



C언어

C언어

Visual Studio 2019



C언어
Hello World!



- N회차 형식으로 진행될 것입니다.
 - 1회차가 끝나면 더 많은 정보로 2회차가 나오고 그게 반복되는 구조입니다.
 - 회차간의 난이도는 꽤 높은 정도의 벽입니다.
 - 회차가 커질수록 정보의 양이 기하급수적으로 늘어날 것입니다.
- 많은 정보의 양을 한번에 얻기는 매우 매우 어렵고 천재나 컴퓨터배경지식이 있는 사람을 제외하면
 - 대부분의 사람들은 어려울 것이니 회차를 나누었습니다.
- 저번 회차의 정보가 다음 회차의 정보랑 합쳐져서 점점 더 많은 지식을 탐처럼 쌓을 수 있습니다.

CONTENT

1 About C언어

제작자 및 기본설명

2 Visual Studio 2019

환경설정 및 환경설명

3 C언어 시작

새프로젝트 및 기초문법

4 변수

int float char

5 진법/진수

16진수 10진수 2진수

6 삼항 연산자 if switch while for

조건식

About C언어

제작자 및 기본설명

- C언어는 1972년 탄생
 - 데니스 리치(Dennis Ritchie)가 유닉스(UNIX) 사용하기위한 B언어(제작자: 켄 톰슨)의 개선판
 - 1969년에 초기 유닉스는 대부분 어셈블리어로 작성되어서 하드웨어가 바뀌면 새로 개발하는 문제
 - 그래서 데니스리치가 하드웨어 상관없이 사용할 수 있는 언어인 C언어를 개발





데니스 리치

켄 톰슨



- 장점 시스템 프로그래밍이 가능합니다.
 - (OS, Operating System, 운영체제)를 개발하기 위한 언어임으로 하드웨어 제어
- 이식성을 가진 프로그램을 만들 수 있다.
 - (다른 컴퓨터에 이식이 가능하고 표준을 지켜 개발하면 다른 컴파일러도 가능)
- 함수를 사용해서 개별 기능을 구현할 수 있다.
 - (기능별로 프로그래밍이 가능해서 에러 수정 및 유지보수가 좋고 잘 만들어진 함수는 코드 재활용이 가능하다)





- C -> C++ -> C#
 - A->B->C순으로 언어가 생기다가 C++이 탄생했는데 C#은

C++ C++ C#



Mar 2021	Mar 2020	Change	Programming Language	Ratings	Change
1	2	⬆	C	15.33%	-1.00%
2	1	⬇	Java	10.45%	-7.33%
3	3		Python	10.31%	+0.20%
4	4		C++	6.52%	-0.27%
5	5		C#	4.97%	-0.35%
6	6		Visual Basic	4.85%	-0.40%
7	7		JavaScript	2.11%	+0.06%
8	8		PHP	2.07%	+0.05%
9	12	⬆	Assembly language	1.97%	+0.72%
10	9	⬇	SQL	1.87%	+0.03%
11	10	⬇	Go	1.31%	+0.03%
12	18	⬆	Classic Visual Basic	1.26%	+0.49%
13	11	⬇	R	1.25%	-0.01%
14	20	⬆	Delphi/Object Pascal	1.20%	+0.48%
15	36	⬆	Groovy	1.19%	+0.94%
16	14		Ruby	1.18%	+0.13%
17	17	⬇	Perl	1.15%	+0.24%
18	15	⬇	MATLAB	1.04%	+0.05%
19	13	⬇	Swift	0.95%	-0.28%

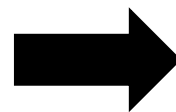
(출처: <https://www.tiobe.com/tiobe-index>)



- 우리가 입력한 문자들은 규칙을 지키고 있다면 그것은 프로그램으로 변환 할 수 있다.

소스코드

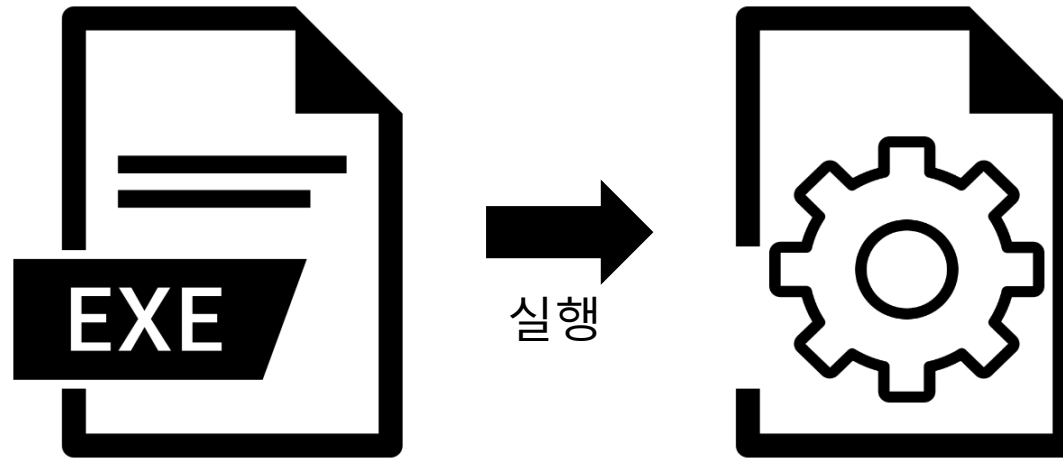
```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello World!");
6 }
```



컴파일

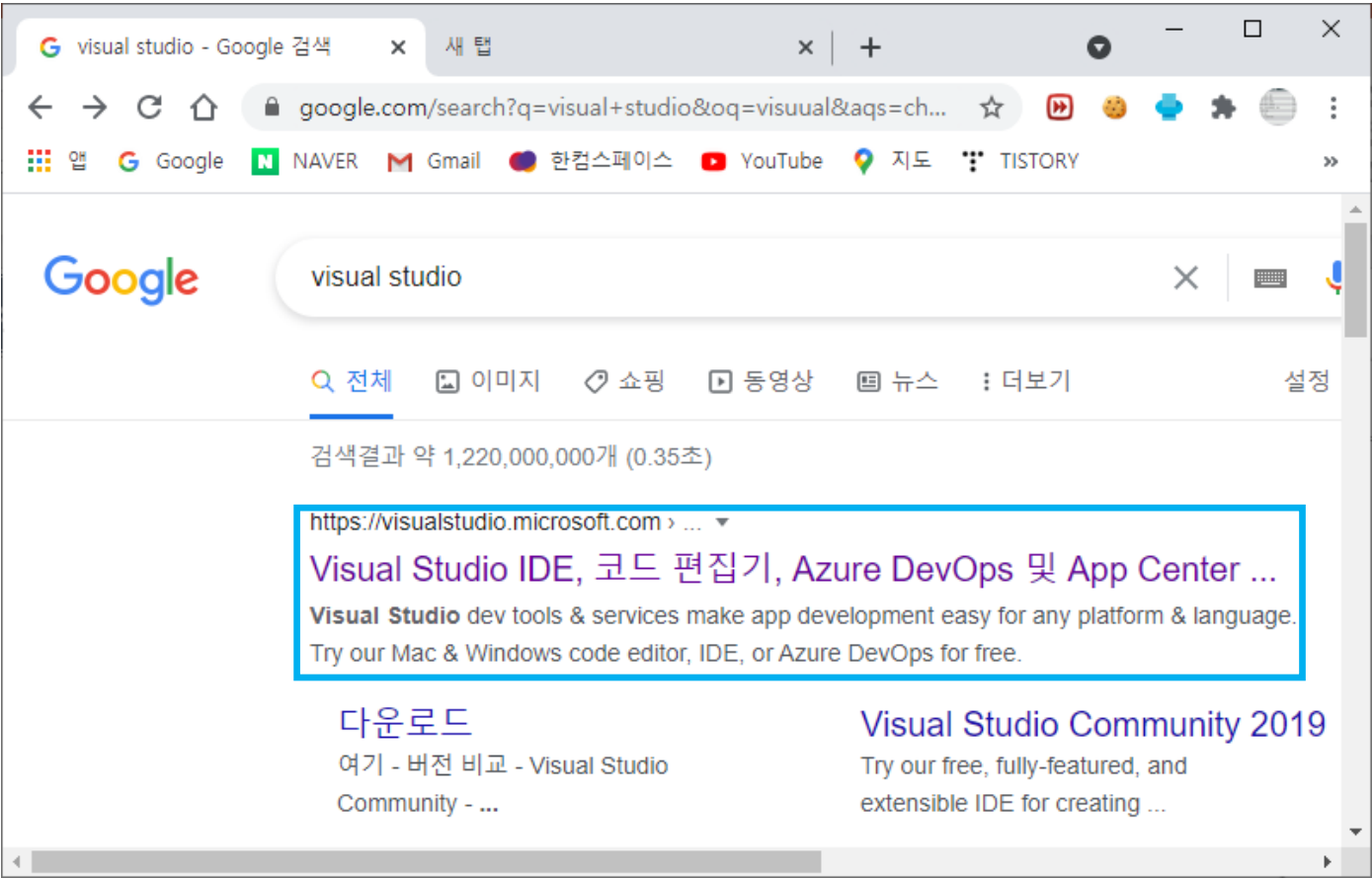


- 프로그램(정의) 컴퓨터에 저장되어 실행이 가능한 완성된 코드들의 집합체
- 컴퓨터는 실행파일(exe)를 실행시켜 프로세스라는 공간(메모리)을 할당합니다.
 - (정리)exe가 실행되면 프로세스라는 공간이 생성된다.

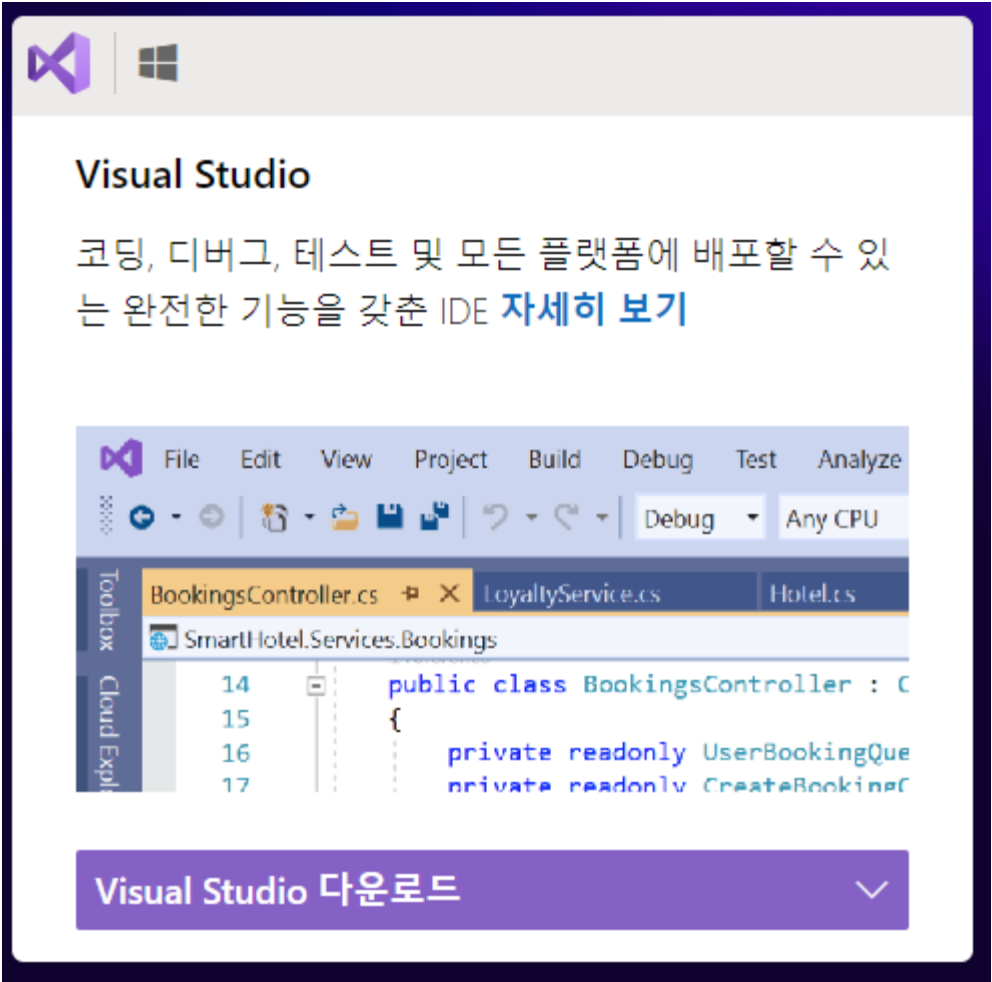


Visual Studio 2019

환경설정 및 환경설명

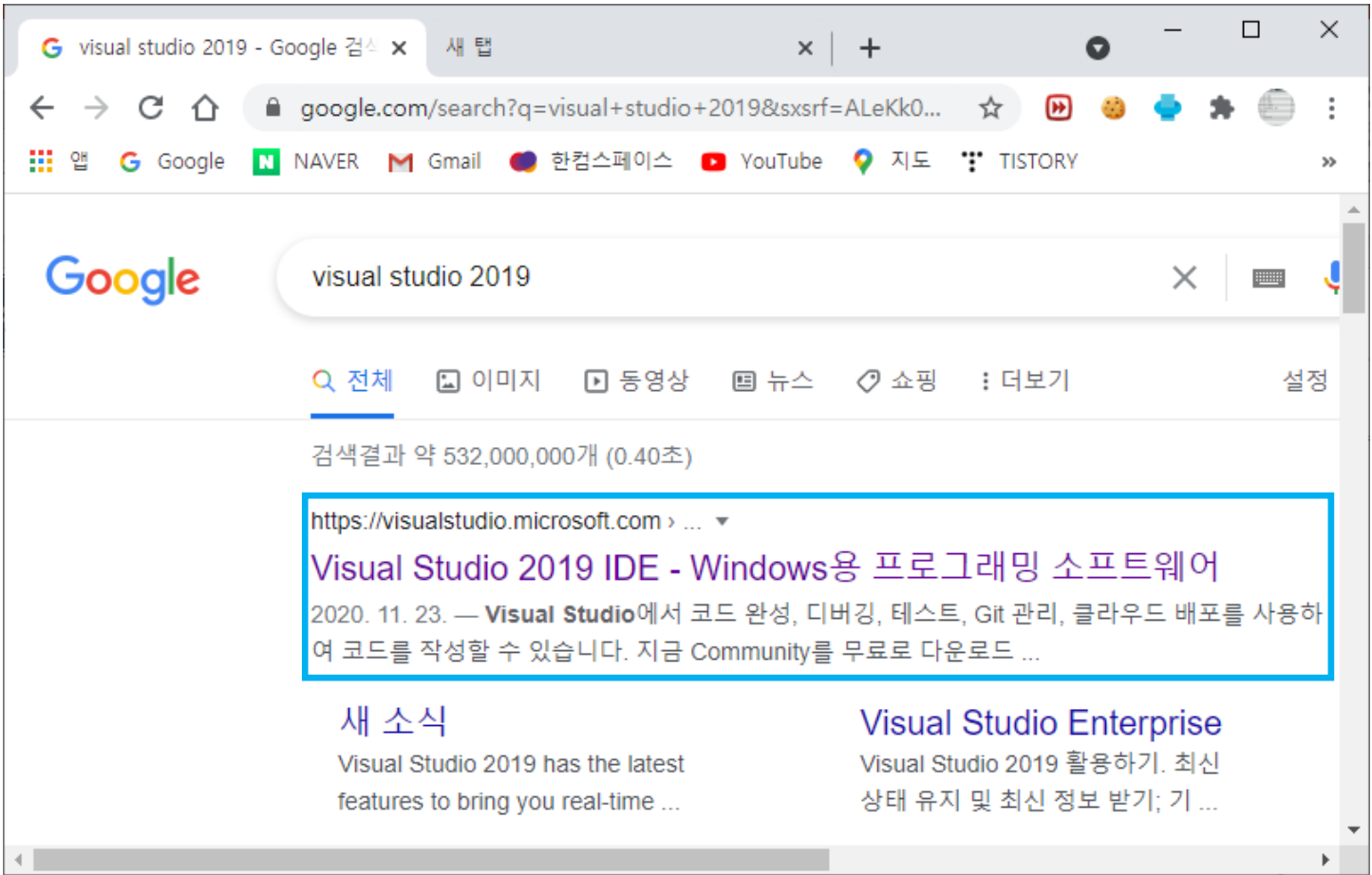


<https://visualstudio.microsoft.com/ko/>

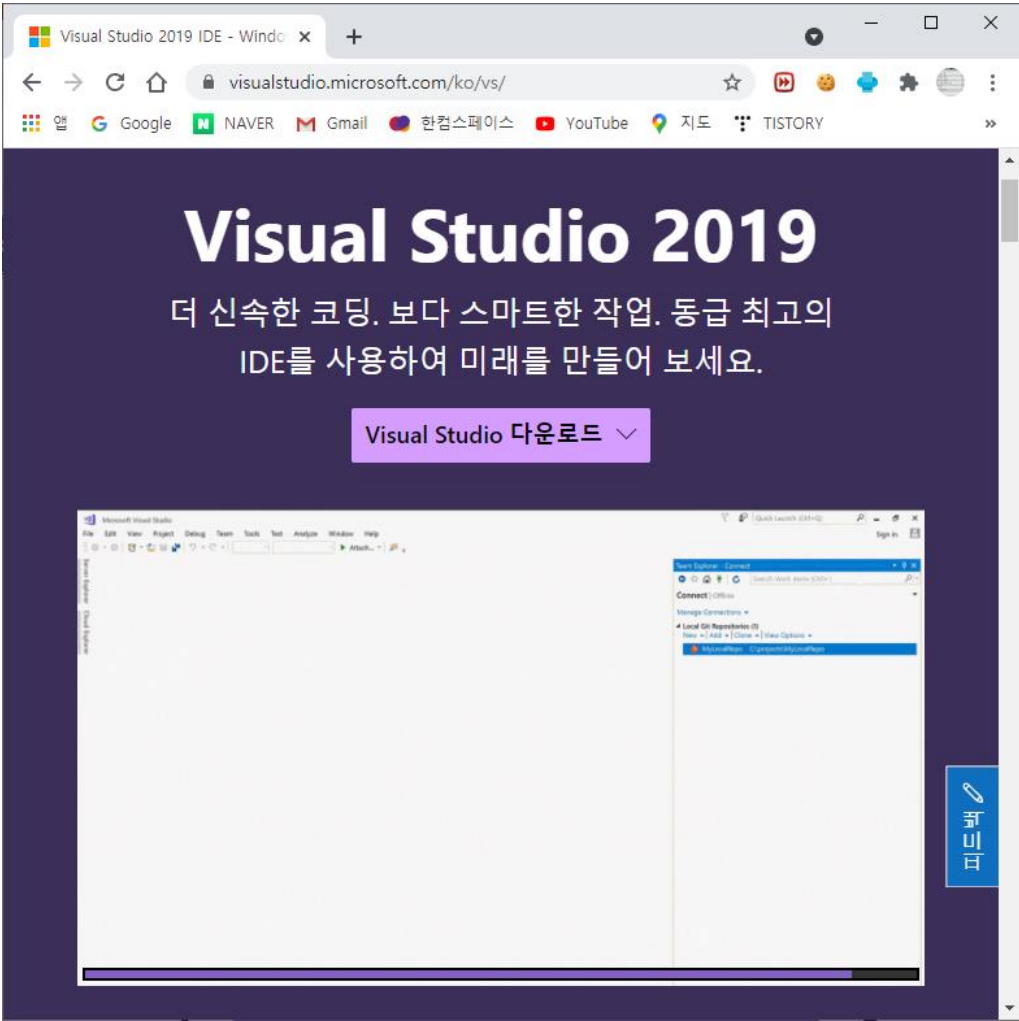


Visual Studio 다운로드	
Community 2019	↓
Professional 2019	↓
Enterprise 2019	↓

<https://visualstudio.microsoft.com/ko/>



<https://visualstudio.microsoft.com/ko/vs/>



<https://visualstudio.microsoft.com/ko/>

Visual Studio 다운로드 ▾


Community 2019 ↓


Professional 2019 ↓


Enterprise 2019 ↓



Visual Studio Installer

 거의 완료되었습니다... 모든 항목을 준비하는 중입니다.

 다운로드됨

 설치됨

설치 중 — Visual Studio Community 2019 — 16.9.4

워크로드 개별 구성 요소 언어 팩 설치 위치

웹 및 클라우드 (4)

ASP.NET 및 웹 개발
Docker 지원이 포함된 ASP.NET Core, ASP.NET, HTML/JavaScript 및 컨테이너를 사용하여 웹 애플리케이션을 빌드...

Python 개발
Python에 대한 편집, 디버깅, 대화형 개발 및 소스 제어입니다.

Azure 개발
.NET Core 및 .NET Framework를 사용하여 클라우드 앱을 개발하고 리소스를 만들기 위한 Azure SDK, 도구 및 프로...

Node.js 개발
비동기 이벤트 구동 JavaScript 런타임인 Node.js를 사용하여 확장 가능한 네트워크 애플리케이션을 빌드합니다.

데스크톱 및 모바일 (5)

.NET 데스크톱 개발
.NET Core and .NET Framework와 함께 C#, Visual Basic 및 F#를 사용하여 WPF, Windows Forms 및 콘솔 애플리케이션을...

C++를 사용한 데스크톱 개발 ☒
MSVC, Clang, CMake 또는 MSBuild 등 선택한 도구를 사용하여 Windows용 최신 C++ 앱을 빌드합니다.

유니버설 Windows 플랫폼 개발
C#, VB 또는 C++(선택 사항)를 사용하여 유니버설 Windows 플랫폼용 애플리케이션을 만듭니다.

.NET을 사용한 모바일 개발
Xamarin을 사용하여 iOS, Android 또는 Windows용 플랫폼 간 애플리케이션을 빌드합니다.

위치
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community 변경...

계속하면 선택한 Visual Studio 버전에 대한 [라이선스](#)에 동의하게 됩니다. Microsoft는 Visual Studio와 함께 다른 소프트웨어를 다운로드할 수 있는 기능을 제공합니다. 이 소프트웨어는 [타사 고지 사항](#) 또는 해당 라이선스에 명시된 것처럼 별도로 라이선스가 부여됩니다. 계속하면 이러한 라이선스에도 동의하게 됩니다.

필요한 전체 공간 4.54GB

다운로드하는 동안 설치 설치(I)



×

Visual Studio

Visual Studio에 로그인

- 디바이스 간에 설정 동기화

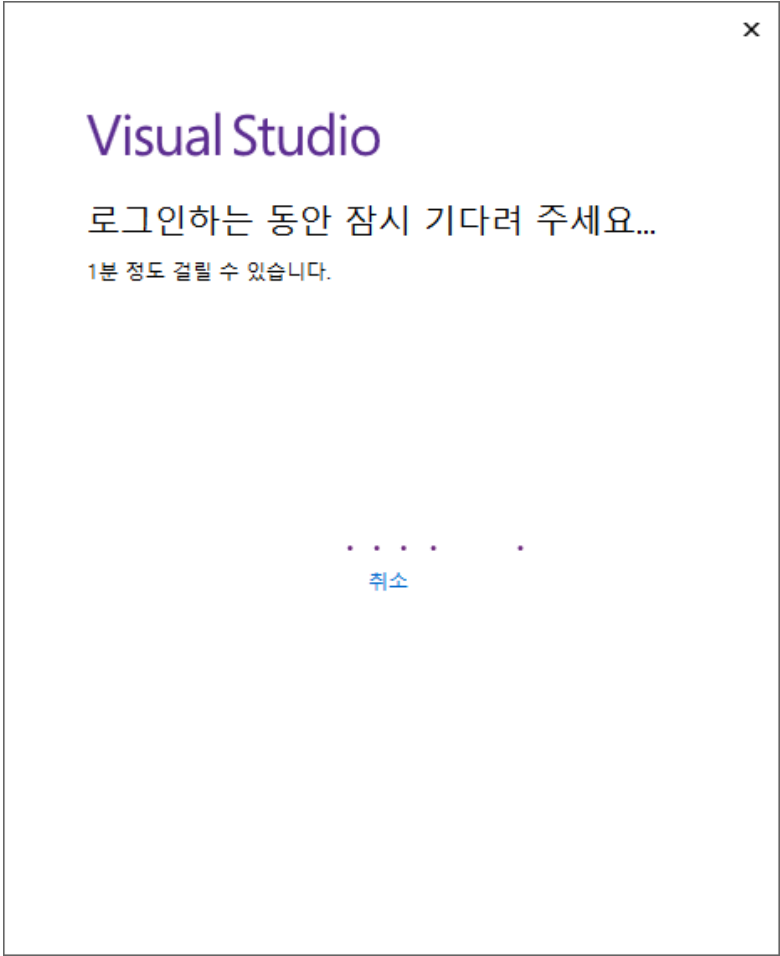
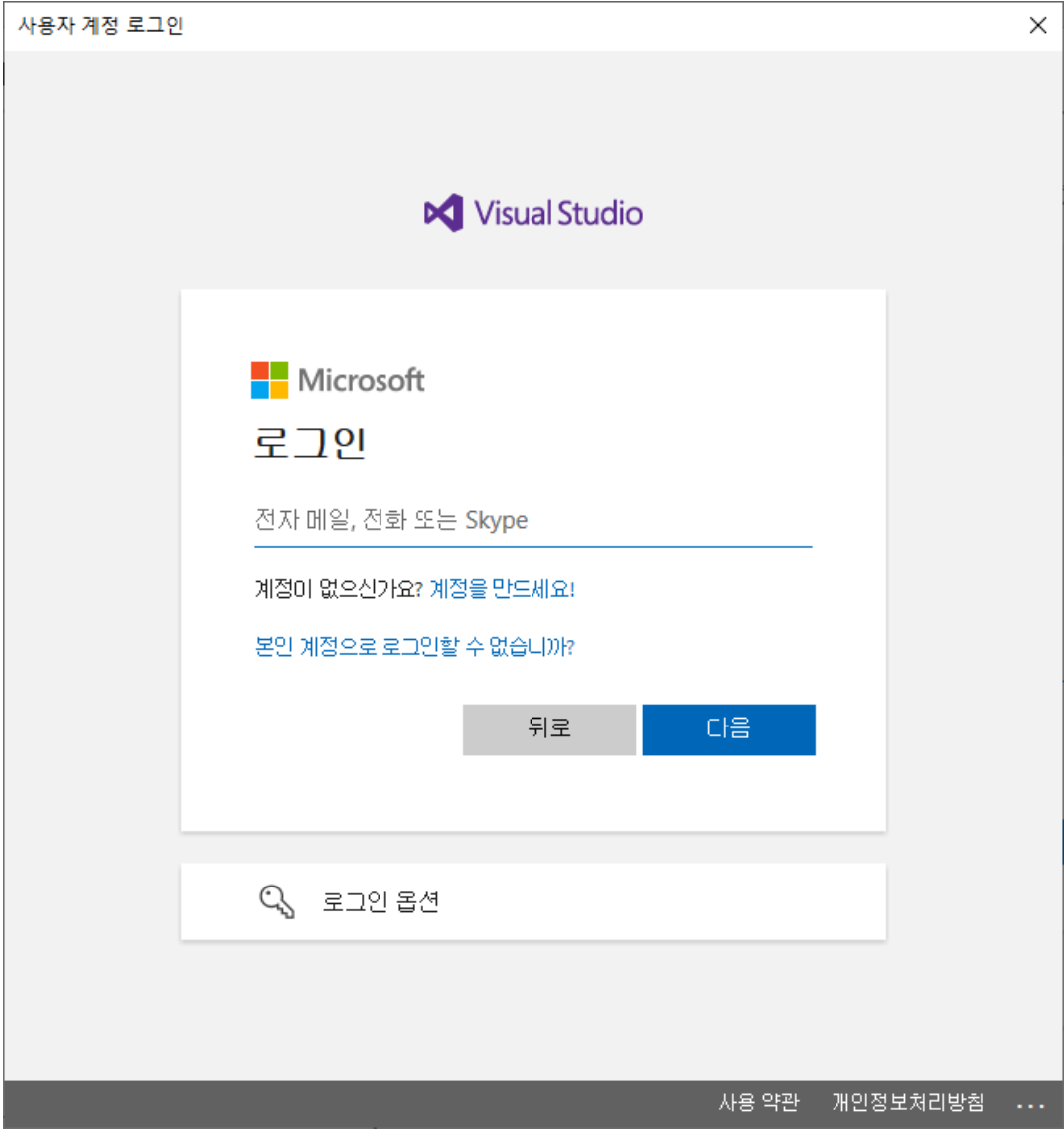
- Live Share를 사용하여 실시간으로 협업

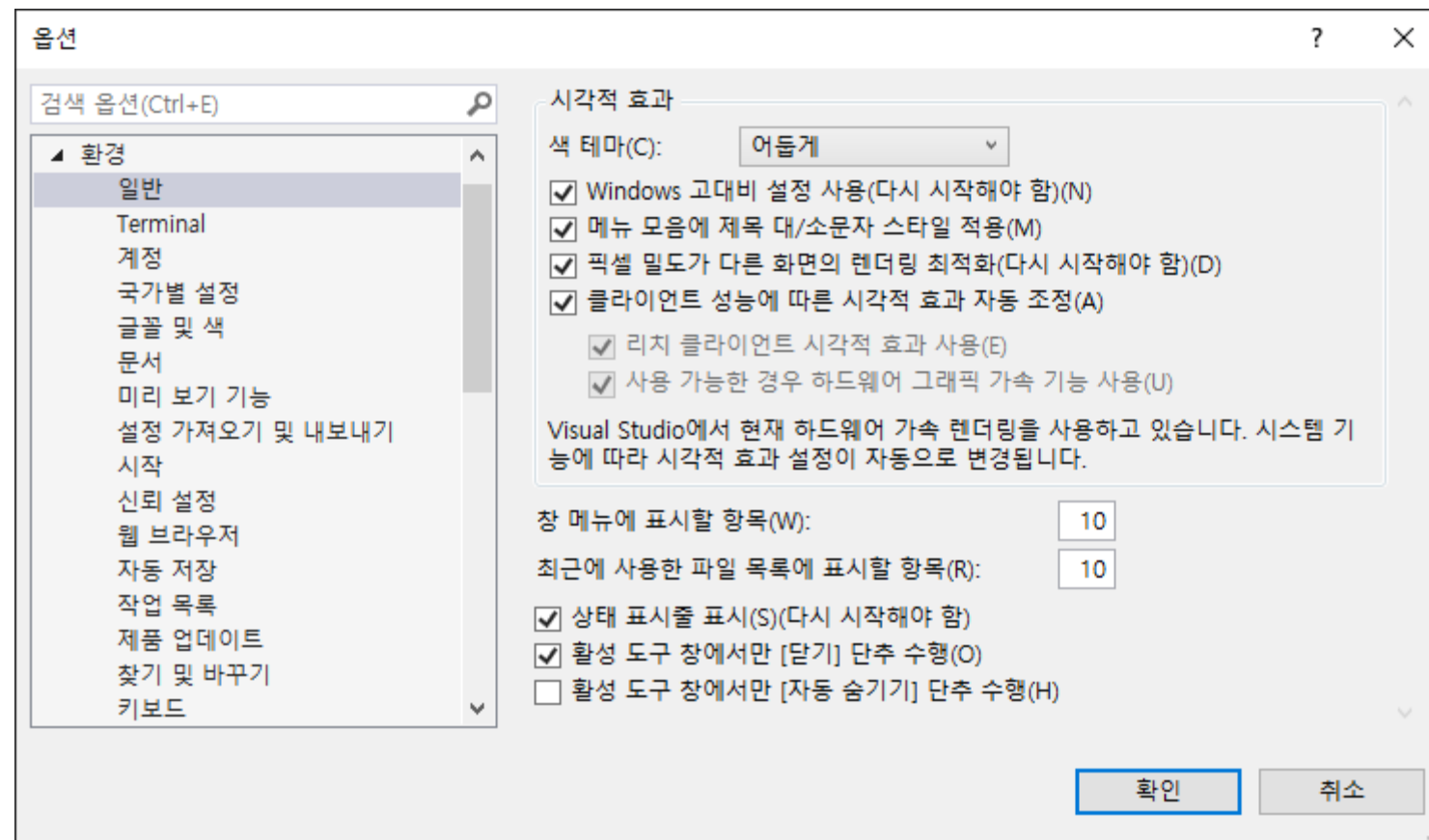
- Azure 서비스와 원활하게 통합

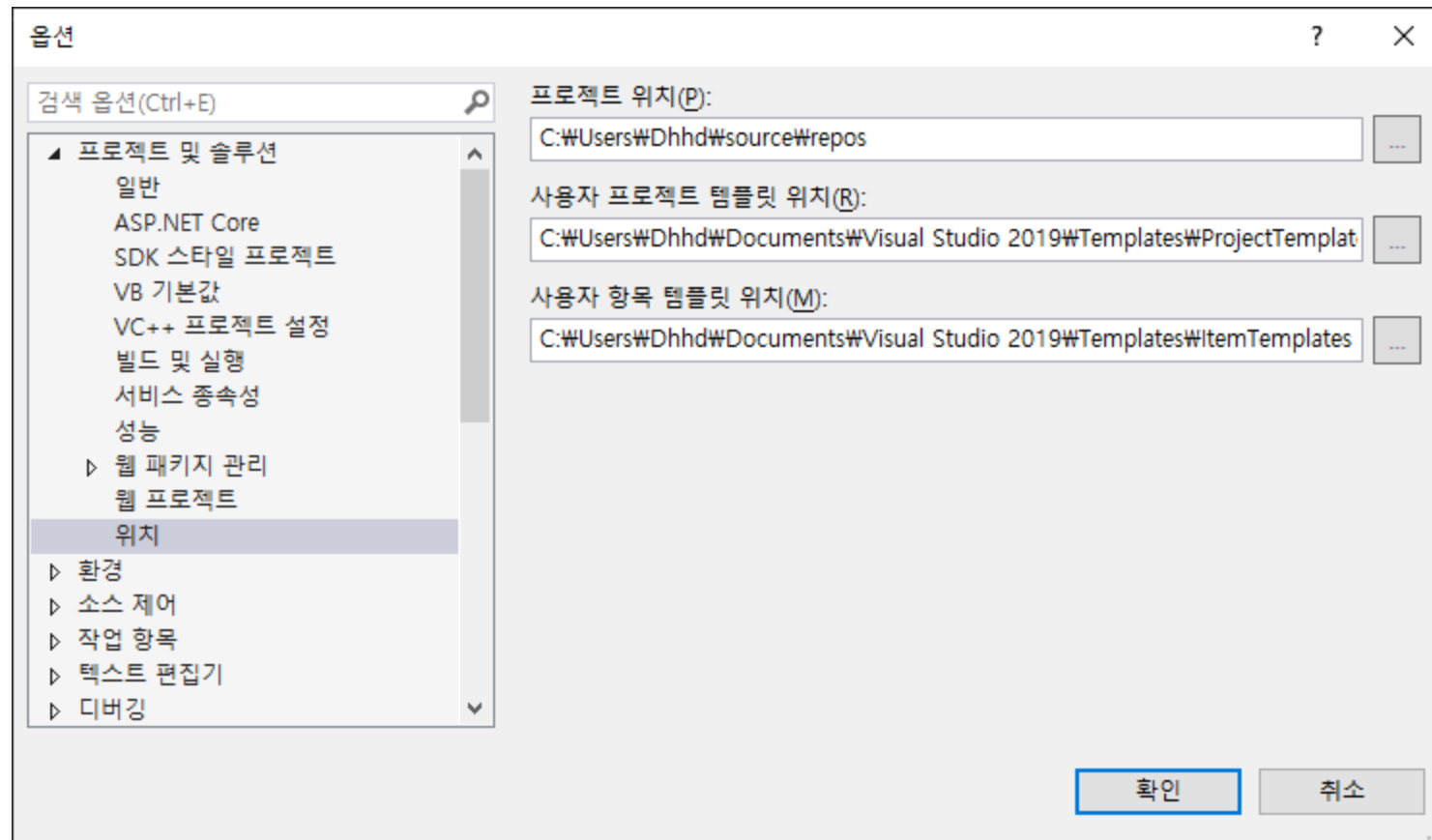
로그인(I)

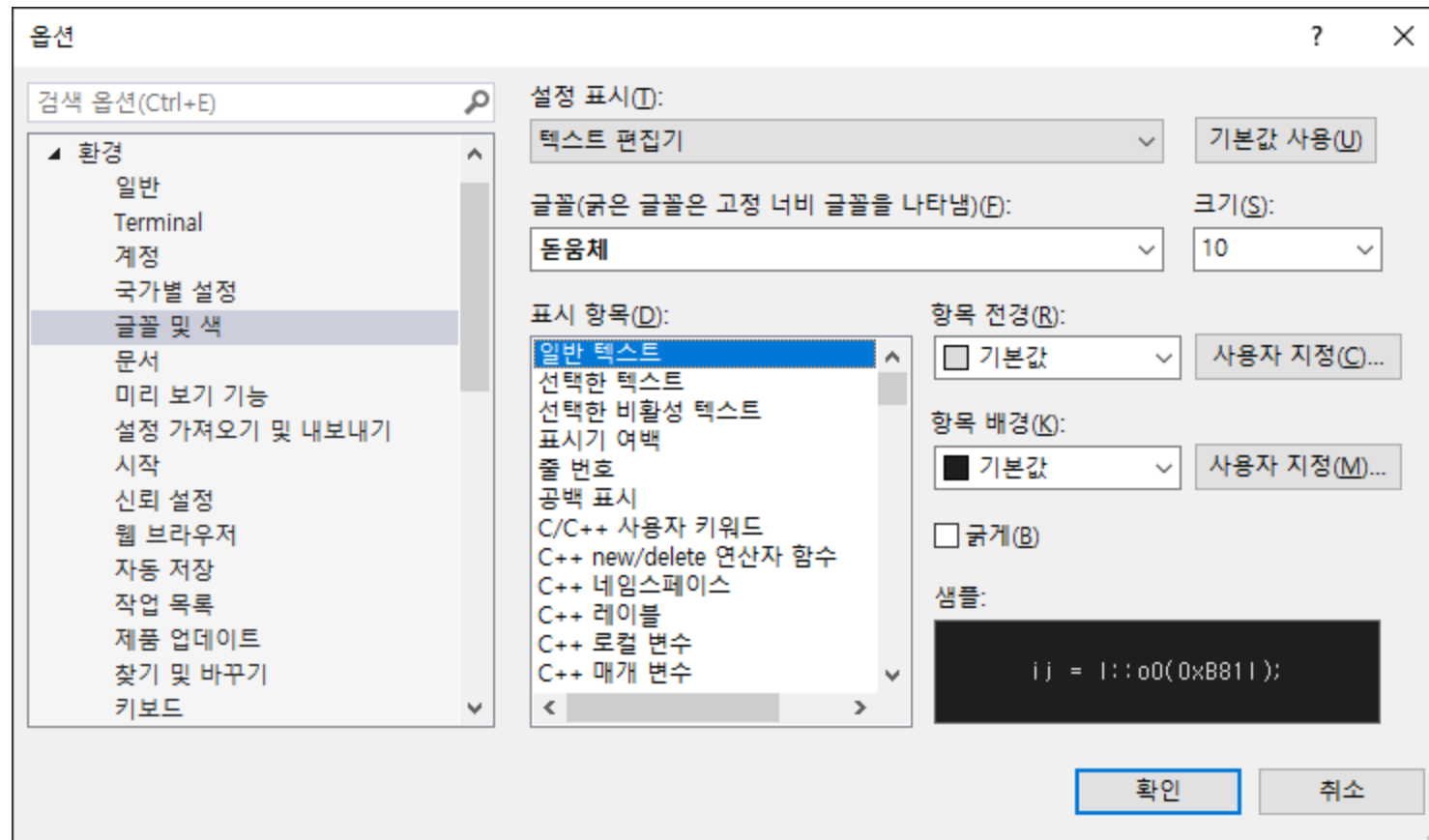
계정이 없는 경우 새로 만드세요!

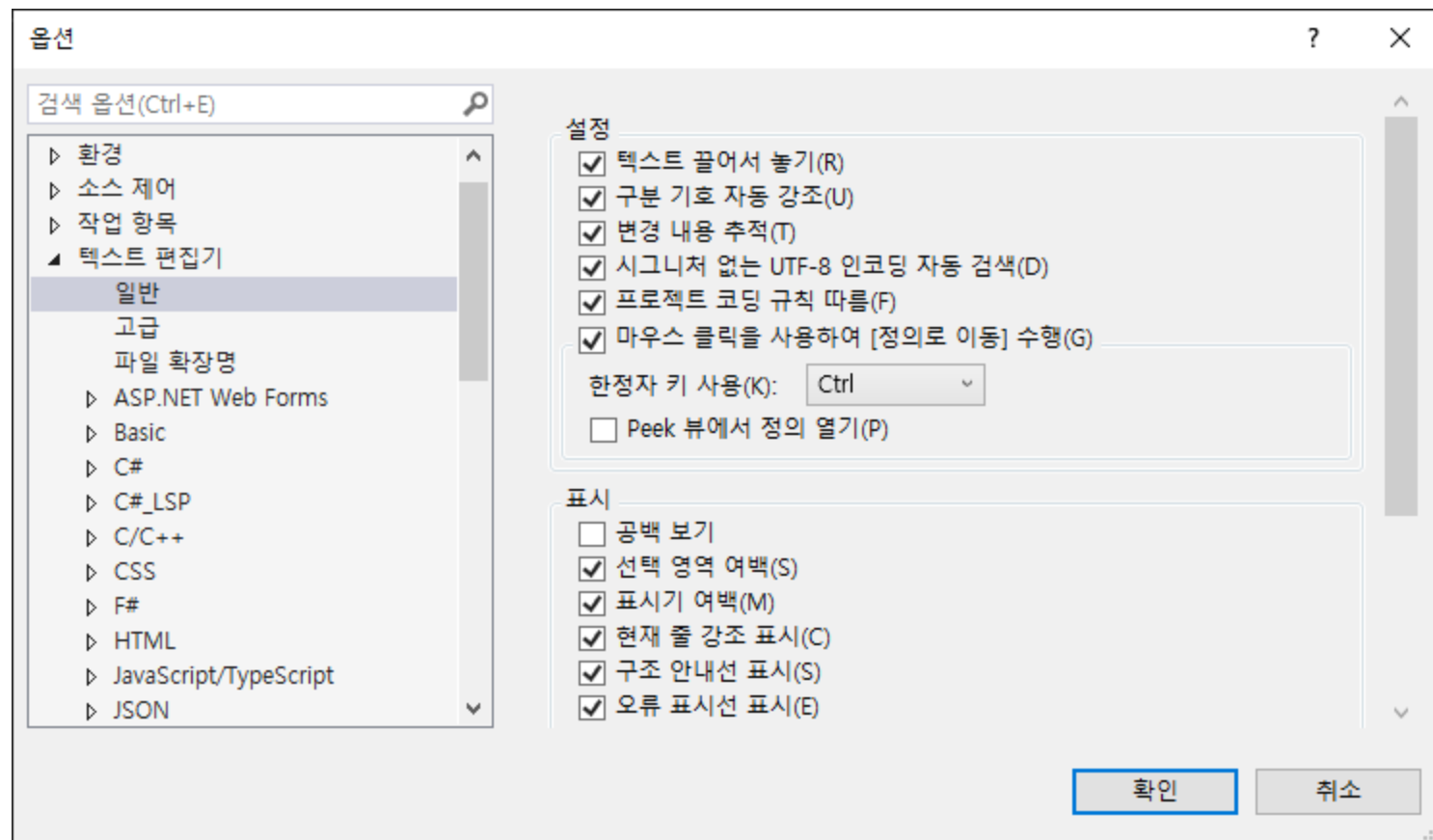
나중에 로그인

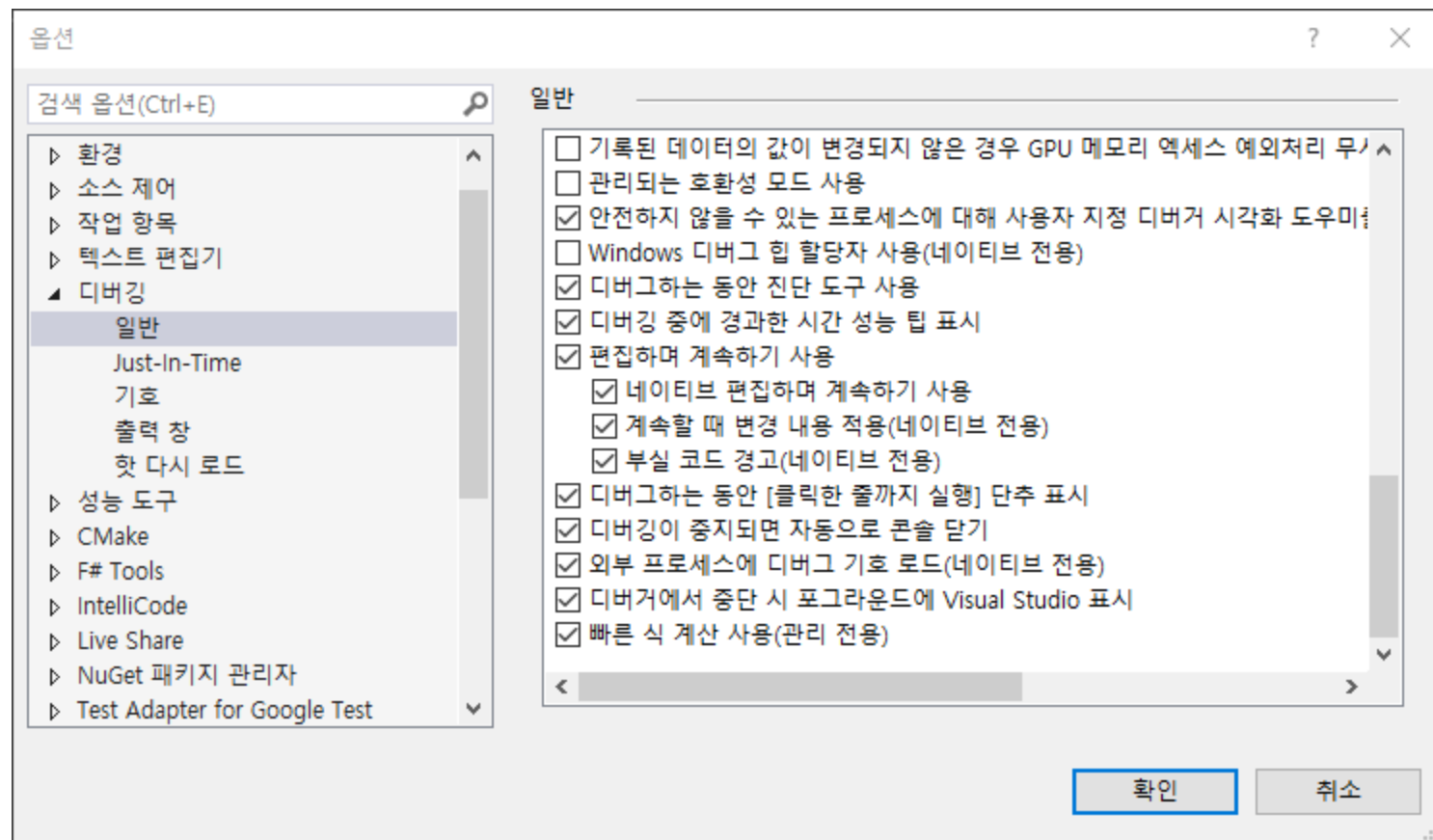














- 코딩(소스코드 입력)[텍스트 편집기]
- 컴파일(빌드 자동화)[컴파일러]
- 디버거
- 고대에는 컴파일러, 텍스트 편집기, 디버거 는 따로 사용했다.

C언어 시작

새프로젝트 및 기초문법





Visual Studio 2019


최근 파일 열기(R)


Visual Studio를 사용할 때 여는 프로젝트, 폴더 또는 파일은 빠른 액세스를 위해 여기에 표시됩니다
항상 목록의 맨 위에 표시되도록 자주 여는 항목을 고정할 수 있습니다.


시작

**코드 영역에 연결(E)**
클라우드 기반 개발 환경 만들기 및 관리

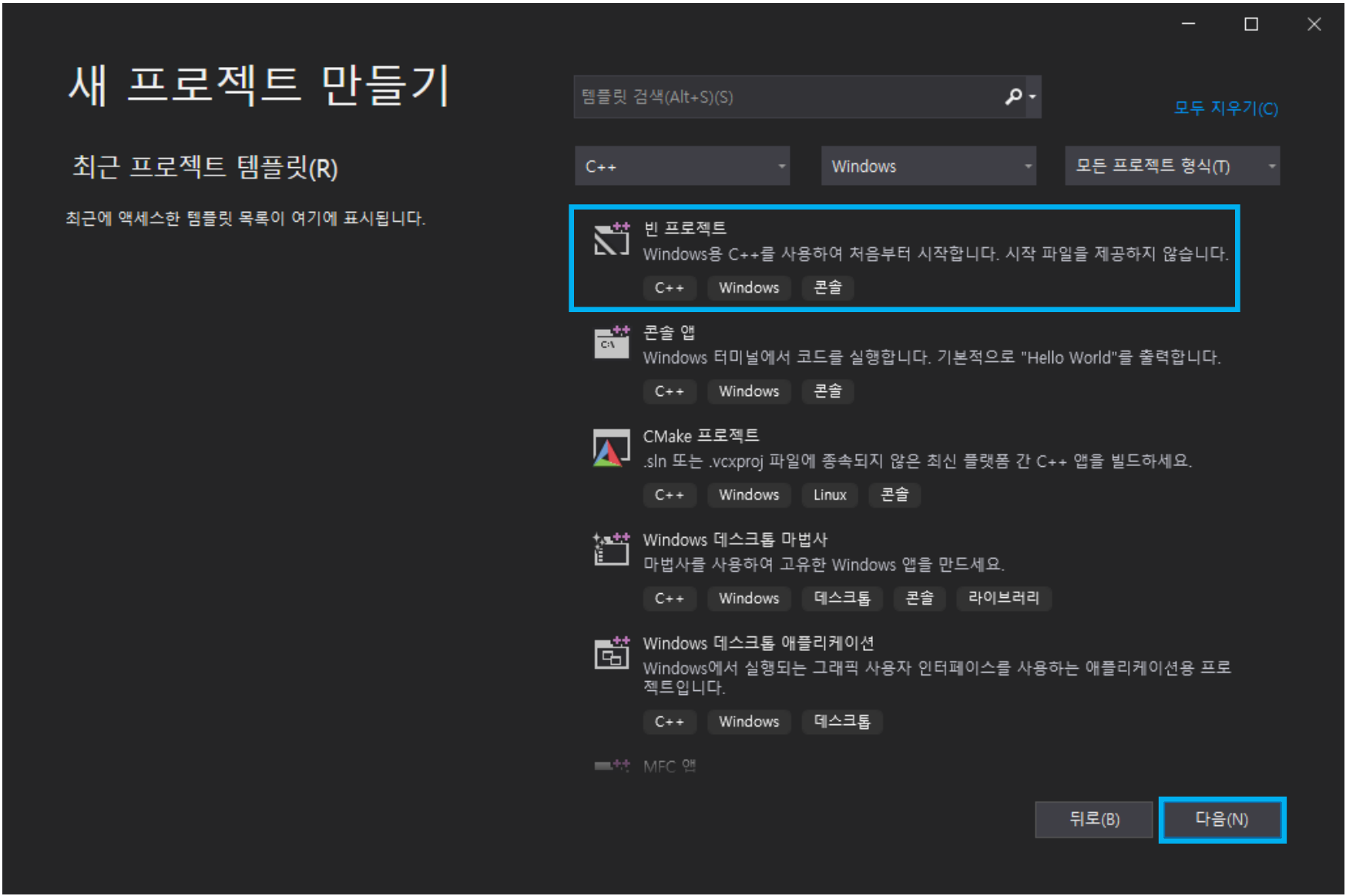
**리포지토리 복제(C)**
GitHub 또는 Azure DevOps 같은 온라인 리포지토리에서 코드 가져오기

**프로젝트 또는 솔루션 열기(P)**
로컬 Visual Studio 프로젝트 또는 .sln 파일 열기

**로컬 폴더 열기(F)**
폴더 내에서 탐색 및 코드 편집

**새 프로젝트 만들기(N)**
시작하려면 코드 스캐폴딩과 함께 프로젝트 템플릿을 선택하세요.

[코드를 사용하지 않고 계속\(W\) →](#)





새 프로젝트 구성

빈 프로젝트

C++

Windows

콘솔

프로젝트 이름(N)

Project1

위치(L)

C:\Users\Dhhd\source\repos

...

솔루션 이름(M) ⓘ

Project1

☒ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

언어/플랫폼/프로젝트형식

프로젝트(이름)

저장위치

찾아보기

솔루션 이름

솔루션 파일위치

뒤로(B)

만들기(C)

찾아보기

Dhhd > source > repos

repos 검색

구성 새 폴더

즐거찾기

바탕 화면

다운로드

문서

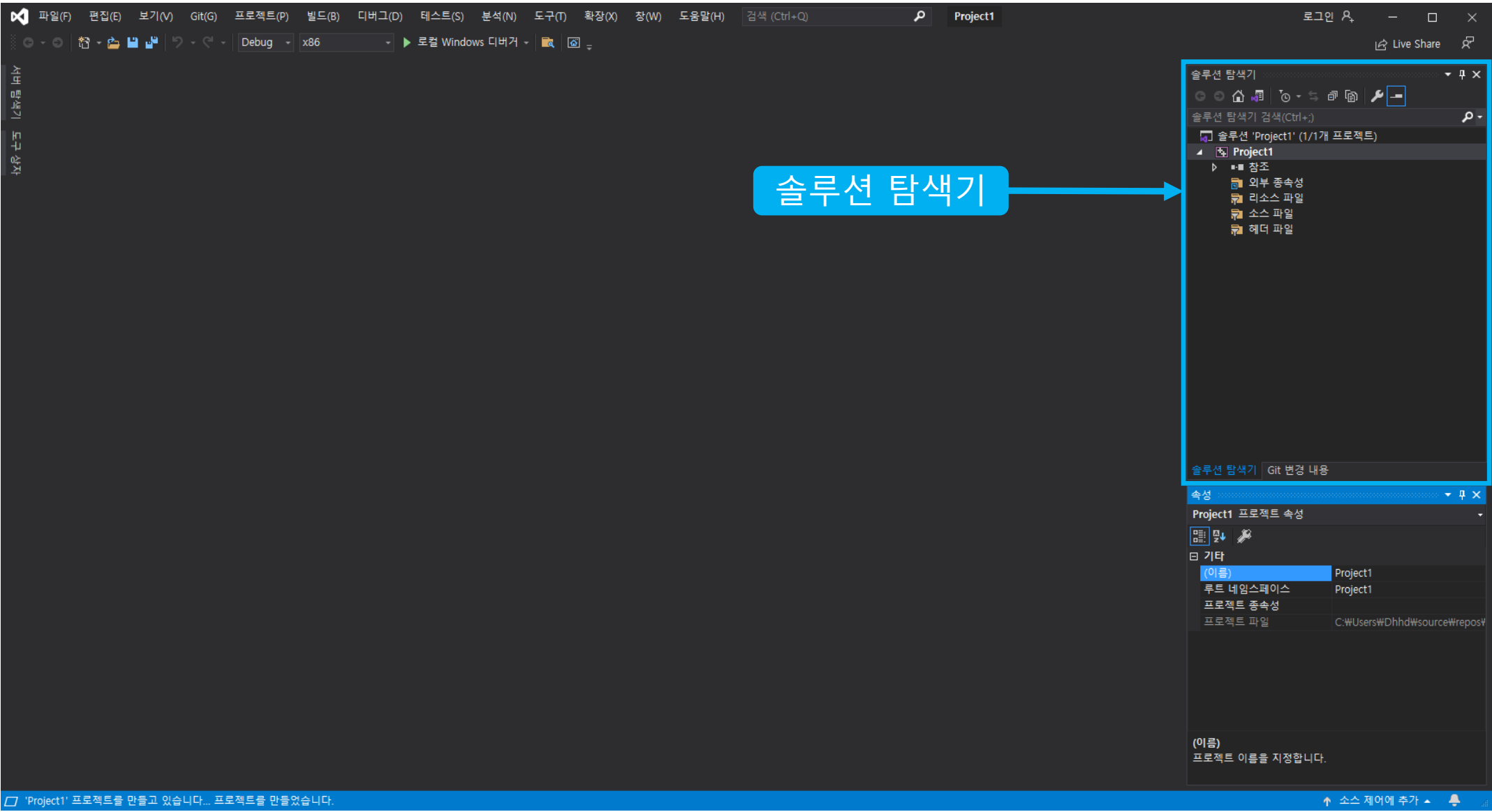
사진

이름	수정한 날짜	유형
Project1	2021-03-11 오후 9:54	파일 폴더

폴더:

폴더 선택

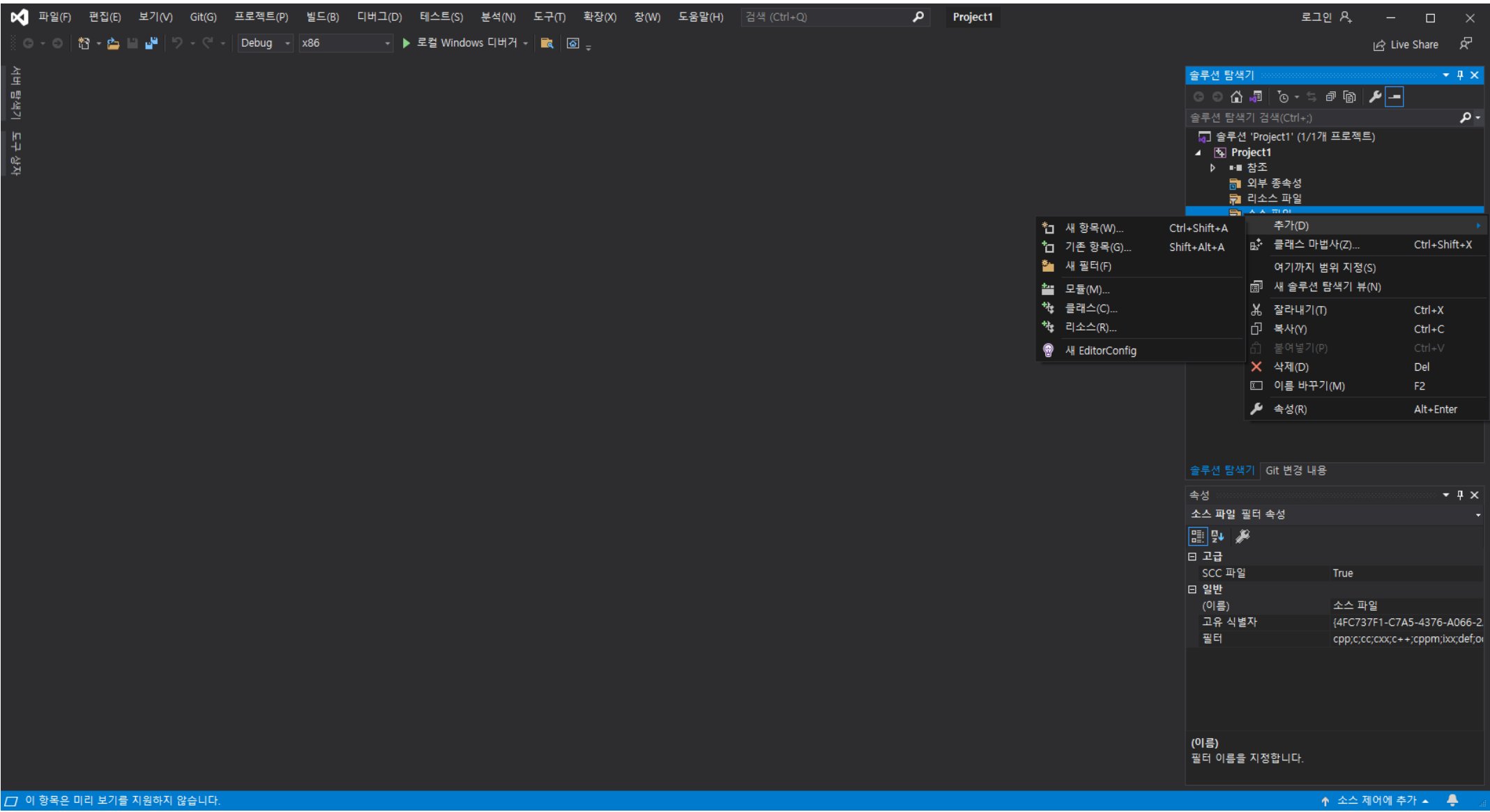
취소





소스코드 추가하기

C언어





새 항목 추가 - Project1

설치됨

Visual C++

코드

서식

ATL

데이터

리소스

웹

유틸리티

속성 시트

HLSL

Test

그래픽

온라인

정렬 기준: 기본값

검색(Ctrl+E)

C++ 파일(.cpp)

Visual C++

헤더 파일(.h)

Visual C++

C++ 클래스

Visual C++

C++ 모듈 인터페이스 단위(.ixx)

Visual C++

유형: Visual C++

C++ 소스 코드를 포함하는 파일을 만듭니다.

찾아보기

Dhhd > source > repos

repos 검색

구성 새 폴더

즐거찾기

바탕 화면

다운로드

문서

사진

이름

수정한 날짜

유형

Project1

2021-03-11 오후 9:54

파일 폴더

폴더:

폴더 선택

취소

이름(N):

소스.cpp

위치(L):

C:\Users\Dhhd\source\repos\Project1\

찾아보기(B)...

추가(A)

취소

설명

찾아보기

설명



- main함수는 C언어의 소스코드에서 가장 먼저 사용하는 함수의 이름이다.
- C언어는 절차지향(위에서 아래로 흐름이 일어난다)적 언어이다.
- 문장의 끝은 한국어처럼 .이 아니라 ;으로 문장을 끝낸다. (C언어 또한 언어이다)

```
#include <stdio.h>

int main( )
{
    ① printf("Hello world!\n");
    ② return 0;
}
```

① printf함수를 호출한다
② 0을 반환 시킨다(종료)

main함수(반환 자료형 int)(매개변수 없음)

문장의 끝([;]세미콜론)

return 0;(0을 반환 시킨다)

코드의 시작(코드블럭)



- include <stdio.h>는 stdio.h(헤더파일)을 포함시키는 문장이다.
- stdio는 standard input output의 약자로 printf함수가 정의 되어있다.
- 우리는 printf함수를 #include <stdio.h>라는 문장을 통해 정의를 불러온 것이다.

```
#include <stdio.h>
```

```
int main()  
{  
    printf("Hello world!\n");  
  
    return 0;  
}
```

```
_Check_return_opt_  
_CRT_STDIO_INLINE int __CRTDECL printf(  
    _In_z_ _Printf_format_string_ char const* const _Format,  
    ...)  
#if defined _NO_CRT_STDIO_INLINE
```



```
#include <stdio.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

문자열은 "" 사이에 넣는다

standard input output

main함수(int)

printf함수(parameter : 문자열)

return 0;(함수 종료용)



```
#include <stdio.h>
```

standard input output

```
int main( )
```

main함수(int)

```
{
```

```
printf( "%s\n", "Hello world!" );
```

printf함수(parameter : 문자열)

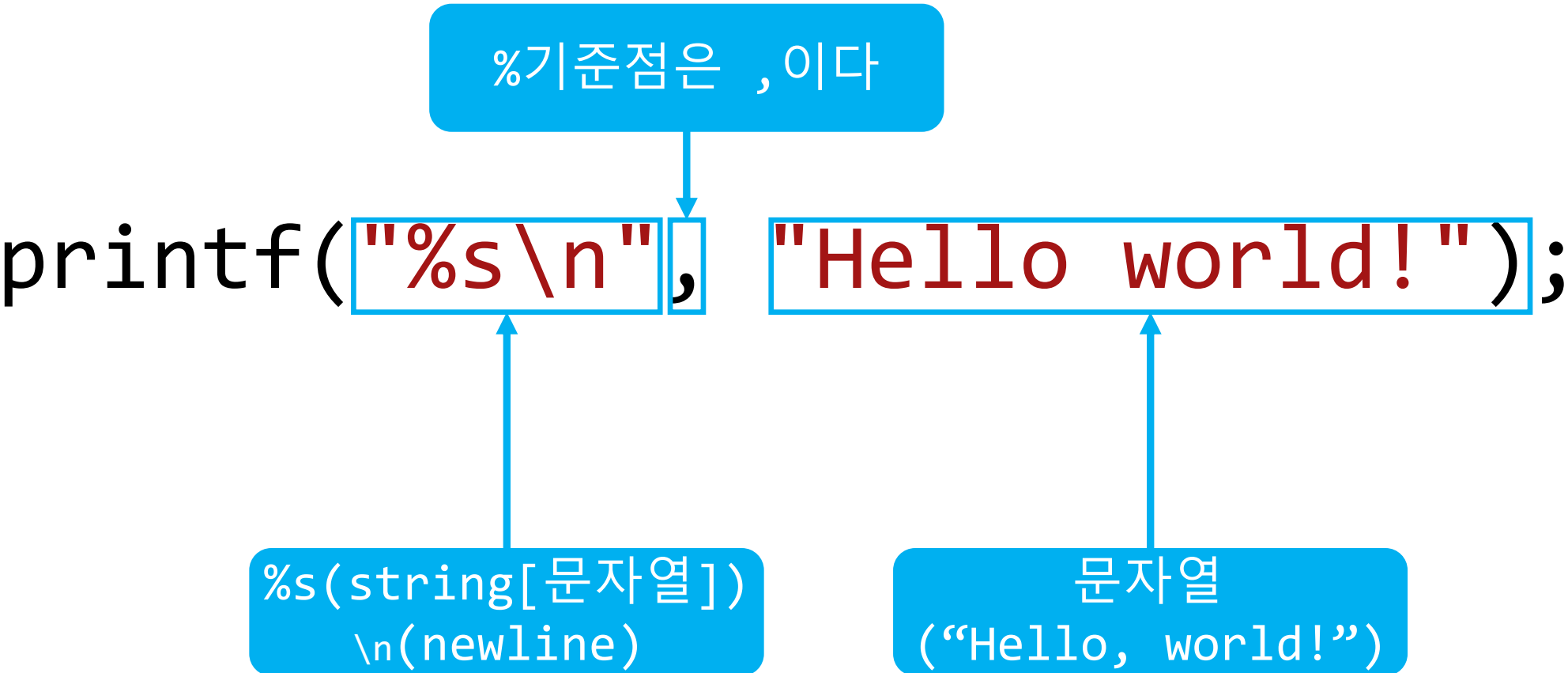
```
return 0;
```

return 0;(함수 종료용)

```
}
```



- 함수 속의 퍼센트(%) 기호들은, "Format Specifier" 라고 하는데, 출력 형식을 지정하는 것





```
#include <stdio.h>
```

```
int main()
```

```
{
```

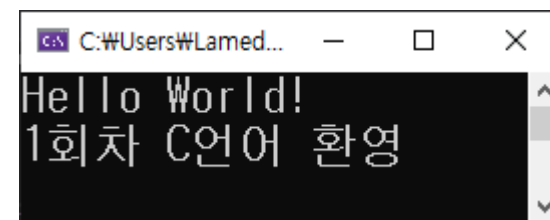
```
    printf( );
```

```
    printf( );
```

```
    return 0;
```

```
}
```

출력결과





```
#include <stdio.h>

int main()
{
    printf("Hello world!\n");
    printf("C언어 2회차 환영\n");

    return 0;
}
```



```
#include <stdio.h>
```

```
int main()
```

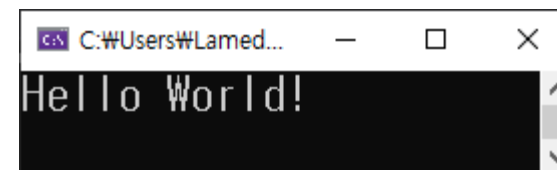
```
{
```

```
    printf("Hello %s\n",         );
```

```
    return 0;
```

```
}
```

출력결과





```
#include <stdio.h>

int main()
{
    printf("Hello %s\n", "World!");

    return 0;
}
```



```
1 자료형 main() ← main이 꼭 있어야하는 함수  
2 { ← 블록  
3 함수 1();  
4 함수 2();  
5 여러가지 문법; ← (예 int a = 5;)  
6 : ← 레이블 goto등에 사용  
7 } ← 블록
```

`printf("Hello world!");` ← 문장의 마침표

문장(statement)

작은따옴표' ', 큰따옴표" ", 소괄호 (), 중괄호{ }, 대괄호[]



- 주석은 코드에 대한 설명이나 특정 코드를 임시로 사용하지 않기 위해 주로 사용됩니다.

뒤에 있는걸 주석처리

// Hello, world! 출력

주석이 된 코드는 초록색이 된다

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



- 부분(블럭)주석은 부분적으로 코드를 주석 처리할 때 사용합니다.

```
/*  
주석사용 방법  
이 주석은 블록 형태로 주석이 됩니다.  
*/  
/* 이렇게 *사이에 문장이 주석이 됩니다. */  
  
// 이 블록은 한 줄만 주석이 됩니다.  
// 주석은 해당 코드에 대한 코멘트입니다.
```

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

특정 코드를 임시로 사용하지 않을 때

```
int a = 1 + 2; // 더하기
```

코드의 대한 설명

```
printf("Hello, world!\n"); //printf("1234567890");
```

특정 코드를 바꾸어 가면서 사용할 때

```
/* printf("Hello, world!\n");  
printf("1234567890"); */
```

특정 코드들을 임시로 사용하지 않을 때

```
printf("Hello" /* 안녕하세요 */);
```

코드이 부분에 주석 넣기



```
int main()  
{  
}  

```

줄의 시작

```
int main() {  
}  

```

줄의 마지막



- 코드가 보기 좋게 띄어쓰기를 Tab키나 Space로 간격을 맞추어서 쓴다

```
#include <stdio.h>

int main()
{
    printf("Hello world!");
    return 0;
}
```



문자	의미	기능
\n	new line	칸을 다음 줄 앞으로 이동
\b	backspace	한 칸 뒤로 한 칸 이동
\t	tab	일정 간격 띄움
\r	carriage return	현재 줄의 처음으로 이동
\0	null	널(0) 문자 출력
\'	singe quote(‘)	작은 따옴표 출력
\”	double quote(“)	큰 따옴표 출력
\a	alert	벨 소리 발생
\\	backslash	역 슬래시 출력
\f	from feed	한 페이지 넘김



```
#include <stdio.h>
```

```
int main()  
{
```

```
    printf("제어문자의 사용\n"); //new line
```

```
    printf("1\t2\n"); //tab
```

```
    printf("1");
```

```
    printf("24\b34\n"); //backspace
```

```
    printf("223\r1\n"); //carriage return(줄의 맨 앞으로 이동)
```

```
    printf("I should be an example, not an exception\a\n");
```

```
    return 0;
```

```
}
```



```
C:\Users\Lamed\Desktop\Project1\Debug\Project1.exe
제어문자의 사용
1      2
1234
123
I should be an exampe, not an exception
```



```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    1 + 1; //더하기는 +
```

```
    1 - 1; //빼기는 -
```

```
    1 / 1; //나누기는 /
```

```
    1 * 1; //곱하기는 *
```

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

```
    printf("%f\n", 3.14);
```

```
    printf("%3d\n", 10); //숫자d에서 숫자는 자릿수를 의미합니다
```

```
    printf("%.2f\n", 3.14); //소수점자릿수f
```

```
    return 0;
```

```
}
```



```
C:\Users\Lamed...  
10  
3.140000  
10  
3.14
```



출력 데이터	사용 예제	출력 결과
문자열	<code>printf("Lamed");</code>	Lamed 출력
제어 문자	<code>printf("Lamed\\n");</code>	Lamed 출력 후 줄 바꿈
정수	<code>printf("%d", 10);</code>	정수 10 출력
실수	<code>printf("%f", 3.14);</code>	실수 3.140000 출력
수식	<code>printf("%d", 10 + 20);</code>	10 + 20 인 30을 출력

제어 문자	의미	기능
<code>\\n</code>	개행(new line)	줄을 바꾼다
<code>\\t</code>	탭(tab)	탭을 넣는다
<code>\\r</code>	캐리지 리턴(carriage return)	출력 위치를 줄의 맨 앞으로 바꾼다
<code>\\b</code>	백스페이스(backspace)	출력 위치를 한 칸 왼쪽으로 옮긴다
<code>\\a</code>	알럿(alert) 경고	벨(bell)소리를 낸다



```
#include <stdio.h>

int main()
{
    int a = 1 + 2;
    printf("%d", a);
    printf("%d", 1 + 2);
    printf("%d", 2 - 1);
    printf("%d", 2 * 2);
    printf("%d", 2 / 2);
    printf("%d", 13 % 2);
}
```

- ① 더하기
- ② 빼기
- ③ 곱하기
- ④ 나누기
- ⑤ 나머지(나누고 남은 나머지)



기호	이름
+=	더하고 대입
-=	빼고 대입
*=	곱하고 대입
/=	나누고 대입
%=	나누고 나머지 대입

변수1 = 변수1 + 변수2;

변수1 = 변수1 ? 변수2;

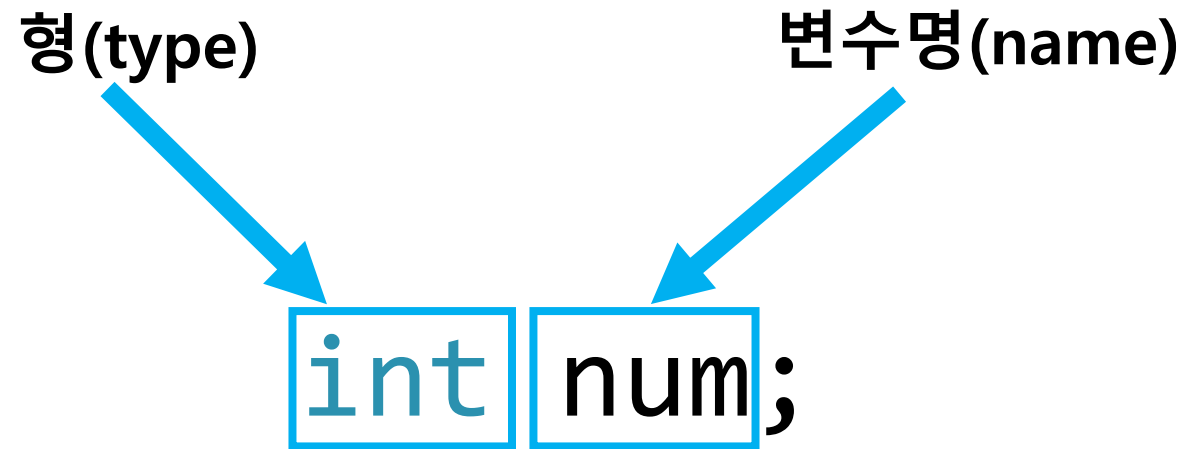
변수1 ?= 변수2;

변수

int float char



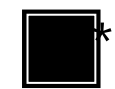
- **int**는 정수를 뜻하는 **integer**의 줄임말 입니다
- 변수는 메모리(memory)라는 공간에 존재한다.





- 변수 = 값;
- 식(expression)

```
int a;  
a = 777;
```

- 숫자로 시작할 수 없으면
- 언어에서 이미 사용중인 int return 등으로 사용할 수 없고
- 특수문자는 _ 제외 안됩니다.
 - (- + / * [] ()) 등은 이미 C언어에서 의미를 갖고있다
 - _ 는 C언어에서 의미를 갖고 있지 않습니다.
 - @, \$ 은 Visual Studio 기준 인식할 수 없는 문자입니다.
- 한글은 변수로 사용이 가능합니다.(Visual Studio 2019)

```
int 한글 = 1;  
printf( "%d", 한글 );
```

```
int money돈_홍길동 = 777;
```

```
int a = 777;  
int A = 777;  
int a_v2 = 777;  
int Math = 777;
```

```
int 0a = 777;  
int int = 777;  
int return = 777;  
int is<>-a = 777;
```

배열하다 配列하다/排列하다 [배:열하다] 🔊 +

- 1. 동사 일정한 차례나 간격에 따라 벌여 놓다.
- 2. 동사 정보·통신 동일한 성격의 데이터를 관리하기 쉽도록 하나로 묶다.

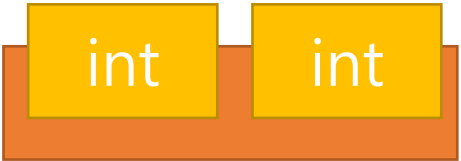
표준국어대사전

배열² 配列/排列 [배:열] 🔊 +

- 1. 명사 일정한 차례나 간격에 따라 벌여 놓음.
- 2. 명사 정보·통신 동일한 성격의 데이터를 관리하기 쉽도록 하나로 묶는 일.

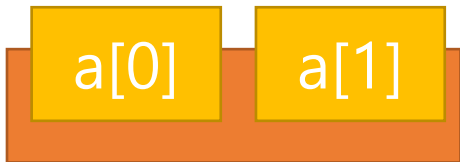
표준국어대사전

```
int a = 777;  
int a2 = 777;
```



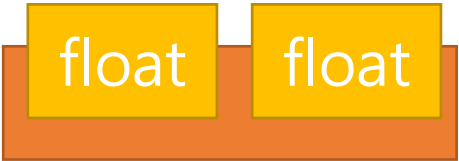
요소(element)

```
int a[2] = { 777, 777 };
```



인덱스(index)

```
float a = 777;  
float a2 = 777;
```



요소(element)

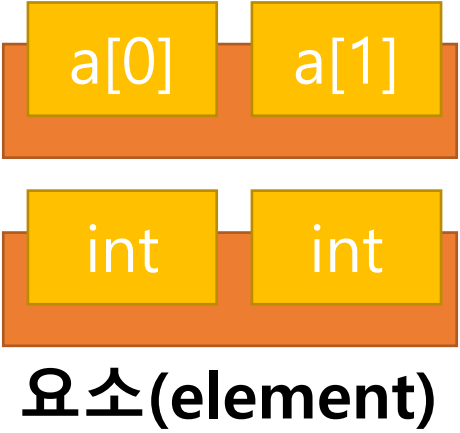


```
int a = 777;  
int a2 = 777;
```

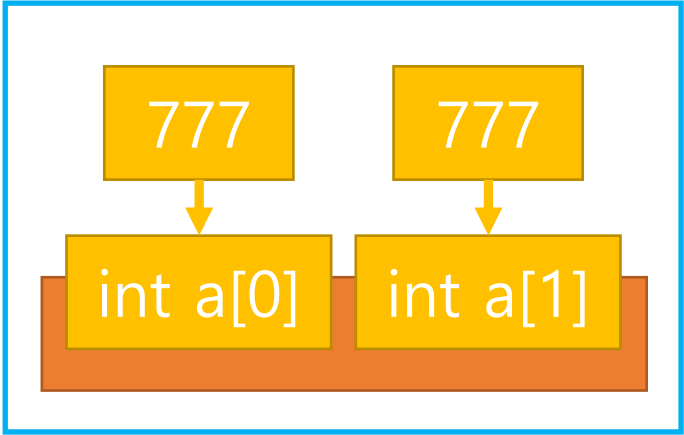
```
float a = 777;  
float a2 = 777;
```

```
int a[2] = { 777, 777 };
```

```
float a[2] = { 777, 777 };
```



```
a[0] = 777;  
a[1] = 777;
```





```
#include <stdio.h>
```

```
int main()  
{
```

```
    int 점수[5];
```

```
    점수[0] = 90;
```

```
    점수[1] = 80;
```

```
    점수[2] = 70;
```

```
    점수[3] = 60;
```

```
    점수[4] = 50;
```

```
    for (int i = 0; i < 5; i++)
```

```
    {
```

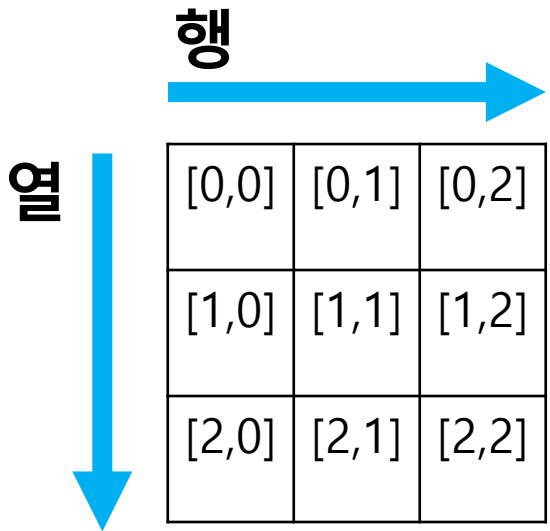
```
        printf("%d 번째 사람의 점수는 %d이다", i + 1, 점수[i]);
```

```
    }
```

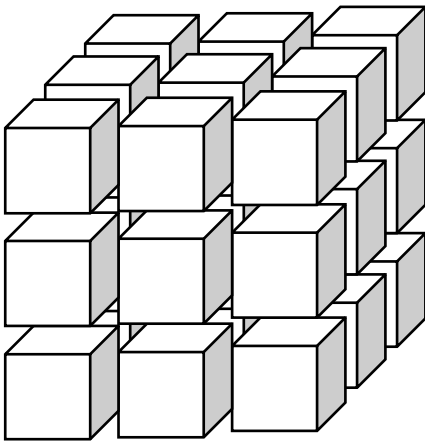
```
}
```

```
int 점수[5] = { 90,80,70,60,50 };
```

```
int 점수[] = { 90,80,70,60,50 };
```



```
int 행열[3][3];
```



```
int _3차원배열[3][3][3];
```

[0,0,0]	[0,0,1]	[0,0,2]	[1,0,0]	[1,0,1]	[1,0,2]
[0,1,0]	[0,1,1]	[0,1,2]	[1,1,0]	[1,1,1]	[1,1,2]
[0,2,0]	[0,2,1]	[0,2,2]	[1,2,0]	[1,2,1]	[1,2,2]

- 문자열과 배열은 관계가 있다.

```
#include <stdio.h>

int main()
{
    char string[6];

    string[0] = 'H';
    string[1] = 'e';
    string[2] = 'l';
    string[3] = 'l';
    string[4] = 'o';

    puts(string);
}
```

열⁴ 列

- 명사 사람이나 물건이 죽 벌여 늘어선 줄.
- 명사 사람이나 물건이 죽 벌여 늘어선 줄을 세는 단위.

표준국어대사전

문자열 文字列 [문짜열]

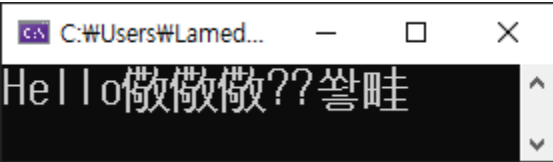
명사 정보-통신 데이터로 다루는 일련의 문자. 숫자를 포함하는 것이어도 계산하는 숫자로서가 아니라 코드 의 일부로 쓰는 것이면 문자열이라 할 수 있다.

표준국어대사전

배열² 配列/排列 [배:열]

- 명사 일정한 차례나 간격에 따라 벌여 놓음.
- 명사 정보-통신 동일한 성격의 데이터를 관리하기 쉽도록 하나로 묶는 일.

표준국어대사전





- 문자열의 끝을 나타낼 때는 'w0'으로 끝을 0으로 해주어야 한다.

```
#include <stdio.h>

int main()
{
    char string[6];

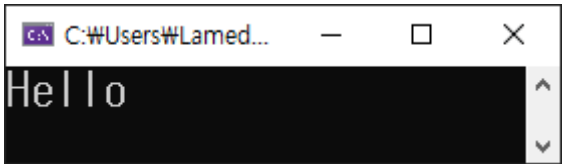
    string[0] = 'H';
    string[1] = 'e';
    string[2] = 'l';
    string[3] = 'l';
    string[4] = 'o';
    string[5] = '\0';

    puts(string);
}
```

```
char string[6] = { 'H','e','l','l','o','\0' };
```

```
char string[6] = "Hello";
```

"문자열" 은 문자열 뒤에 w0인 공백까지 포함하는 규칙이다

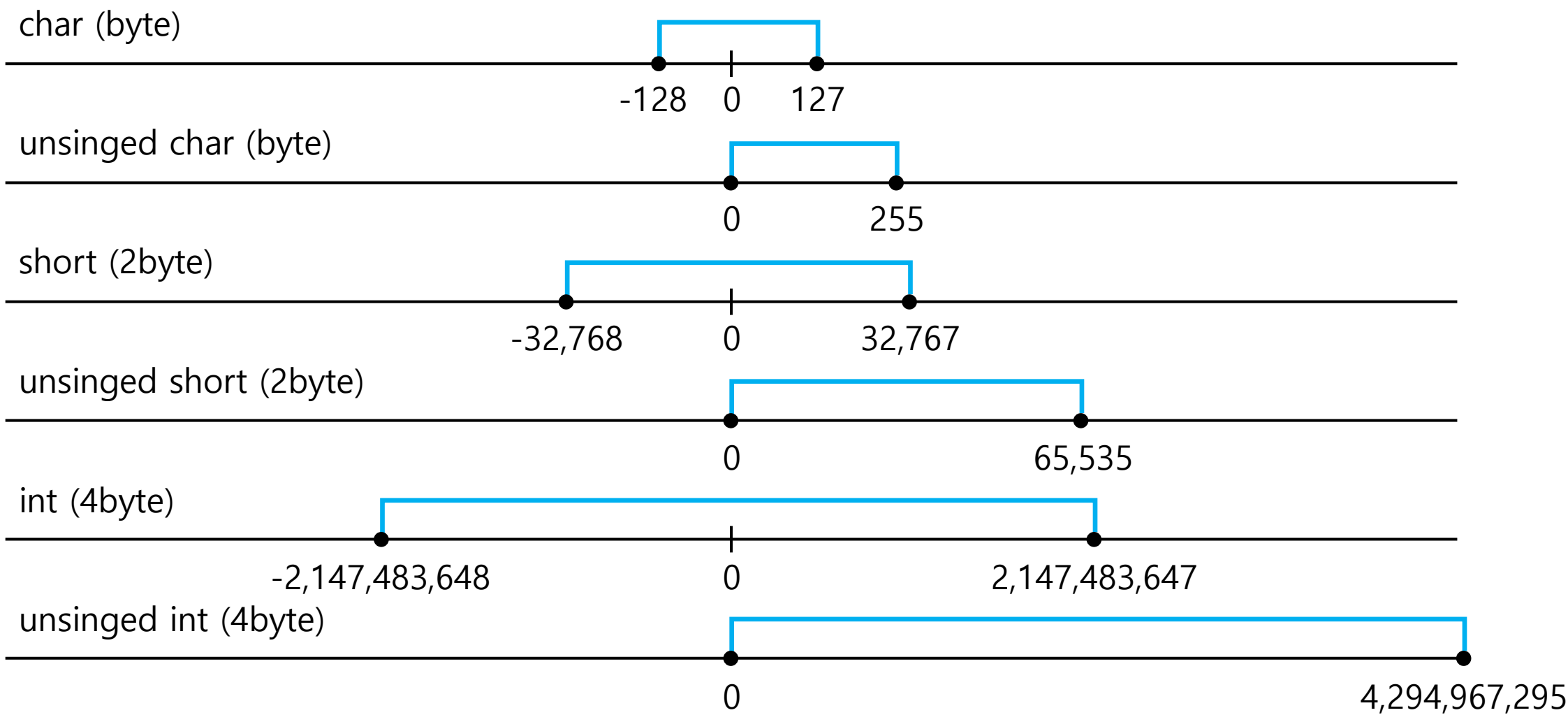




자료형 / 명칭		크기	값의 표현범위
정수형	char	1바이트	-128이상 +127이하
	short	2바이트	-32,768이상 +32,767이하
	int	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long long	8바이트	-9,223,372,036,854,775,808이상
			+9,223,372,036,854,775,807이하
실수형	float	4바이트	$\pm 3.4 \times 10^{-37}$ 이상 $\pm 3.4 \times 10^{38}$ 이하
	double	8바이트	$\pm 1.7 \times 10^{-307}$ 이상 $\pm 3.4 \times 10^{308}$ 이하
	long double	8바이트 이상	double 이상의 표현범위



구분	명칭	설명
부호가 있는 변수	signed	기본(default) 형식
부호가 없는 변수	unsigned	음수를 표현할 수 없고, 양수 값의 표현범위가 두배 정도 늘어남





2진수	10진수
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13

비트(bit)는 0000에서 0같이 한자리를 의미하면 0000 이렇게 8개를 모아 바이트(byte)라고 한다.

1byte = 8bit(0000 0000)

2byte = 16bit(0000 0000 0000 0000)

4byte = 32bit(0000 0000 0000 0000 0000 0000 0000 0000)

8byte = 64bit(0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000)



- 음수는 byte기준으로 가장 앞에 있는 (1000 0000)이렇게 맨 앞자리가 1이면 음수라고 친다.
- 양수는 그냥 계속 숫자가 늘어난다.

기본(signed)

2진수	10진수
0000 0001	1
0000 0010	2
0000 0011	3
0111 1110	126
0111 1111	127
1000 0000	-127
1000 0001	-126
1111 1101	-3
1111 1110	-2
1111 1111	-1

unsigned

2진수	10진수
0000 0001	1
0000 0010	2
0000 0011	3
0111 1110	126
0111 1111	127
1000 0000	128
1000 0001	129
1111 1101	253
1111 1110	254
1111 1111	255



자료형	크기	범위	비고
char signed char	1바이트, 8비트	-128~127	
unsigned char	1바이트, 8비트	0~255	
short short int	2바이트, 16비트	-32,768~32,767	int 생략 가능
unsigned short unsigned short int	2바이트, 16비트	0~65,535	int 생략 가능
int signed int	4바이트, 32비트	-2,147,483,648~ 2,147,483,647	
unsigned unsigned int	4바이트, 32비트	0~4,294,967,295	int 생략 가능
long long int signed long signed long int	4바이트, 32비트	-2,147,483,648~ 2,147,483,647	int 생략 가능
unsigned long unsigned long int	4바이트, 32비트	0~4,294,967,295	int 생략 가능
long long long long int signed long long signed long long int	8바이트, 64비트	-9,223,372,036,854,775,808~ 9,223,372,036,854,775,807	int 생략 가능
unsigned long long unsigned long long int	8바이트, 64비트	0~18,446,744,073,709,551,615	int 생략 가능

<https://docs.microsoft.com/ko-kr/cpp/c-language/c-type-specifiers?view=msvc-160>



- 상수는 자료형 앞에 const를 붙여 사용할 수 있으면 상수는 값을 고정시키고 시킬 때 사용한다.
 - 상수는 변수임으로 공간을 차지한다.
- 고정시킨 const의 값을 바꾸려고 하는 경우 오류는 내보낸다.
 - 하지만 상수 변수는 바꾸자고 노력한다면 바꿀 수 있다.(C언어3)

```
6      const float version = 2.3f;
7
8      version = 3.0f;
```

- #define 이름 값;으로 상수를 지정해서 사용할 수 있다.
 - #define 이름 값은 변수가 아니라서 공간이 없고 그저 숫자를 다른 이름으로 정의해둔 것뿐이다.
- #define 이름 함수로 함수를 간단하게 실행하는 형태로도 사용할 수 있다.
 - 이것도 정해진 함수를 이름만 다른 형태로 할 뿐 같다.
- #define은 매크로로써 보기 좋게 숫자와 간단한 함수를 가공할 수 있다.

```
#define ONE 1;
```

```
3 #define ONE 1;
```

```
#define Hello_World_Function printf("Hello World");
```

```
3 #define Hello_World_Function printf("Hello World");
```



```
#include <stdio.h>

int main()
{
    short short_a = 1;
    int local_a;

    local_a = short_a;
    printf("%d", short_a);

    short_a = local_a + 2;
    printf("%d", short_a);
}
```

- 자주 사용하는 코드를 모아서 한덩어리로 만들고 부를 수 있게 한 작업을 함수라고 합니다.

```
① int ② sum(③ int a, int b)
{
    ④ return a+b;
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값

```
int main()
{
}
```



```
① ② ③  
int sum(int a, int b);
```

```
① ② ③  
int sum(int a, int b)  
{  
  ④  
  return a+b;  
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값

```
int sum(int a, int b);  
  
int main()  
{  
    int a = sum(1, 2);  
}  
  
int sum(int a, int b)  
{  
    return a + b;  
}
```



```
① int ② sum(③ int a, int b)
{
    ④ return a+b;
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값

```
int a = sum(1, 2);
```




```
① int ② sum(③ int a, int b)
{
  ④ return a+b;
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값



```
① int ② sum(③ int a = 0, int b = 0)
{
  ④ return a + b;
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값

```
int a = sum( );
```

```
int a = sum(1, 2);
```



```
① int ② sum(③ int a, int b)
{
  ④ return a+b;
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값

```
① float sum(int a, int b)
{
  return a + b;
}
```



```
inline int sum(int a, int b);
```

```
inline int sum(int a, int b)
{
    return a + b;
}
```



```
#include <stdio.h>

int sum(int a, int b)
{
    return a+b;
}

int main()
{
    int a = sum(1, 1);
    printf("%d\n", a);
    printf("%d\n", sum(1,2));
}
```

```
① int ② sum ③ (int a, int b)
{
    ④ return a+b;
}
```

- ① 반환 형
- ② 함수 이름
- ③ 매개 변수
- ④ 반환값

- 변수는 선언된 블록 안에서만 사용할 수 있고 블록 바깥에서는 사용할 수 없다.

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int a = 10;
```

```
    printf("%d", a);
```

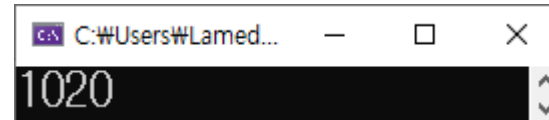
```
{
```

```
    int a = 20;
```

```
    printf("%d", a);
```

```
}
```

```
}
```

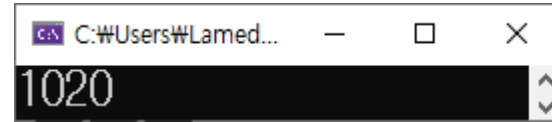


C:\Users\Lamed... 1020

- 변수는 선언된 블록 안에서만 사용할 수 있고 블록 바깥에서는 사용할 수 없다.

```
#include <stdio.h>

int main()
{
    int a = 10;
    printf("%d", a);
    {
        int a = 20;
        printf("%d", a);
    }
}
```



A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Lamed...' and standard window controls. The command prompt displays the number '1020' on a single line, which is the output of the program shown in the code block.



```
int global_a = 3;
```

전역변수

```
int main()  
{
```

```
    int a = 5;  
}
```

지역변수

지역변수는 중괄호 { }안에서만 사용할 수 있다.
전역변수는 전역 즉 전체에서 사용할 수 있다.



```
static int 정적변수;
```



```
#include <stdio.h>

int main()
{
    int* 동적변수;
    동적변수 = (int*)malloc(sizeof(int));
}
```



```
typedef int 정수;  
정수 변수;
```



```
enum Numbers {  
    One,  
    Two,  
    Three,  
    Four,  
    Five,  
    Six,  
    Seven  
};
```

```
enum Numbers {  
    One = 1,  
    Two,  
    Three,  
    Four,  
    Five,  
    Six,  
    Seven  
};
```



```
struct positon  
{  
    int x, y;  
};
```



```
int main( )
{
    typedef struct positon
    {
        int x, y;
    }POS;
    POS 위치;
}
```

```
typedef struct positon
{
    int x, y;
}POS;
```



```
union positon  
{  
    int x, y;  
};
```



```
#include <stdio.h>
```

```
int main()  
{
```

```
    // 정수형 변수 선언
```

```
    int a1;
```

```
    int a2;
```

```
    int a3;
```

```
    // 변수에 값 할당(저장)
```

```
    a1 = 1;
```

```
    a2 = 2;
```

```
    a3 = 3;
```

```
    printf("%d %d %d\n", a1, a2, a3); //변수에 저장된 값을 %d로 출력
```

```
    return 0;
```

```
}
```

할당



a1 = 1;



```
#include <stdio.h>
```

```
int main()  
{
```

```
    // 변수를 콤마(,)로 구분하여 변수 여러 개를 선언  
    int a1, a2, a3;
```

```
    // 변수에 값 할당
```

```
    a1 = 1;
```

```
    a2 = 2;
```

```
    a3 = 3;
```

```
    a1 = 1, a2 = 2, a3 = 3;
```

```
    //변수에 저장된 값을 %d로 출력
```

```
    printf("%d %d %d\n", a1, a2, a3);
```

```
    return 0;
```

```
}
```



```
#include <stdio.h>

int main()
{
    // 변수를 선언하면서 값 할당(초기화)
    int a1 = 1;

    // 변수 여러 개를 선언하면서 값 할당(초기화)
    int a2 = 2, a3 = 3;

    // 변수에 저장된 값을 %d로 출력
    printf("%d %d %d\n", a1, a2, a3);

    return 0;
}
```



- 변수 여러 개를 선언하면서 값 초기화하기

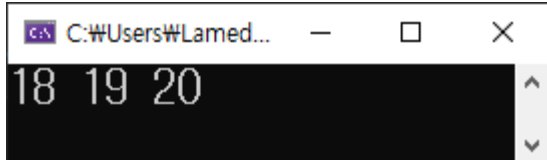
```
#include <stdio.h>

int main()
{
    [REDACTED]

    printf("%d %d %d\n", a1, a2, a3);

    return 0;
}
```

출력결과



C:\Users\Lamed...
18 19 20



```
#include <stdio.h>

int main()
{
    int a1 = 18, a2 = 19, a3 = 20;

    printf("%d %d %d\n", a1, a2, a3);

    return 0;
}
```



```
#include <stdio.h>

int main()
{
    int a;           // int 형 변수 a 선언
    int b, c;        // 2개의 int형 변수 b,c를 동시에 선언
    float f;         // float형 변수 f 선언
    char ch;         // char형 변수 ch 선언

    a = 10;          // int형 변수 a에 정수 10 대입
    b = a;           // int형 변수 b에 변수 a의 값 대입
    c = a + 20;       // int형 변수 c에 변수 a의 값과 정수 20에 더한 값 대입
    f = 3.5;         // float형 변수 f에 실수 3.5 대입
    ch = 'A';        // char형 변수 ch에 문자 'A' 대입

    printf("a = %d\n", a);
    printf("b = %d\n", b);
    printf("c = %d\n", c);
    printf("f = %.1f\n", f);
    printf("ch = %c\n", ch);

    return 0;
}
```

```
C:\Users\Lamed...
a = 10
b = 10
c = 30
f = 3.5
ch = A
```



변수

리터럴, 변수, 수식

```
a1 = 1;  
a2 = a1;  
a3 = a1 + 1;
```

변수에 리터럴 대입

변수에 변수 대입

변수에 수식 대입

- 리터럴(literal)이란 소스 코드의 고정된 값을 대표하는 용어다.(출처 나무위키)

- 리터럴의 종류는

<https://ko.wikipedia.org/wiki/리터럴>

- 숫자 리터럴
- 문자 리터럴
- 문자열 리터럴

```
int a = 1;  
char b = 'a';  
char c[5] = "ABCD";
```

```
printf( "%d\n", 7 );  
printf( "%f\n", 7.6f );  
printf( "%c\n", 'L' );  
printf( "%s\n", "Hello world!" );
```

```
int a = 777;  
int a_V2 = 36'000'000;  
int a2 = 0777;  
int a3 = 0x777;  
int a4 = 0b0001;  
float a5 = 0.7f;  
float a5_V2 = 0.7e+1f;  
double a6 = 0.9;  
double a6_V2 = 0.9l;
```

```
printf("%d\n", 777);  
printf("%d\n", 36'000'000);  
printf("%o\n", 0777);  
printf("%x\n", 0x777);  
printf("%d\n", 0b0001);  
printf("%f\n", 0.7f);  
printf("%f\n", 0.7e+1f);  
printf("%lf\n", 0.9);  
printf("%lf\n", 0.9l);
```




- 실수인 float double등에서는
- e+? / e-? 를 이용한 지수 표기부가 있다.
- 3e-1f는 0.3f이다
- 3e-2f는 0.03f이다
- 3e+1f는 30f이다
- 3e+2f는 300f이다
- $3^{+n} = 3e+n$, $3^{-n} = 3e-n$
 - 지수 표기법으로 표기할 때는 정수 부분은
 - 한 자릿수만 적고, 소수자릿수
 - 뒤에 e와 지수를 표기합니다.

<https://docs.microsoft.com/ko-kr/cpp/cpp/numeric-boolean-and-pointer-literals-cpp?view=msvc-160>

접미사	자료형
생략	int
l, L	long
u, U	unsigned int
ul, UL	unsigned long
ll, LL	long long
ull, ULL	unsigned long long
f, F	float
l, L	long double



```
#include <stdio.h>
```

```
int main()  
{
```

```
    char ch = 'A';           // char형 변수 ch에 문자 'A' 대입 선언
```

```
    printf("ch = %d\n", ch);
```

```
    printf("ch = %c\n", ch);
```

```
    return 0;
```

```
}
```

```
C:\Users\Lamed...  
ch = 65  
ch = A
```

진법/진수

16진수 10진수 2진수



Printable ASCII characters

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	
2-	^{SPACE}	!	"	#	\$	%	&	'	()	*	+	,	-	.	/	2-
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	3-
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	4-
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_	5-
6-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	6-
7-	p	q	r	s	t	u	v	w	x	y	z	{		}	~	✕	7-
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	

Hexadecimal 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21

Hello, World!



Hello, World!

Decimal 72 101 108 108 111 44 32 87 111 114 108 100 33

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	
3-	×	×	^{SPACE}	!	"	#	\$	%	&	'	3-
4-	()	*	+	,	-	.	/	⁴⁸ 0	1	4-
5-	2	3	4	5	6	7	8	9	:	;	5-
6-	<	=	>	?	@	⁶⁵ A	B	C	D	E	6-
7-	F	G	H	I	J	K	L	M	N	O	7-
8-	P	Q	R	S	T	U	V	W	X	Y	8-
9-	Z	[\]	^	_	`	⁹⁷ a	b	c	9-
10-	d	e	f	g	h	i	j	k	l	m	10-
11-	n	o	p	q	r	s	t	u	v	w	11-
12-	x	y	z	{		}	~	×	×	×	12-
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	



$1234.56 =$
 $1 \times 1000 +$
 $2 \times 100 +$
 $3 \times 10 +$
 $4 \times 1 +$
 $5 \times 0.1 +$
 $6 \times 0.01 =$
 1×10^3
 2×10^2
 3×10^1
 4×10^0
 5×10^{-1}
 6×10^{-2}

$1010_{(2)} =$
 1×2^3
 0×2^2
 1×2^1
 0×2^0

$1010_{(8)} =$
 1×8^3
 0×8^2
 1×8^1
 0×8^0

$ABC1_{(16)} =$
 $A \times 16^3$
 $B \times 16^2$
 $C \times 16^1$
 1×16^0

$???_{(n)} =$
 $? \times n^2$
 $? \times n^1$
 $? \times n^0$

$0_{(16)} = 00_{(10)}$
 $1_{(16)} = 01_{(10)}$
 $2_{(16)} = 02_{(10)}$
 $3_{(16)} = 03_{(10)}$
 $4_{(16)} = 04_{(10)}$
 $5_{(16)} = 05_{(10)}$
 $6_{(16)} = 06_{(10)}$
 $7_{(16)} = 07_{(10)}$
 $8_{(16)} = 08_{(10)}$
 $9_{(16)} = 09_{(10)}$
 $A_{(16)} = 10_{(10)}$
 $B_{(16)} = 11_{(10)}$
 $C_{(16)} = 12_{(10)}$
 $D_{(16)} = 13_{(10)}$
 $E_{(16)} = 14_{(10)}$
 $F_{(16)} = 15_{(10)}$

$9_{(10)} =$

$$\begin{array}{r} 2 \overline{) 9 \dots 1} \\ 2 \overline{) 4 \dots 0} \\ 2 \overline{) 2 \dots 0} \\ 2 \overline{) 1 \dots 1} \end{array}$$



$1001_{(2)}$

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



2진수



10진수



16진수



10진수	2진수	8진수	16진수
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F



- 10진수 경우
 - 0123456789를 다 쓴 경우 자리가 올라가서 10이 된다
- 2진수 경우
 - 01 두개로 한계치까지 다 쓴 경우 자릿수가 올라간다
- 8진수 경우
 - 01234567을 다 쓴 경우 자릿수가 올라간다
- 16진수 경우
 - 0123456789ABCDEF를 다 쓴 경우 자릿수가 올라간다

10진수	2진수	8진수	16진수
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10



HEXADECIMAL

53 = 35

0 1 2 3 4 5 6 7 8 9 10 11 12

35 99 100 563

5x10x10 6x10 3

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

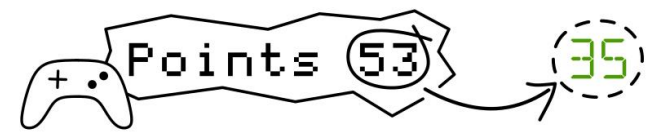
35 = 53

FF = 255

100 = 256

3E8 = 1000

2017/08/18
@ANGEALBERTINI
CORKAMI.COM





비트 연산자	설명
&	and
	or
^	xor
~	not
<<	left shift
>>	right shift

A	B	and	or	xor
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0



기호	이름
&=	and비트 연산후 대입
^=	xor비트 연산후 대입
=	or비트 연산후 대입
<<=	좌시프트 연산후 대입
>>=	우시프트 연산후 대입

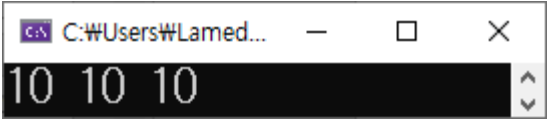


10진수	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16진수	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8진수	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17

C언어식 진법 표현법

10진수	0	1
16진수	0x0	0x1
8진수	00	01

출력결과



```
#include <stdio.h>

int main()
{
    int a1 = 10, a2 = 0xa, a3 = 012;
    printf("%d %d %d", a1, a2, a3);
    return 0;
}
```

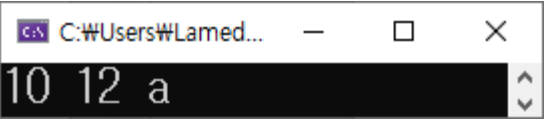


10진수	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16진수	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8진수	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17

C언어식 진법 표현법

10진수	0	1
16진수	00	01
8진수	0x0	0x1

출력결과



```
#include <stdio.h>

int main()
{
    //d는 decimal[10진수]
    //o는 octal[8진수]
    //h는 hex(hexadecimal)[16진수]

    int a1 = 10;
    printf("%d %o %x", a1, a1, a1);

    return 0;
}
```



서식 문자	대응 변수	출력
%d	char(정수일때), short, int	부호를 포함한 10진수
%u	unsigned int	부호없는 10진수
%o	unsigned int	부호없는 8진수
%x,%X	unsigned int	부호없는 16진수 (포맷 스트링의 대문자 소문자에따라 A이상의 값 대소문자 결정)
%f	float, double	10진수 방식의 부동 소수점 실수
%e	float, double	부동 소수점 실수 (지수형태 표기, 대소문자에 따라 E의 대소문자 결정)
%g,G	float, double	값에 따라서 %f를 쓰거나 %e 를 선택 (역시나 대소문자에 따라서 대소문자 결정)
%c	char(문자일때)	문자 출력
%s	char*	문자열(문자 집합)
%p	void*	포인터 주소값

형식 문자	인수
c	문자
C	문자
d	정수
i	정수
o	정수
u	정수
x	정수
X	정수
e	부동 소수점
E	부동 소수점
f	부동 소수점
F	부동 소수점
g	부동 소수점
G	부동 소수점
a	부동 소수점
A	부동 소수점
n	정수 포인터
p	포인터 유형
s	String
S	String
Z	ANSI_STRING 또는 UNICODE_STRING 구조체

출력 형식

printf 함수와 함께 사용될 때 단일 바이트 문자를 지정하고, wprintf 함수와 함께 사용될 때 와이드 문자를 지정합니다.

printf 함수와 함께 사용될 때 와이드 문자를 지정하고, wprintf 함수와 함께 사용될 때 단일 바이트 문자를 지정합니다.

부호 있는 10진수 정수입니다.

부호 있는 10진수 정수입니다.

부호 없는 8진수 정수입니다.

부호 없는 10진수 정수.

부호 없는 16 진수 정수 " abcdef "를 사용 합니다.

부호 없는 16 진수 정수 " ABCDEF "를 사용 합니다.

형식이 [-] *d*.*dddd* **e** ± 인 부호 있는 값입니다 *dd* [*d* . 여기서 *d* 는 10 진수 이며,은 *dddd* 지정 된 전체 자릿수에 따라 하나 이상의 10 진수이 고,은 숫자 *dd* [*d* 의 [출력 형식](#) 및 크기에 따라 두 자리 또는 세 개의 10 진수입니다.

지 수를 **e** 도입 한다는 점을 제외 하 고는 형식과 동일 **E e** 합니다.

형식이 [-] . 인 부호 있는 값입니다 *dddd* . *dddd* . 여기서 *dddd* 는 하나 이상의 10 진수입니다. 소수점 앞의 자릿수는 수의 크기에 따라 다르며, 소수점 뒤의 자릿수는 요청된 전체 자릿수에 따라 다릅니다. 기본 자릿수는 6입니다.

f Infinity 및 nan 출력의 대문자를 제외 하 고는 형식과 동일 합니다.

부호 있는 값은 또는 형식으로 표시 되며 **f e** , 지정 된 값과 전체 자릿수에 대해 더 압축 됩니다. **e** 형식은 값의 지수가-4 보다 작거나 ~~전체 자릿수~~ 인수 보다 크거나 같은 경우에만 사용 됩니다. 뒤에 나오는 0은 잘리고, 소수점은 뒤에 하나 이상의 수가 나오는 경우에만 나타 납니다.

g E 보다는 (해당 하는 경우) 지 수를 도입 한다는 점을 제외 하 고는 형식과 동일 **e** 합니다.

형식이 [-] *0xhp* ± 인 부호 있는 16 진수 배정밀도 부동 소수점 값입니다. *dd* 여기서 *hhhh* 는가 수의 16 진수 (소문자 사용) 이며 *dd* 지 수의 자릿수는 하나 이상입니다. 정밀도는 소수점 뒤의 자릿수를 지정합니다.

형식이 [-] *0xHP* ± 인 부호 있는 16 진수 배정밀도 부동 소수점 값입니다. *dd* 여기서 *hhhh* 는가 수의 16 진수 (대문자 사용)이 고 *dd* 는 지 수에 대 한 하나 이상의 숫자입니다. 정밀도는 소수점 뒤의 자릿수를 지정합니다.

지금까지 성공적으로 스트림 또는 버퍼에 쓴 문자의 수입니다. 이 값은 주소가 인수로 지정된 정수에 저장됩니다. 가리키는 대상의 정수 크기는 인수 크기의 사양 접두사로 컨트롤할 수 있습니다. **n** 지정자는 기본적으로 사용 하지 않도록 설정 되어 있습니다. 자세한 내용은 중요 보안 정보를 참조 하세요.

인수를 16 진수로 된 주소로 표시 합니다.

printf 함수와 함께 사용될 때 단일 바이트 또는 멀티바이트 문자열을 지정하고, wprintf 함수와 함께 사용될 때는 와이드 문자열을 지정합 니다. 첫 번째 null 문자 직전까지 또는 *precision* 값에 도달할 때까지 문자가 표시됩니다.

printf 함수와 함께 사용될 때 와이드 문자열을 지정하고, wprintf 함수와 함께 사용될 때는 단일 바이트 또는 멀티바이트 문자열을 지정합 니다. 첫 번째 null 문자 직전까지 또는 *precision* 값에 도달할 때까지 문자가 표시됩니다.

[ANSI_STRING](#) 또는 구조체의 주소가 [UNICODE_STRING](#) 인수로 전달 되 면 구조체의 필드가 가리키는 버퍼에 포함 된 문자열을 표 시 Buffer 합니다. 의 *size* 한정자 접두사를 사용 **w** 하 여 인수를 지정 UNICODE_STRING 합니다 (예:) %wZ . 구조체의 Length 필드를 문 자열의 길이(바이트 단위)로 설정해야 합니다. 구조체의 MaximumLength 필드를 버퍼의 길이(바이트 단위)로 설정해야 합니다.

일반적으로 **Z** 형식 문자는 및와 같은 변환 사양을 사용 하는 드라이버 디버깅 함수 에서만 사용 됩니다 dbgPrint kdPrint .

<https://docs.microsoft.com/ko-kr/cpp/c-runtime-library/format-specification-syntax-printf-and-wprintf-functions?view=msvc-160>

삼항 연산자 if switch while for
조건식



```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>
```

scanf 경고 무시용

```
int main()  
{
```

```
    int a;  
    float b;  
    char c;
```

```
    scanf("%d %f", &a, &b);  
    scanf("%c", &c);  
    printf("%d %f %c", a, b, c);
```

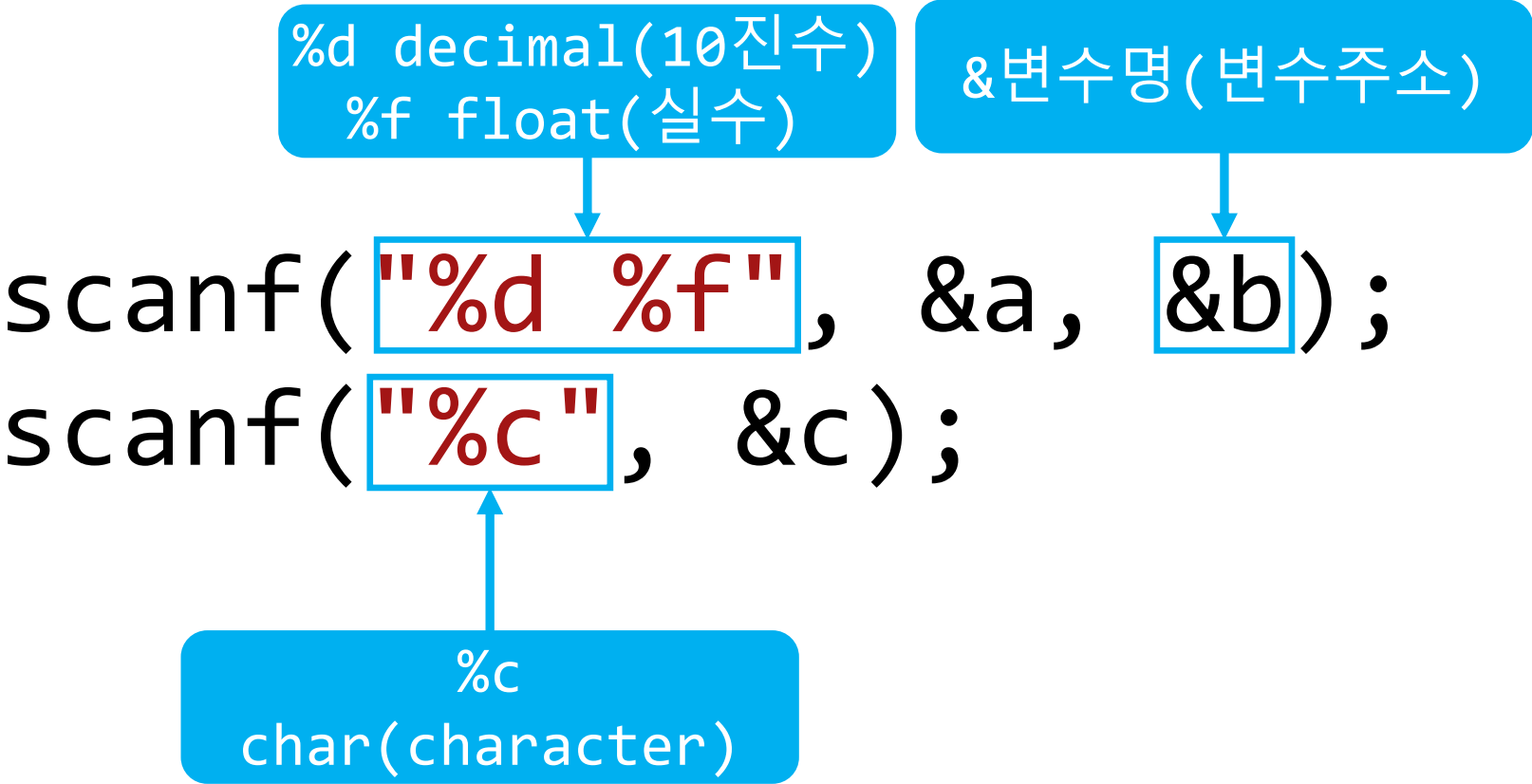
scanf함수

```
}
```





- 문자열 뒤에 ,뒤부분을 보면 변수의 주소를 가리키는 &변수명으로 사용한다
- 앞에 서식(format)을 스캔해서 뒤에 &변수명(변수의주소)로 값을 입력 받는 함수이다





연산자	설명	예제
==	같음(equal)	num1 == num2
!=	같지 않음(not equal)	num1 != num2
>	큼(greater)[above]	num1 > num2
<	작음(less)[below]	num1 < num2
>=	크거나 같음(greater or equal)	num1 >= num2
<=	작거나 같음(less or equal)	num1 <= num2

<https://docs.microsoft.com/ko-kr/cpp/c-language/c-relational-and-equality-operators?view=msvc-160>



```
#include <stdio.h>

int main()
{
    int a = 1;

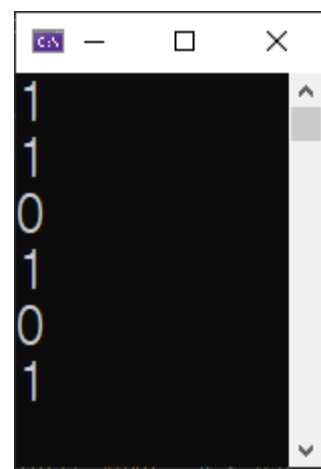
    printf( "%d\n", a == 1);
    printf( "%d\n", a != 2);

    printf( "%d\n", a > 3);
    printf( "%d\n", a < 4);

    printf( "%d\n", a >= 5);
    printf( "%d\n", a <= 6);

    return 0;
}
```

출력결과





- 거짓의 정의된 값은 0이다.
 - 참인 값은 0이 아닌 값 즉 거짓(0)이 아닌 모든 값(숫자)은 참이 된다.
 - 참(true)의 정의된 값은 1이다

```
#include <stdbool.h>

int main()
{
    int a = true;
    int a2 = false;

    return 0;
}
```

```
//
// stdbool.h
//
//      Copyright (c) Microsoft Corporation. All rights reserved.
//
// The C Standard Library <stdbool.h> header.
//
#ifndef _STDBOOL
#define _STDBOOL

#define __bool_true_false_are_defined 1

#ifndef __cplusplus

#define bool _Bool
#define false 0
#define true 1

#endif /* __cplusplus */

#endif /* _STDBOOL */
```

- 조건식 ? 실행문(참) : 실행문(거짓);

- 예시

`1 == 1 ? printf("1은 1이다"); : printf("1은 1이아니다");`

`1 > 2 ? 1 : 2;`

`a > b ? a : b;`

```
#include <stdio.h>

int main()
{
    int num1 = 1;

    printf("%s\n", num1 == 1 ? "TRUE" : "FALSE");

    return 0;
}
```



```
if(조건식)
{
    코드1
    코드2
}

if(조건식)
    조건식이 참(true)일 때 실행될 코드

else if(조건식)
{
    위에 if 조건식이 거짓(false)이고
    밑에 if문이 참(true)일 때 실행될 코드
}

else(조건식)
    위에 if or else if
    조건식이 거짓(false)일 때 실행될 코드
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int num = 0;
    scanf("%d", &num);

    if (num == 1)
        printf("1\n");
    else if (num == 2)
        printf("2\n");
    else if (num)
        printf("%d\n", num);
    else
        printf("0\n");

    return 0;
}
```




```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int num = 0;
    scanf("%d", &num);

    if (num == 1)
        printf("1\n");
    else if (num == 2)
        printf("2\n");
    else if (num)
        printf("%d\n", num);
    else
        printf("0\n");

    return 0;
}
```

scanf 경고 무시용

정수형 num변수 0 할당

num변수의 값이 1일 경우

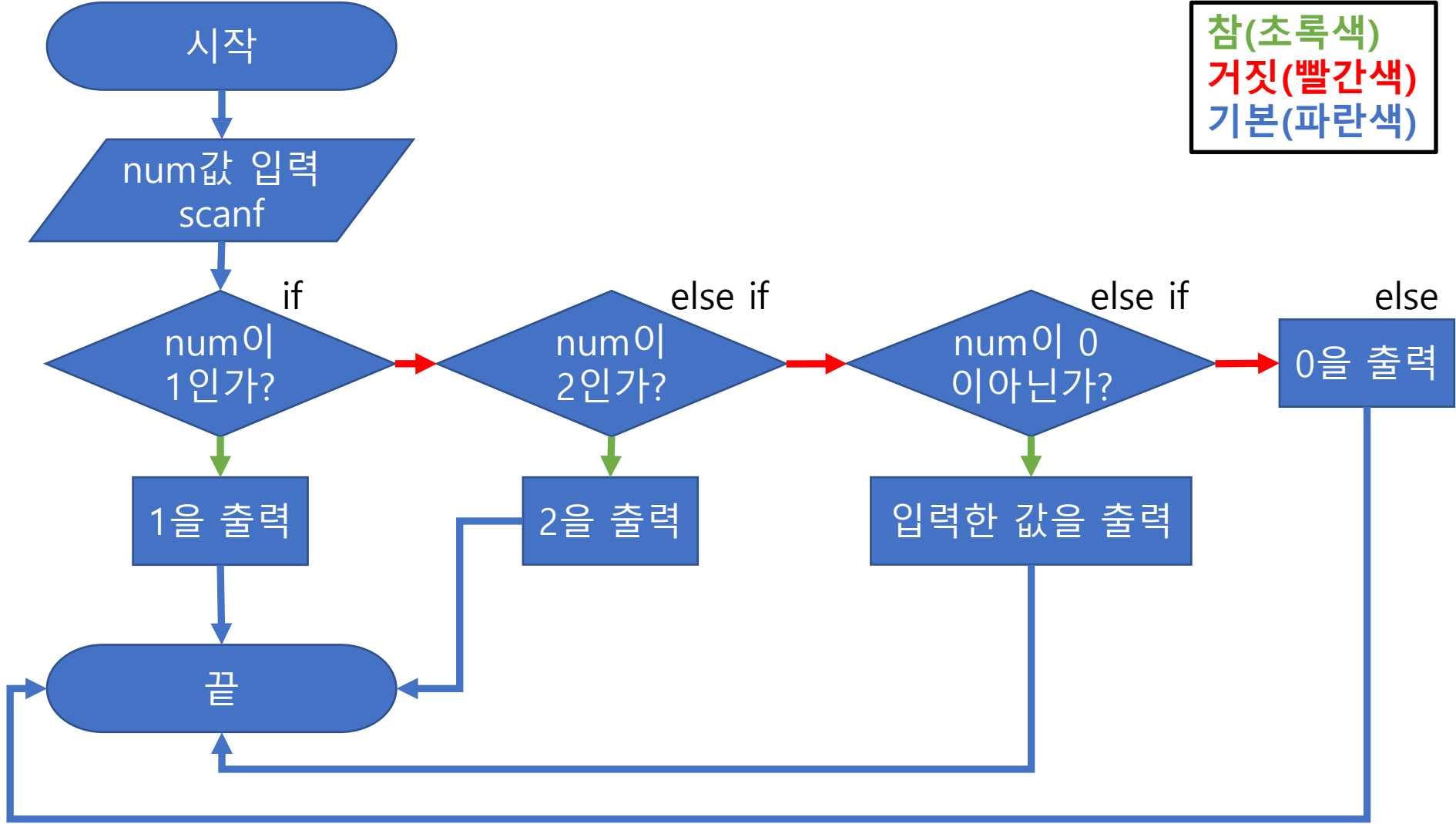
num변수의 값이 2일 경우

num변수의 값이 0이 아닌 경우

위에 조건들이 전부 아닌 경우



참(초록색)
거짓(빨간색)
기본(파란색)





```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int num = 0;
    scanf("%d", &num);

    if (num == 1)
        printf("1\n");
    else if (num == 2)
        printf("2\n");
    else if (num)
        printf("%d\n", num);
    else
        printf("0\n");

    return 0;
}
```

참(초록색)
거짓(빨간색)
기본(파란색)



A	B	A && B	A B	!A
1 (true)	1 (true)	1 (true)	1 (true)	0 (false)
1 (true)	0 (false)	0 (false)	1 (true)	0 (false)
0 (false)	1 (true)	0 (false)	1 (true)	1 (true)
0 (false)	0 (false)	0 (false)	0 (false)	1 (true)

연산자	설명
&&	AND
	OR
!	NOT



```
switch(변수(값))
{
case 1: 변수의 값이 1인 경우(변수 == 1)
    코드;
    break;
case 2: 변수의 값이 2인 경우(변수 == 2)
    코드;
    break;
default: 위에 조건들이 다 아닌 경우
    코드;
}
```

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;

    scanf("%d", &a);

    switch (a)
    {
        case 1:
            printf("1\n");
            break;
        case 2:
            printf("2\n");
            break;
        default:
            printf("default\n");
            break;
    }

    return 0;
}
```



```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;

    scanf("%d", &a);

    switch (a)
    {
        case 1:
            printf("1\n");
            break;
        case 2:
            printf("2\n");
            break;
        default:
            printf("default\n");
            break;
    }

    return 0;
}
```

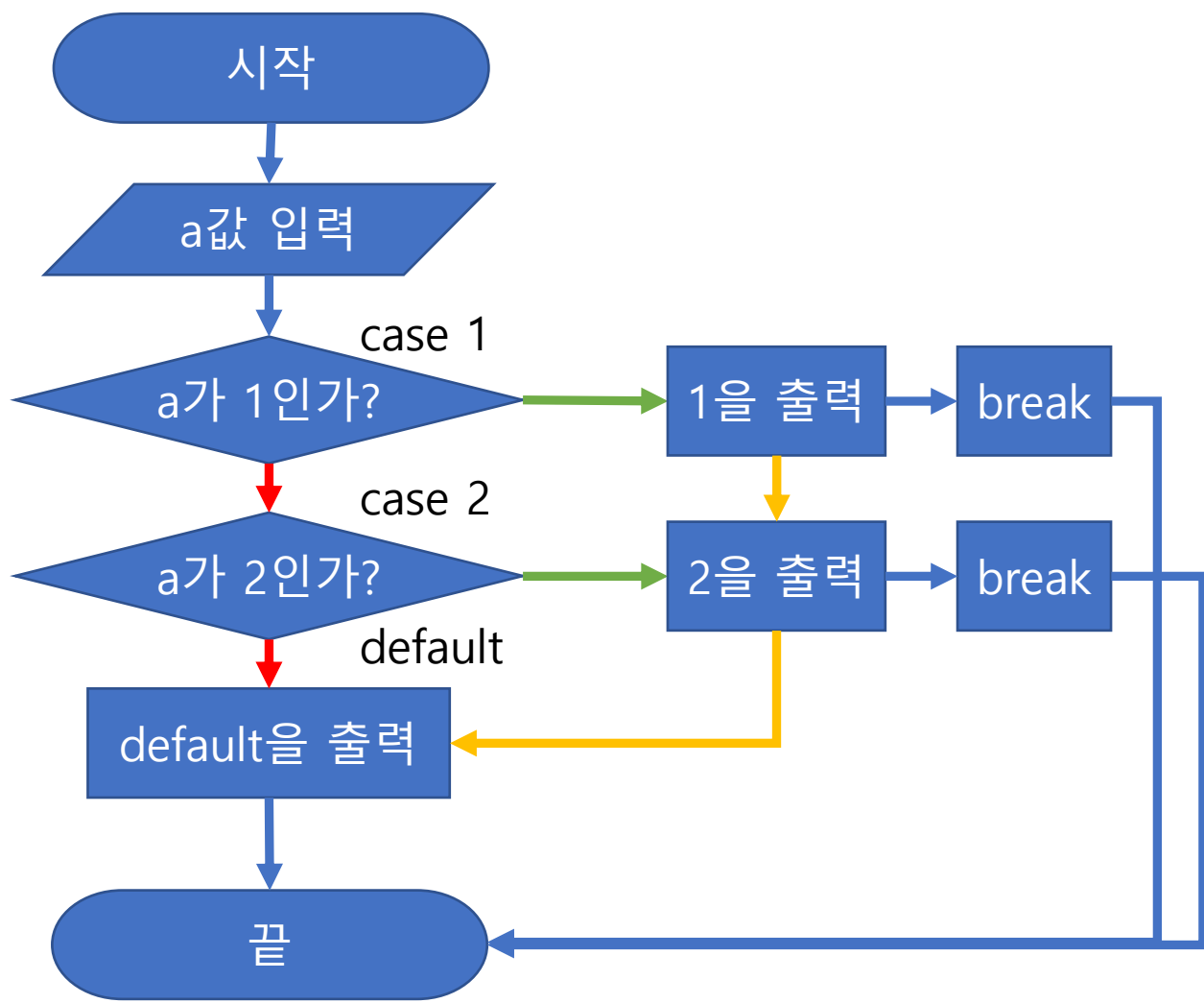
scanf 경고 무시용

a에 10진수로 값 대입

num변수의 값이 1일 경우

num변수의 값이 2일 경우

위에 조건들이 전부 아닌 경우



참(초록색)
거짓(빨간색)
기본(파란색)
break가 없는 경우
(노란색)



```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;

    scanf("%d", &a);

    switch (a)
    {
        case 1:
            printf("1\n");
            break;
        case 2:
            printf("2\n");
            break;
        default:
            printf("default\n");
            break;
    }

    return 0;
}
```

참(초록색)
거짓(빨간색)
기본(파란색)

break는 switch문을 끝낸다



break



continue



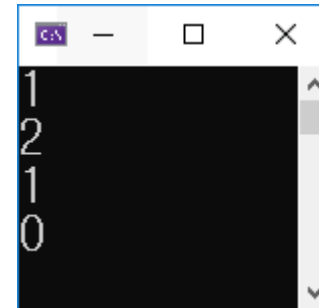




```
#include <stdio.h>

int main()
{
    int a = 0;
    a++;
    printf("%d\n", a);
    ++a;
    printf("%d\n", a);
    a--;
    printf("%d\n", a);
    --a;
    printf("%d\n", a);
}
```

출력결과



++ 변수명
-- 변수명
변수명++
변수명--
++은 값을 1증가시키고
--는 값을 1감소시킨다



while(조건식) 조건이 거짓이 될 때 까지 실행

```
{  
    코드;  
}
```

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>
```

```
int main()  
{  
    char a;  
  
    scanf( "%d", &a);  
  
    while (a)  
    {  
        printf( "%d\n", a-- );  
    }  
}
```



```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    char a;

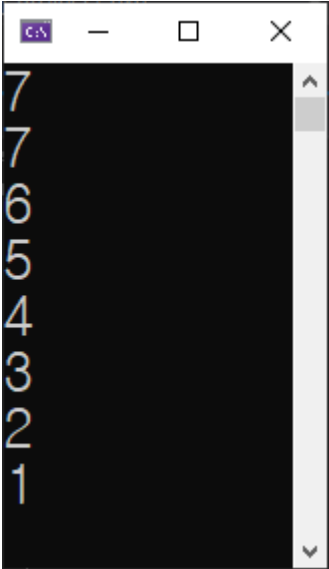
    scanf("%d", &a);

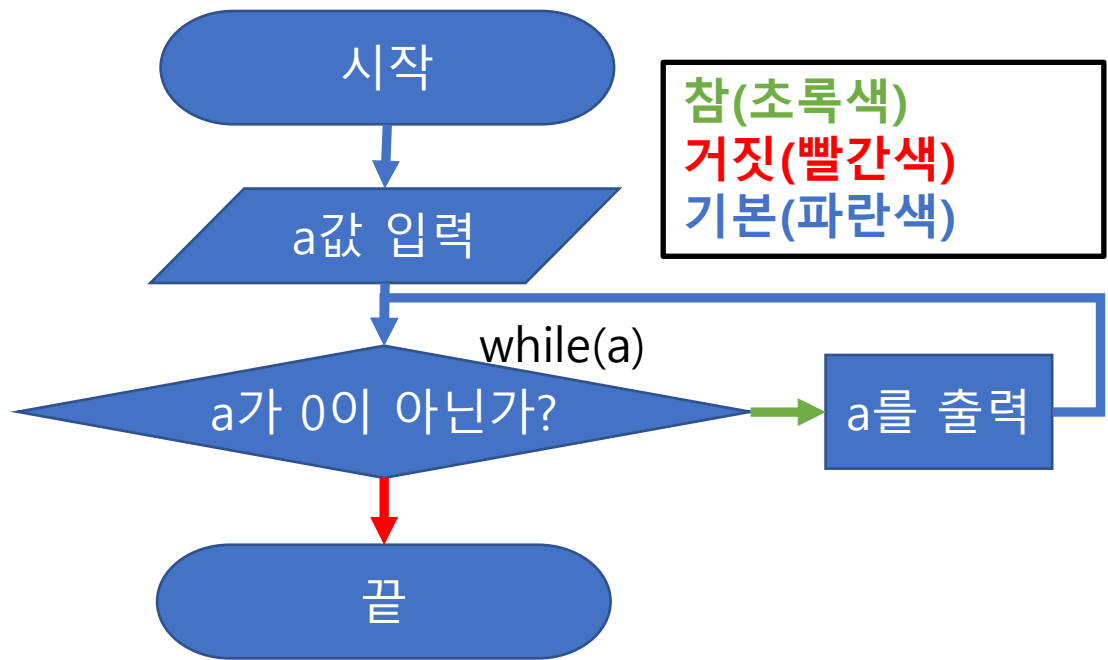
    while (a)
    {
        printf("%d\n", a--);
    }
}
```

a에 10진수로 값 대입

a변수의 값이 0이 아닌 경우

출력결과









```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main()
{
```

```
    char a;
```

```
    scanf( "%d", &a );
```

```
    while (a)
```

```
    {
```

```
        printf( "%d\n", a-- );
```

```
    }
```

```
}
```

참(초록색)
거짓(빨간색)
기본(파란색)



for(초기식; 조건식; 증감식)
조건이 거짓이 될 때 까지 실행
{
 코드;
}

for(; ; 증감식)
초기식과 조건식은 쓰지 않고 ;을 사용해 생략할 수도 있다.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;

    scanf("%d", &a);

    for (int i = 0; i < a; i++)
    {
        printf("%d\n", i);
    }

}
```



```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main()
{
    int a;

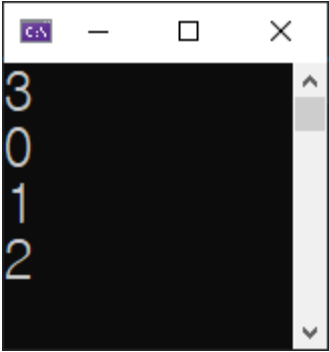
    scanf("%d", &a);

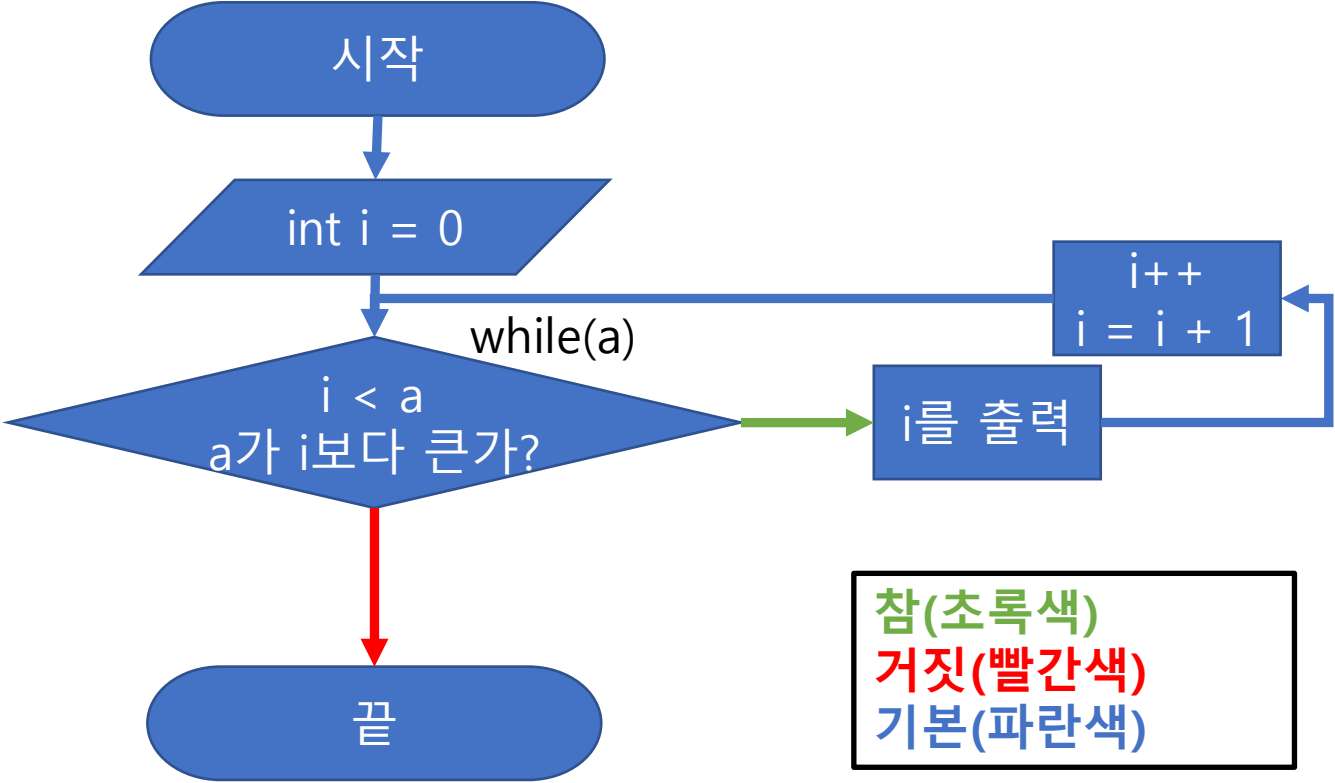
    for (int i = 0; i < a; i++)
    {
        printf("%d\n", i);
    }
}
```

a에 10진수로 값 대입

i가 a보다 작을 경우

출력결과







```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
```

```
int main()
{
```

```
    int a;
```

```
    scanf( "%d", &a );
```

```
    for (int i = 0; i < a; i++)
```

```
    {
```

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

```
    }
```

```
}
```

참(초록색)
거짓(빨간색)
기본(파란색)



- 포인터는 자료형* 형태로 *을 붙여서 만들 수 있습니다.
- 포인터에 값을 넣을 때는 *변수로 입력받을 수 있다

```
int main()  
{
```

```
    int a;
```

```
    int* pointer_a;
```

```
    a = 5;
```

```
    pointer_a = &a;
```

```
    *pointer_a = 1;
```

```
}
```

포인터int 선언

주소접근해서 값 대입











