

UBUNTU 환경

c++을 검색해서 설치

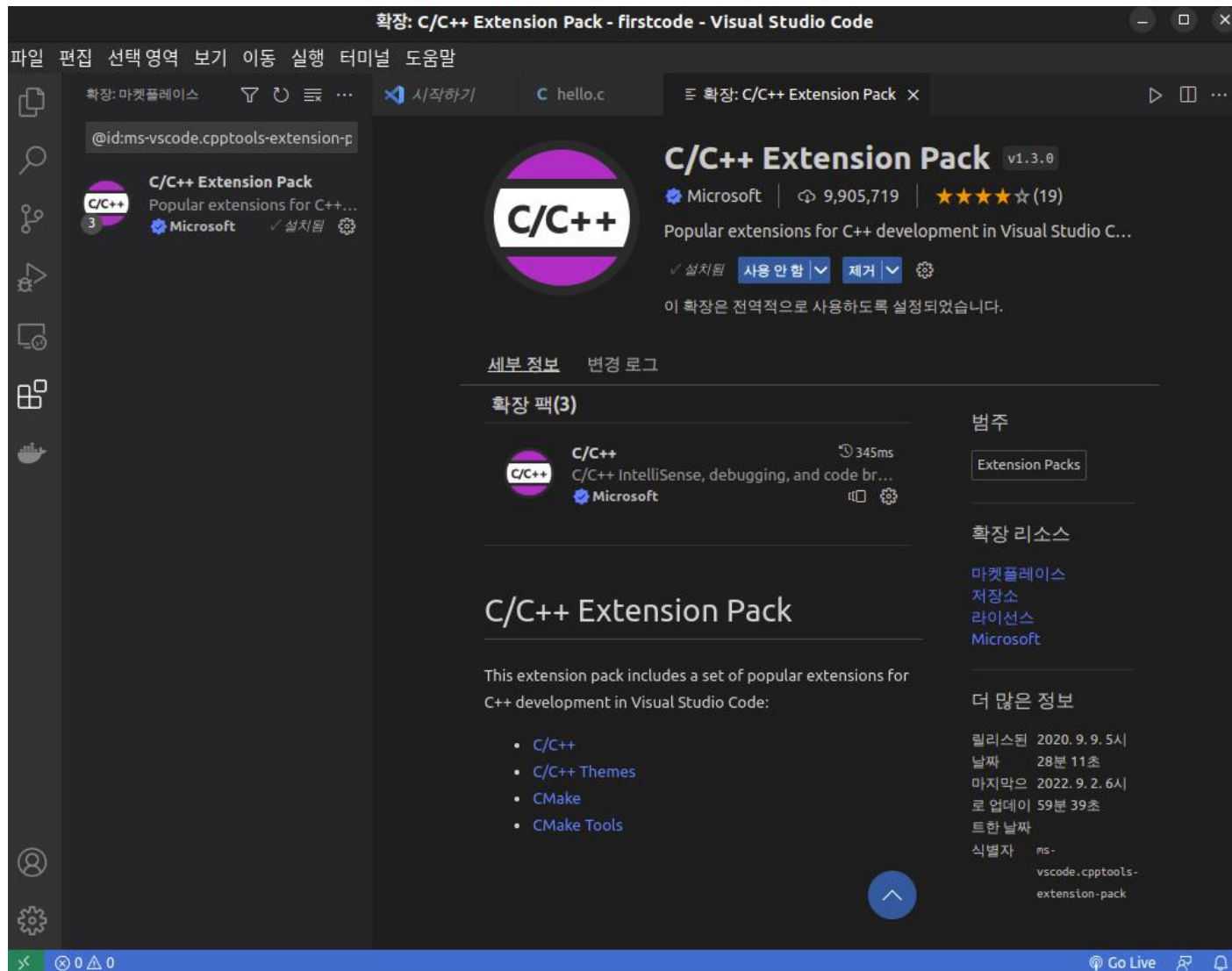


UBUNTU 환경

code runner라고 검색해서 c로 작성한 코드를 쉽게 실행 할 수 있는 확장 프로그램도 설치

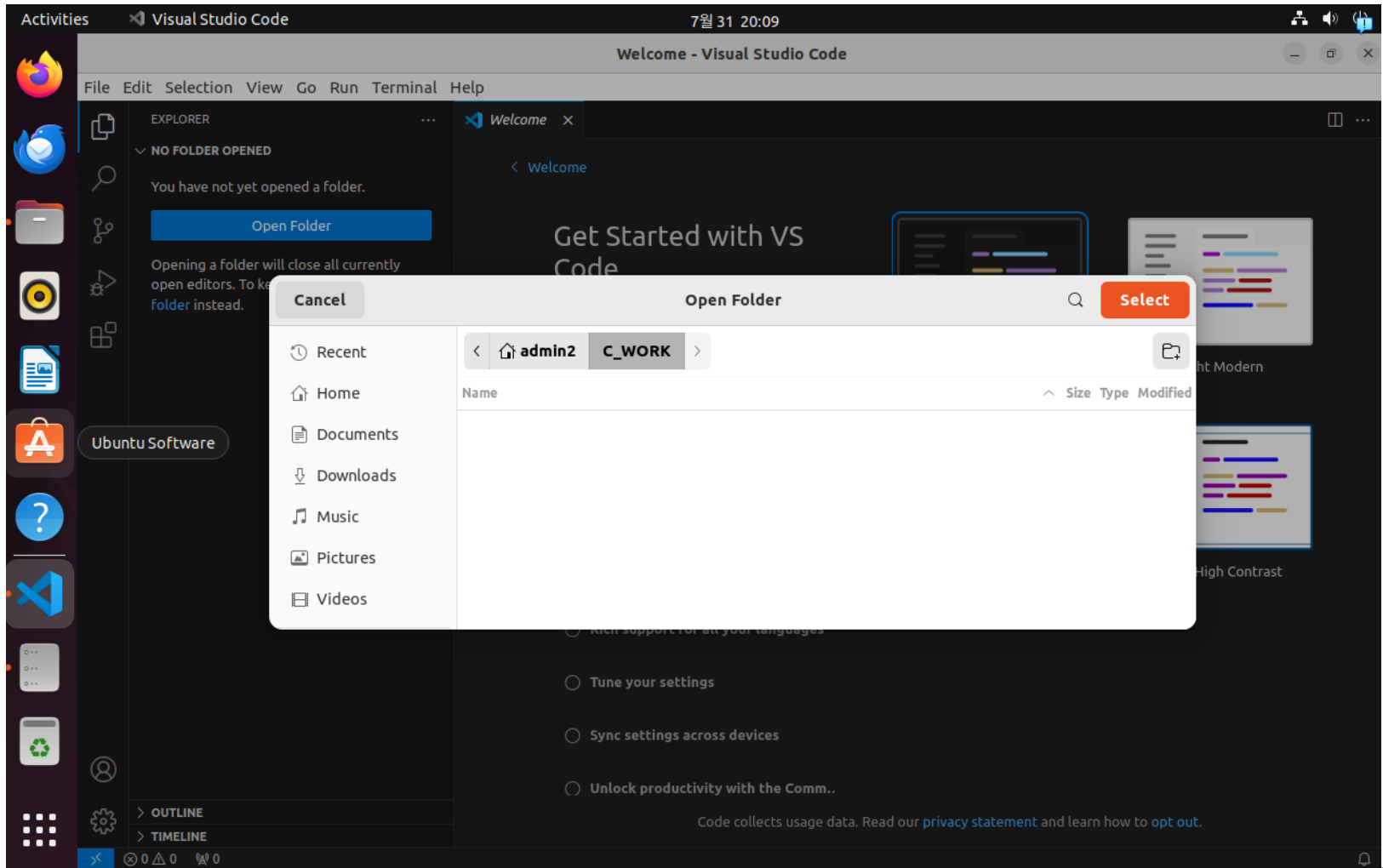


UBUNTU 환경



UBUNTU 환경

■ Code 작업 공간 정의

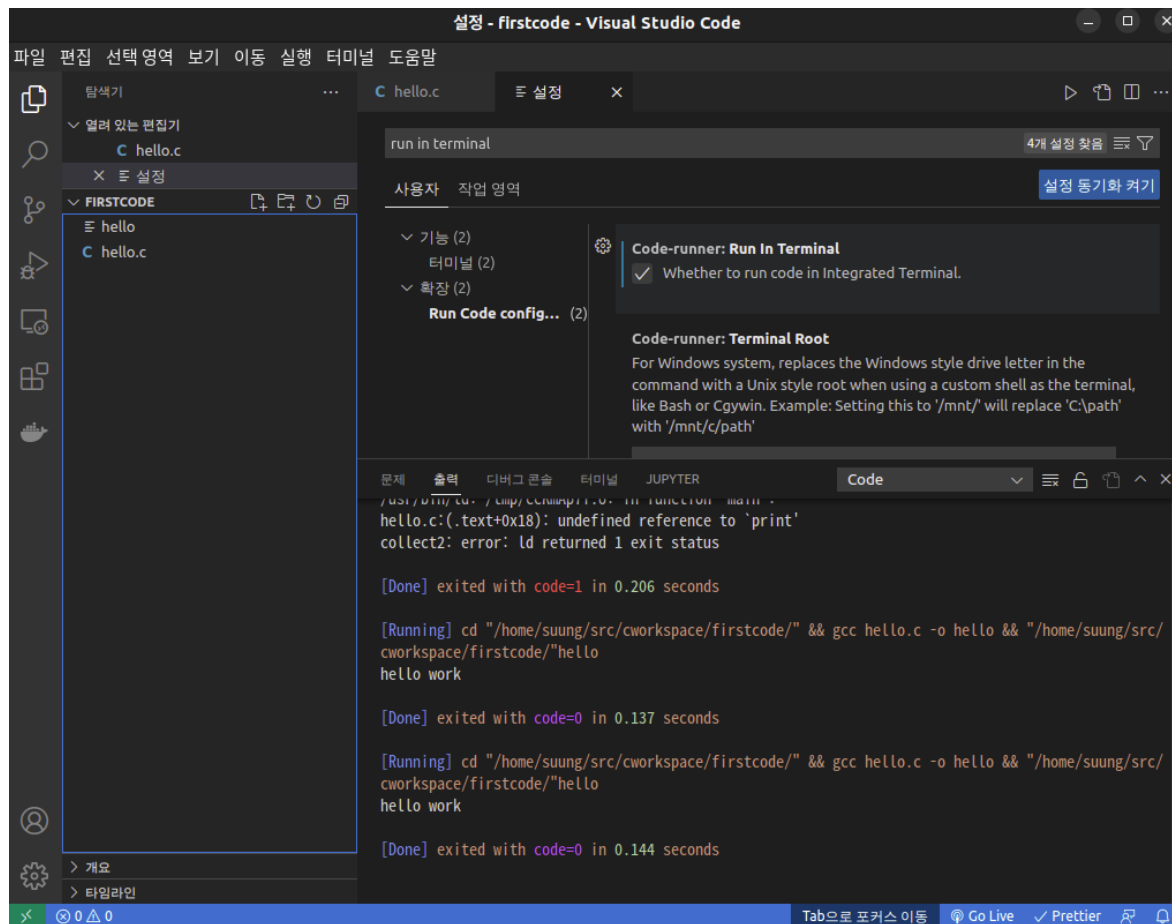


UBUNTU 환경

확장 프로그램 설치가 끝나고 나서
[파일] - [기본 설정] - [설정] 메뉴로 이동

검색할 수 있는 부분에서 run in terminal을 검색

Whether to run code in integrated Terminal 부분을 체크해서 터미널에서 실행되도록 설정합니다.



코드 1-14 First.c

```
1  /* 첫 번째 C 언어 프로그램 */
2  #include <stdio.h>
3
4  int main(void)
5  {
6      printf("My first C program"); // 문자열 출력하는 printf() 함수 호출
7      return 0;
8  }
```

08

프로그래밍 개발 환경 구축

8. 프로그래밍 개발 환경 구축

■ C 프로그래밍 과정

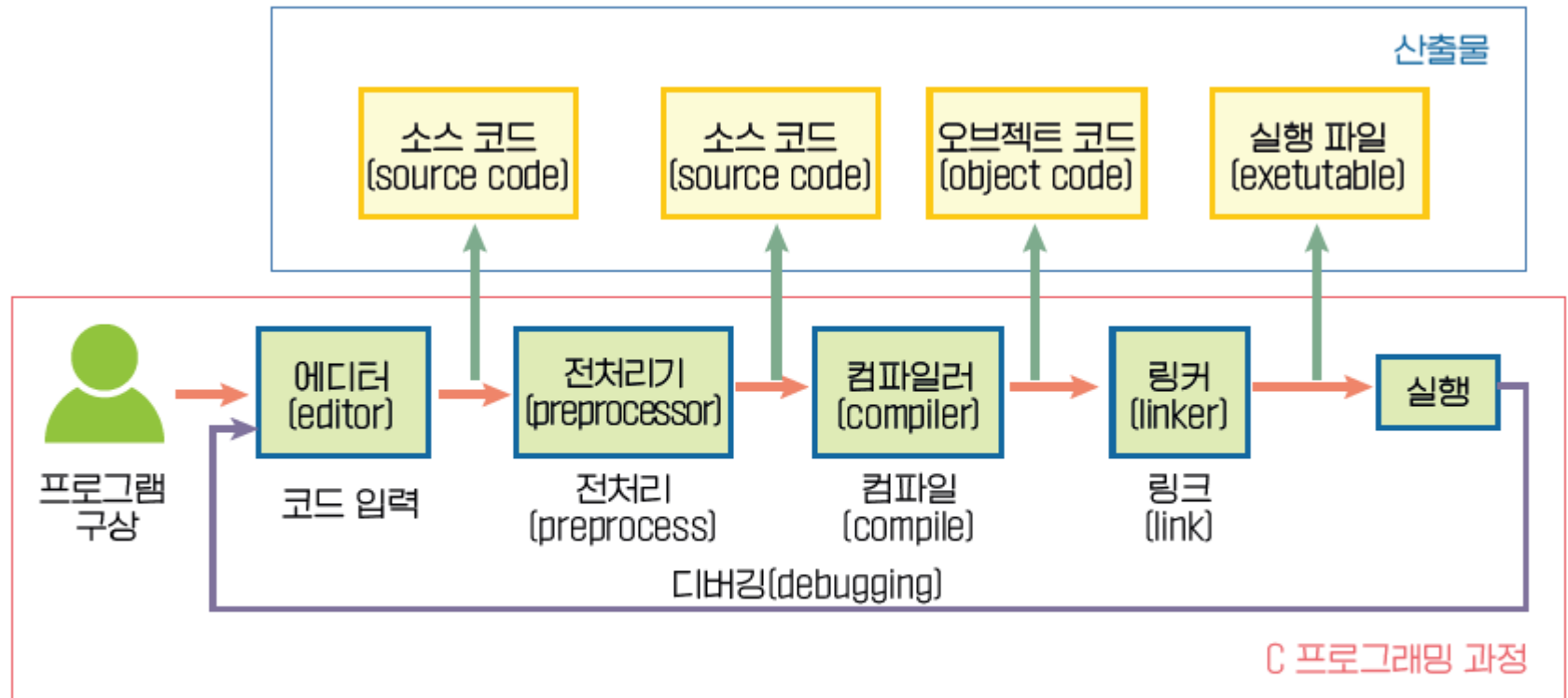


그림 1-11 C 프로그래밍 과정

8. 프로그래밍 개발 환경 구축

■ 통합 개발 환경

- 에디터, 컴파일러, 링커, 디버거 등을 한 개 프로그램에 집약시킨 것
- 프로그래머가 한 개의 프로그램에서 코드 작성, 실행, 실시간 디버깅할 수 있도록 만든 프로그램

표 1-1 통합 개발 환경 비교

통합 개발 환경	지원 플랫폼(운영체제)	비용
Code::Blocks	윈도우, 맥 OS, 리눅스	무료
CLion	윈도우, 맥 OS, 리눅스	상용
비주얼 스튜디오	윈도우	무료/상용
Dev C++	윈도우	무료

8. 프로그래밍 개발 환경 구축

■ 개발 도구 설치(책의 부록에서 설명)

■ 윈도우

- 오픈 소스 컴파일러인 GCC 계열 컴파일러
- 마이크로소프트 사의 비주얼 스튜디오(Visual Studio)

■ 맥

- 엑스코드(Xcode)

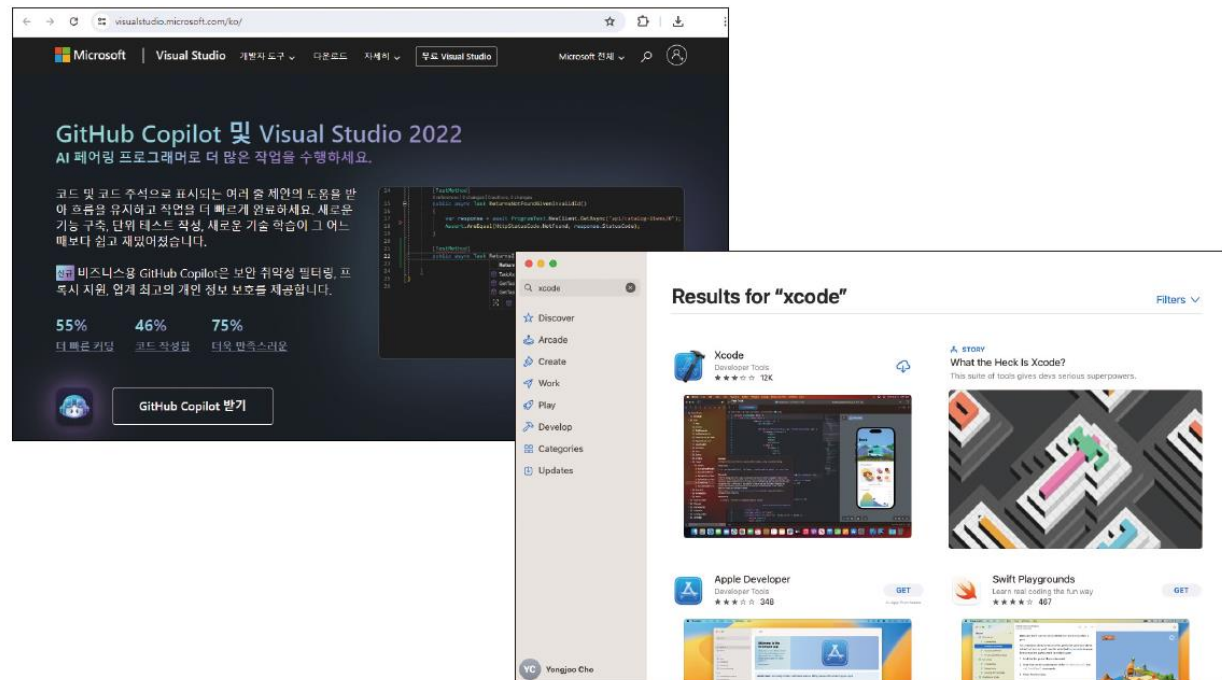


그림 1-12 비주얼 스튜디오(좌)/엑스코드(우)