C언어 학습! (핵심편)

C 학습 – (역사 ~ 연산자)

작성자: 이재선

세명컴퓨터고등학교
SEMYEONG COMPUTER HIGH SCHOOL

C의 역사, 그리고 C언어의 특징

01

C는 AT&A Bell Labs의 Dennis Ritchie가 1972년에 개발
기존 언어인 B 언어를 개선하여 만들어졌으며
현재까지도 시스템 프로그래밍의 표준 언어로 사용됨

C++, Java, C#, Python 등 다수 언어의 기반이 됨

★ 빠른 실행 속도(하드웨어와 가까운 저수준의 언어) → 동작 방식이 저수준에 가까움

★ 메모리 직접 제어 가능(포인터 사용) → 성능 최적화 가능

★ 절차지향 프로그래밍 언어

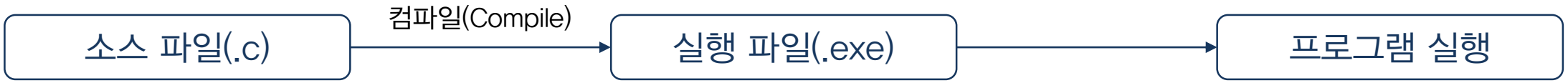
□ 시스템 개발에 최적화

↳ OS, 드라이버, 임베디드 시스템에 널리 사용

메모리 직접 관리로 인해
학습 난이도와 개발 난이도가 높은 언어!

02

우리가 작성한 C 코드는 바로 실행될까?



컴파일?

★ 컴퓨터는 C 문법을 직접 이해하지 못하고 기계어(Binary)만 처리할 수 있음

★ 기계어로 번역하는 역할을 해주는 것이 '컴파일러(Compiler)'이다

C언어 개발 환경 구축의 첫 단계는 컴파일러 설치

C 컴파일러 설치하기

03

① https://winlibs.com/ 사이트 접속

Unless you are targetting older versions of Windows, UCRT as runtime library is the better choice, Upgrade your code to the Universal CRT).

Download

The following downloads are available (for Windows only).
*You will need a decompressor like 7-Zip (free) to unzip .7z archives, but they are a lot smaller tha

Each version comes in 2 flavors:

- Win32 - i686 - Windows 32-bit version, runs natively on and compiles for Windows 32-bit (alsc
- Win64 - x86_64 - Windows 64-bit version, runs natively on and compiles for Windows 64-bit (

Help! I don't know which download to choose!

Don't worry. For most purposes the latest Windows 64-bit release version with MSVCRT runtime a

Download it here.

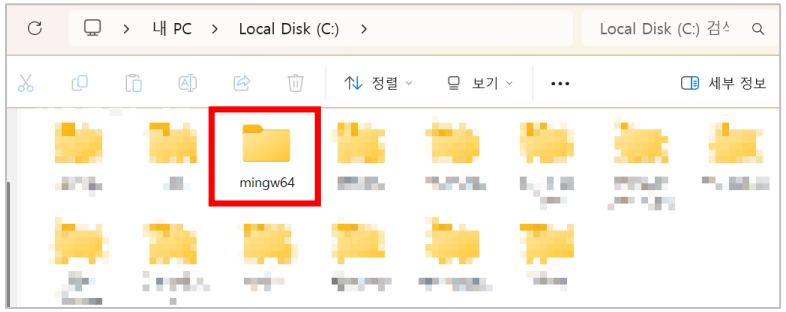
Release versions

UCRT runtime

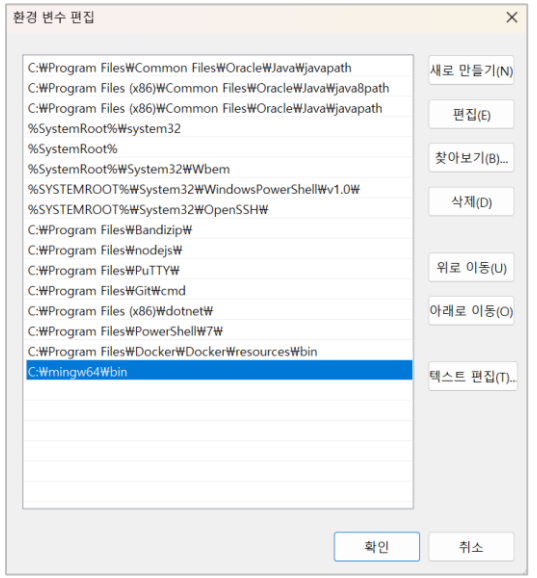
- GCC 15.2.0 (with **POSIX** threads) + MinGW-w64 13.0.0 (UCRT) - release 5 **(LATEST)**
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
  - Win64 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
- GCC 15.1.0 (with **POSIX** threads) + MinGW-w64 13.0.0 (UCRT) - release 4
  - Win32 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
  - Win64 (without LLVM/Clang/LLD/LLDB): 7-Zip archive* | Zip archive
- GCC 15.1.0 (with **POSIX** threads) + MinGW-w64 13.0.0 (UCRT) - release 2

다운로드 후 압축해제

② 압축 해제 이후 C드라이브로 옮기기



③ path 환경 변수에 등록하기(C₩mingw64₩bin)



04

★ Visual Studio Code(VS Code)

☐ Code::Blocks

☐ Dev-C++

☐ CLion

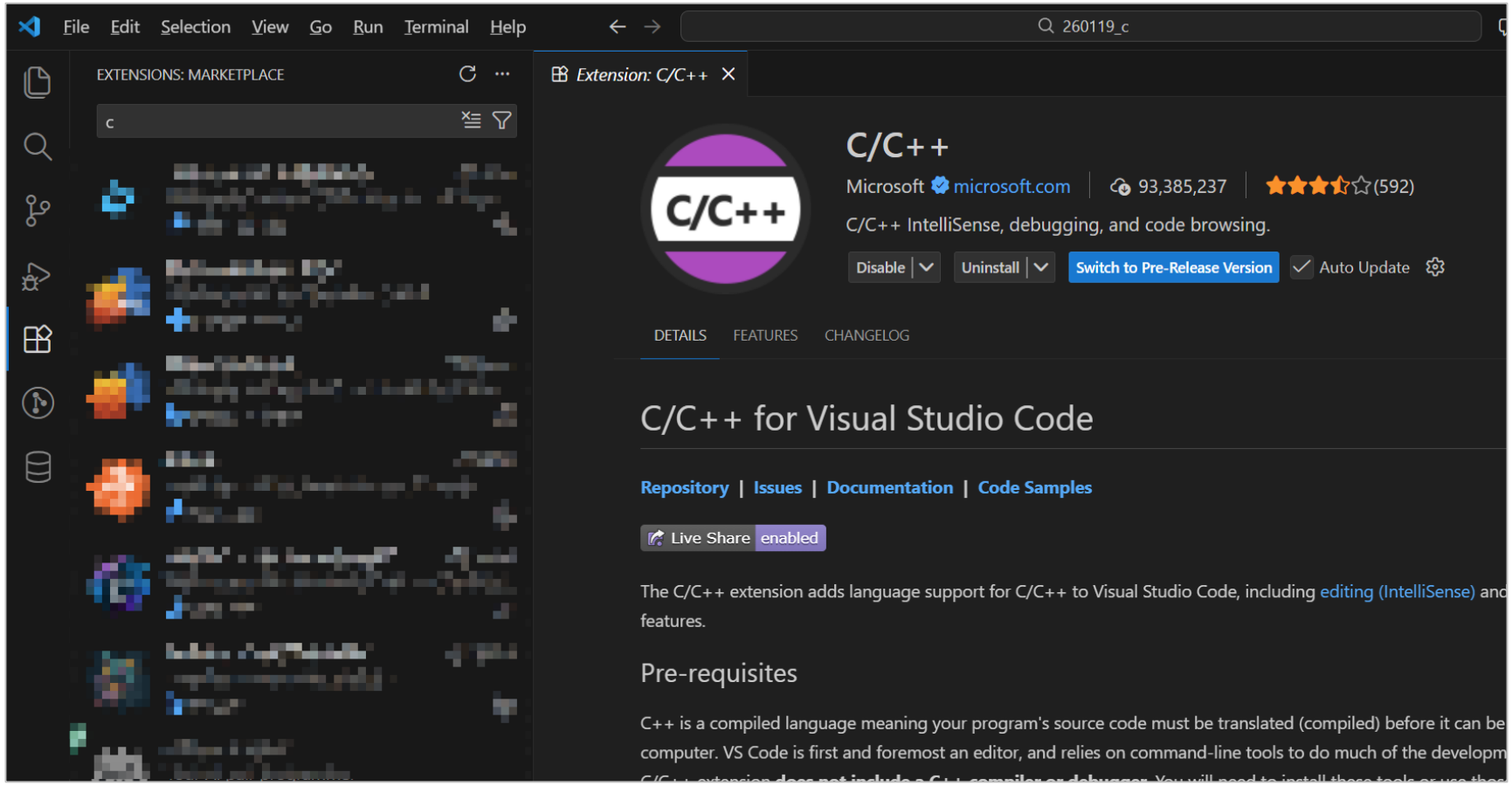
★ Visual Studio

범용적인 개발 도구로 많이 활용되는 VS-CODE

VS Code는 가벼운 코드 에디터이다.

확장 프로그램 설치

05



Extension에서 C/C++ 확장 프로그램을 찾아서 설치

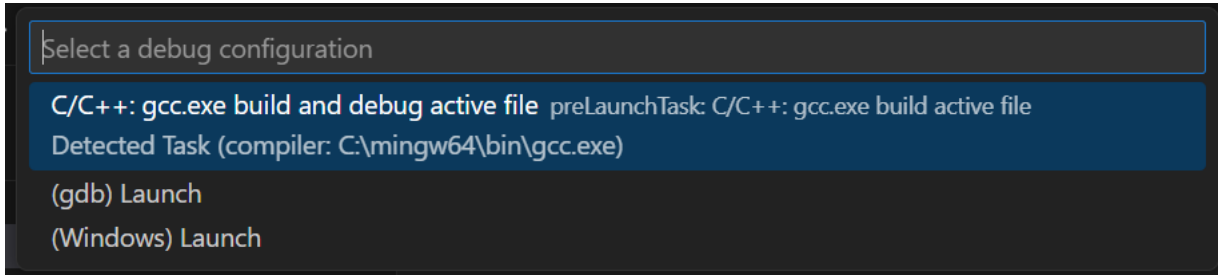
첫 프로그램 작성

06

01_hello_world.c

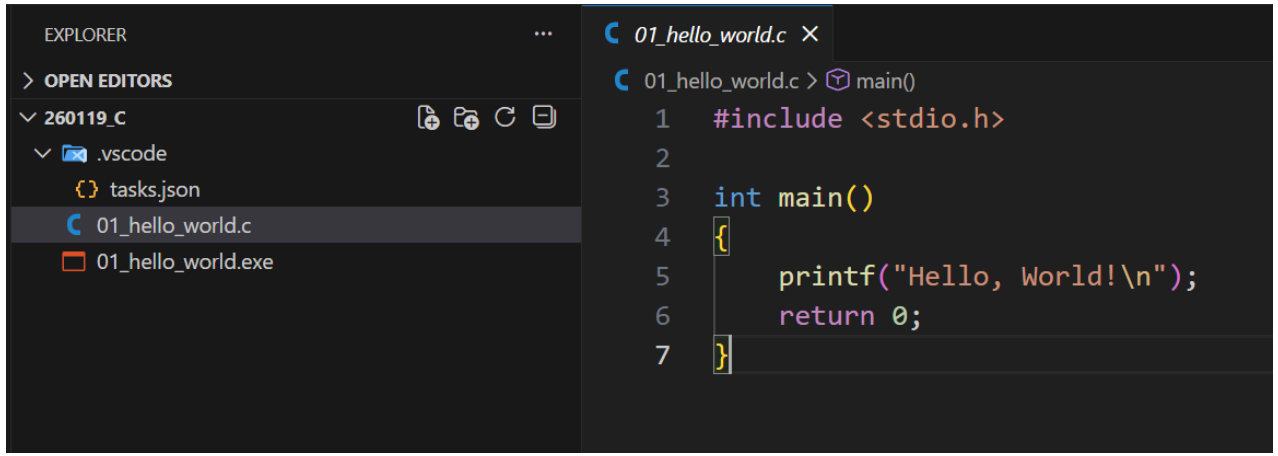
```
1.    #include <stdio.h>
2.
3.    int main() {
4.        printf("Hello World\n");
5.        return 0;
6.    }
```

소스파일 작성



실행 환경 설정하기

07



tasks.json 파일이 생성된다.

이후로는 F5 버튼으로 실행(디버그 모드).

프로그램 실행 결과

08

```
PS C:\Workspace\[redacted]\260119_c>  & '[redacted]\.vscode\extensions\ms-vscode.cpptools-1.29.3-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-mpxudwlc.smh' '--stdout=Microsoft-MIEngine-Out-2pvnnnbx.yrt' '--stderr=Microsoft-MIEngine-Error-al21yae1.bhk' '--pid=Microsoft-MIEngine-Pid-fmskbf1c.m5a' '--dbgExe=C:\mingw64\bin\gdb.exe' '--interpreter=mi'
Hello, World!
PS C:\Workspace\[redacted]\260119_c>
```

프로그램 실행 완료!

주석 – 프로그램 실행에 영향을 미치지 않는다!

09

02_comment.c

```c
#include <stdio.h>

/**
 * 파일이름: 02_comment.c
 * 작성자: 홍길동
 * 작성일: 2026-01-19
 * 설명: 이 파일은 주석에 대한 예제를 포함하고 있습니다.
 */
int main()
{
    printf("10\n");
    printf("3.15\n");
    printf("3 + 5 = %d\n", 3 + 5); // 덧셈 결과 출력
    printf("%f는 실수입니다\n", 3.15);

    return 0;
}
```

주석을 사용한 예제

★ 주석은 프로그램 실행에 영향을 미치지 않음

□ 1) 주석은 /*로 시작하여 */로 끝남

2) 한 줄 주석은 // 로 시작

```
10
3.15
3 + 5 = 8
3.150000는 실수입니다
```

콘솔(Console)창에 나타난 결과값

printf 구조와 서식 지정자


```
int main()
{
    printf("3 + 5 = %d₩n", 3+5);
    printf("%f는 실수입니다.₩n", 3.15);
    return 0;
}
```

★ %d: 정수

★ %f: 실수

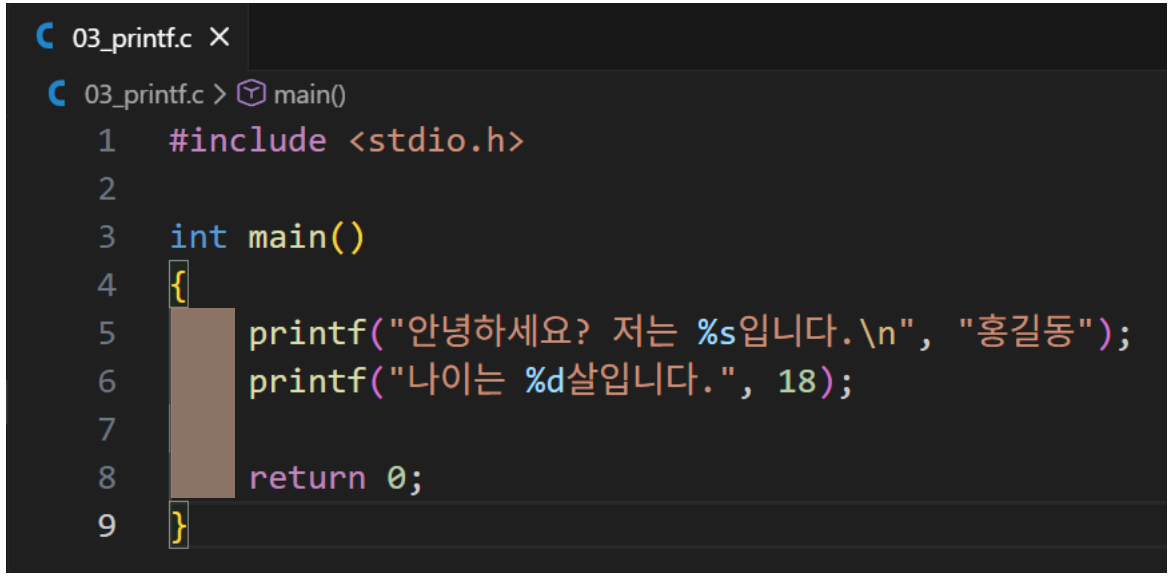
★ %s: 문자열

★ ₩n: 개행 문자

03_printf.c

□ 지문을 따라가며 프로그램을 구현해 보세요.

1. 터미널(콘솔)에 "안녕하세요? 저는 OOO입니다." 출력하기
2. 다음 줄(개행 이후)에 "나이는 OO살 입니다." 출력하기
3. 이 때, OO에는 본인의 이름과 나이를 출력하세요.
4. 본인의 이름 및 나이는 서식 지정자를 활용하세요.



들여쓰기는 굉장히 중요하다.

★ 중괄호로 시작된 영역을 기준으로
Tab을 눌러 4칸 정도 들여쓰기를 함

□ 들여쓰기를 하지 않으면 가독성이 많이 떨어짐

□ 특정 프로그래밍 언어는 들여쓰기가 문법으로 사용

중괄호 방식

03_printf.c

03_printf.c > main()

```c
#include <stdio.h>

int main()
{
    printf("안녕하세요? 저는 %s입니다.\n", "홍길동");
    printf("나이는 %d살입니다.", 18);

    return 0;
}
```

중괄호 방식 – 1

03_printf.c

03_printf.c > main()

```c
#include <stdio.h>

int main() {
    printf("안녕하세요? 저는 %s입니다.\n", "홍길동");
    printf("나이는 %d살입니다.", 18);

    return 0;
}
```

중괄호 방식 – 2

C에서 권고하는 방식은 ①번 방식

◈ 04_use_variable.c

04_use_variable.c

```c
#include <stdio.h>

int main()
{
    int num1;  // 변수 num1 선언
    num1 = 10; // 변수 num1에 값 10 할당

    int num2 = 20;          // 변수 num2 선언과 동시에 값 20 할당
    int num3 = num1 + num2; // 변수 num3에 num1과 num2의 합 할당

    printf("num3의 값: %d\n", num3); // num3의 값 출력
}
```

변수를 선언하는 다양한 방식

★ C에서 = 은 좌변과 우변이 같다는 것을 의미하는게 아니라 우변의 값을 좌변에 대입하는 것을 의미한다.

```
num3의 값: 30
```

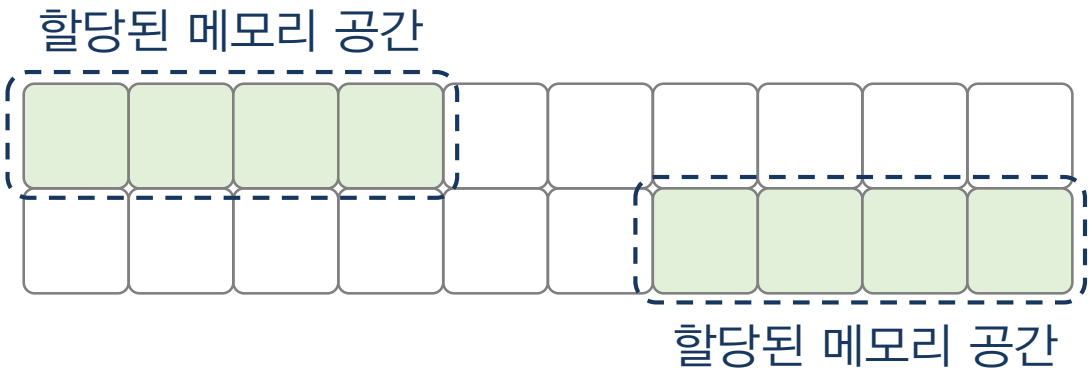
콘솔(Console)창에 나타난 결과값

변수

★ 값을 저장하기 위해 메모리 공간에 붙인 이름

□ 메모리 공간의 할당과 접근을 위해 필요한 도구

□ 변수의 선언은 메모리 공간의 할당으로 이어짐

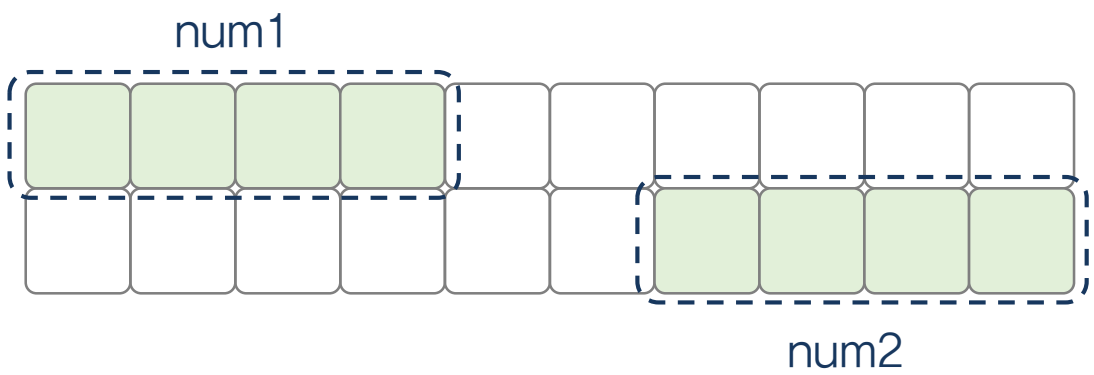


□ 변수의 선언을 통해 두 가지를 결정함

□ 변수의 용도

□ 변수의 이름

```
int num1;  //  변수 num1의 선언
int num2;  //  변수 num2의 선언
```



05_int_variable.c

□ 지문을 따라가며 프로그램을 구현해 보세요.

1. 터미널에 "100 + 200 = 300" 출력하기
2. 이 때, 변수를 자율적으로 사용해 보세요.

변수를 사용하지 않으면?

```
int main()
{
    printf("%d"", 10 + 20);
    return 0;
}
```

□ 10과 20이라는 상수는 재사용이 불가능하고, 사용자에게 값을 입력 받는 행위 등을 할 수 없다.

```
int main()
{
    int n1 = 10;
    int n2 = 20;
    printf("%d"", n1 + n2);
    return 0;
}
```

변수를 사용하면?

□ 변수 n1과 n2는 재사용할 수 있으며
동적 프로그램 작성이 가능하다.

C언어에서 변수 이름 짓기

- □ c는 대소문자를 구분 함 (num과 Num은 다름)
- □ 변수는 숫자로 시작할 수 없음
- □ $과 _이외의 특수문자는 이름에 사용할 수 없음
- ★ 키워드(예약어)는 변수의 이름으로 사용할 수 없음

- ★ 변수 이름은 단어마다 언더바(_)로 써주는 것이 관례
- ★ 변수, 함수의 이름은 시작을 소문자로,
  다음 단어부터는 언더바(_) 로 써주는 것이 관례
  (underscore, 언더스코어)
  EX) int my_age = 20;

키워드란 특정 역할을 하는 단어를 의미(int 등등)

자료형은 변수에 저장할 값의 종류와 메모리 크기, 해석 방법을 정하는 규칙

```
int main()
{
    int age = 17;           // 정수를 저장할 수 있는 변수
    float height = 170.5;   // 실수를 저장할 수 있는 변수
    char grade = 'A';       // 문자를 저장할 수 있는 변수
}
```

왜 필요한가?

- 메모리 낭비를 방지하기 위함(필요한 만큼만 사용)
- 연산 방식 결정(정수 연산 vs 실수 연산)
- 오류 예방(문자를 숫자로 잘못 처리하는 문제 등을 방지)

자료형	데이터	크기	표현 가능 범위
char	문자 / 정수	1 바이트	–128 ~ 127
int	정수	4 바이트	약 –21억 ~ 21억
long		4~8 바이트	약 –21억 ~ 9경 이상(환경 의존)
float	실수	4 바이트	$\pm(3.40 \times 10^{38})$
double		8 바이트	$\pm(1.70 \times 10^{-308})$

□ C에는 기본적으로 Boolean 자료형이 주어지지 않는다.

↳ 정수로 참/거짓을 표현한다(0은 false, 0이 아닌 값은 true).

□ C는 컴파일러와 운영체제에 따라 자료형의 크기(바이트)가 달라질 수 있다.

◈ 06_variable_various.c

06_variable_various.c

06_variable_various.c > main()

```c
#include <stdio.h>

int main()
{
    char ch1 = 'A';
    int num = 123;
    float f_val = 45.67f;
    double d_val = 89.1011;

    printf("Character: %c\n", ch1);
    printf("Integer: %d\n", num);
    printf("Float: %f\n", f_val);
    printf("Double: %lf\n", d_val);
}
```

★ 변수명만 봐도 하는 역할을 알 수 있게 만드는 것이 좋은 습관

예제처럼 의미 없는 변수명은 절대로 사용하지 말기

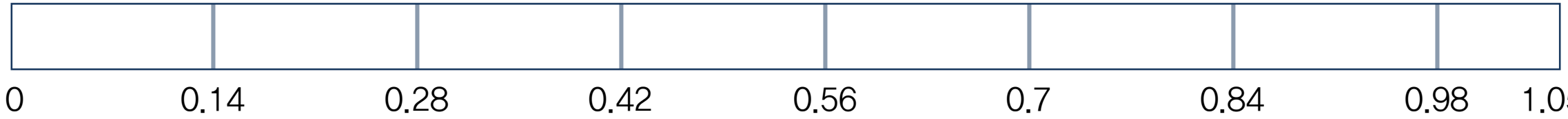
float 자료형은 접미사 f가 사용되는 이유가 무엇인가요?

```
Character: A
Integer: 123
Float: 45.669998
Double: 89.101100
```

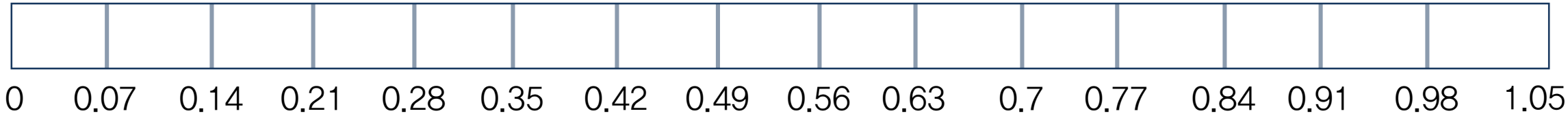
콘솔(Console)창에 나타난 결과값

실수 자료형 선택 기준은 정밀도

float 자료형



double 자료형



float 자료형이라도 작은 수부터 정말 큰 수까지 표현이 가능

따라서 자료형의 크기를 비교하는 것은 큰 의미가 없고, 숫자를 표현하는 정밀도에 의의를 둠

※ 극단적인 예시임

문자 자료형 – ascii code

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	|
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

아스키 코드 표(Table)

- C의 문자 자료형 char
- C는 문자를 1바이트 숫자(아스키 코드)로 저장

★ 작은 따옴표로 묶어서 하나의 문자를 표시

```
char ch1 = 'A' ;  // A의 아스키 코드 값 65 저장
char ch2 = 'a' ;  // a의 아스키 코드 값 97 저장
```

문자 자료형 – 자료형은 해석 방법까지 제공

◈ 07_char_ascii.c

07_char_ascii.c

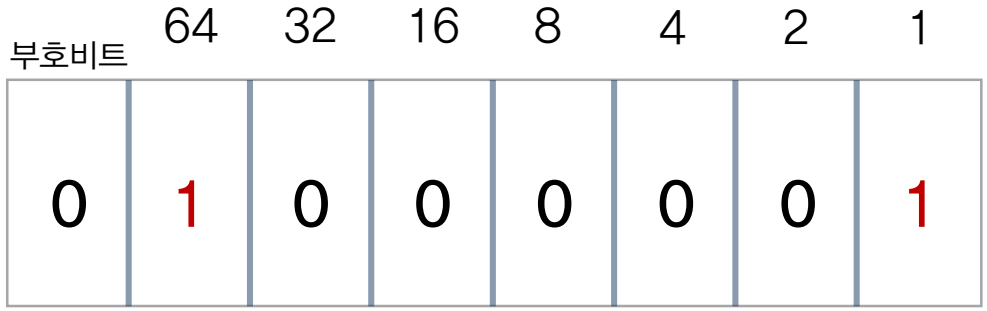
```c
#include <stdio.h>

int main()
{
    char ch1 = 'A';

    printf("Character: %c\n", ch1);
    printf("ASCII value: %d\n", ch1);
}
```

```
Character: A
ASCII value: 65
```

콘솔(Console)창에 나타난 결과값



□ 아스키(ASCII) 코드는 7비트로 128개의 문자를 표현

★ 같은 비트를 해석하는 방식에 따라
A로 나타낼 수도 있고, 65로 나타낼 수도 있음

★ 자료형은 해석 방법까지 내포되고 있음을 알기

콘솔로 값을 입력 받으려면?

◈ 08_scanf.c

08_scanf.c

```c
#include <stdio.h>

int main()
{
    int number;
    printf("정수를 입력하세요: ");
    scanf_s("%d", &number);

    printf("You entered: %d\n", number);
    printf("%p\n", &number); // 변수의 메모리 주소 출력
```

★ 앰퍼샌드(&)가 사용되는 이유는
사용자가 입력한 값을 변수에 저장할 때,
scanf 함수에 변수의 주소를 전달해야 하기 때문이다.

C언어의 꽃 포인터

```
정수를 입력하세요: 100
You entered: 100
00000000005FFE6C
```

콘솔(Console)창에 나타난 결과값

□ 사용자한테 나이를 입력 받아 출력해 보기

1. 09_scanf_exam1.c 파일 생성하기
2. 콘솔에 "나이를 입력하세요: " 출력하기
3. 사용자로부터 값을 입력 받기(정수 값)
4. 콘솔에 "제 나이는 OO살 입니다." 출력하기

```
나이를 입력하세요: 23
제 나이는 23살 입니다.
```

콘솔(Console)창에 나타난 결과값

□ 키와 몸무게를 입력 받는 프로그램을 만들어 보기

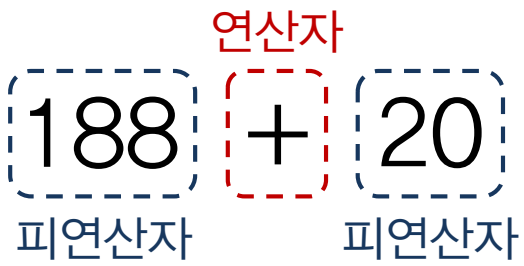
1. 10_scanf_exam2.c 파일 생성하기
2. 콘솔에 "키(cm)와 몸무게(kg)를 입력하세요(예: 170.2 60.3): " 출력하기
3. 사용자로부터 값을 입력 받기(정수 값)
4. 콘솔에 "입력한 키: 170.2cm, 몸무게: 60.3kg" 출력하기

```
키(cm)와 몸무게(kg)를 입력하세요(예: 170.2 60.3): 187.6 78.2
입력한 키는 187.6cm, 몸무게는 78.2kg
```

콘솔(Console)창에 나타난 결과값

+, −, *, /, %, 〈, 〉 등 연산을 할 때 사용하는 기호

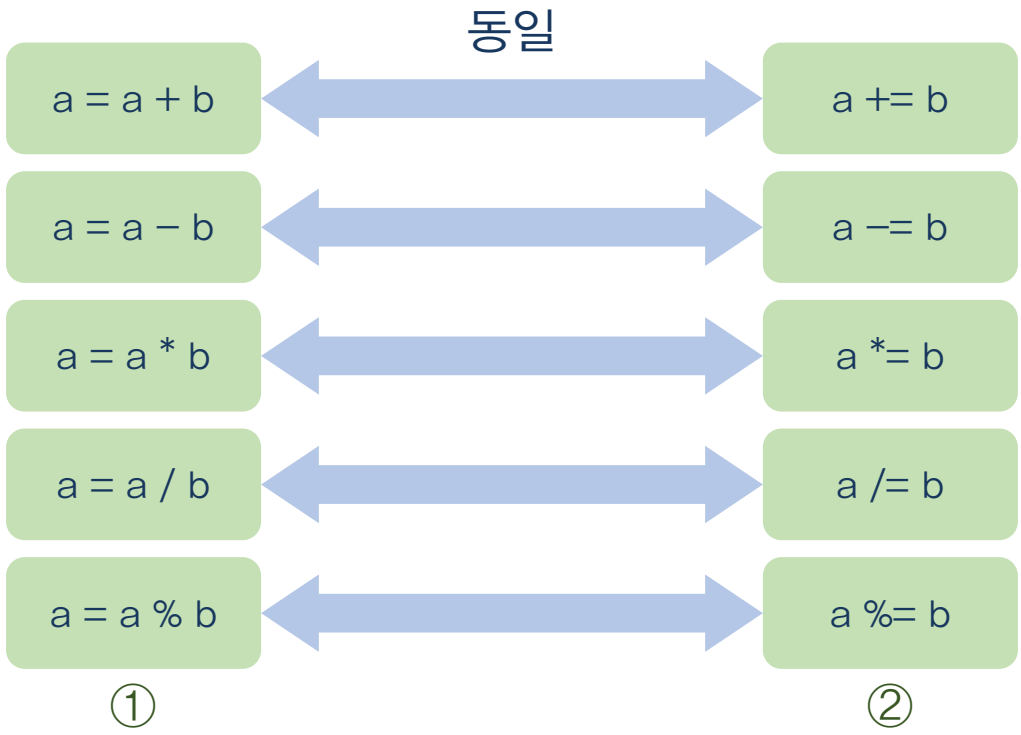
□ 피연산자는 연산자의 기능이 적용되는 대상을 말함(변수, 상수, ...)



연산자	연산자의 기능	결합 방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입 EX) int foo = 20;	←
+	피연산자의 값을 더함 EX) int foo = 10 + 20;	
–	왼쪽의 피연산자 값에서 오른쪽의 피연산자 값을 뺌 EX) int foo = 20 – 10;	
*	피연산자의 값을 곱함 EX) int foo = 10 * 20;	
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눔 EX) int foo = 10 / 5;	
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나누었을 때, 남는 나머지 반환 EX) int foo = 7 % 3;	

대입 연산자와 산술 연산자의 종류

복합 대입 연산자



어느 방식을 사용해도 상관 없으나 개인적으로 ② 방식을 권장 함

연산자	연산자의 기능
<	n10이 n2보다 작은 값인지 확인 EX) n1 < n2;
>	n10이 n2보다 큰 값인지 확인 EX) n1 > n2;
<=	n10이 n2와 같거나 작은 값인지 확인 EX) n1 <= n2;
>=	n10이 n2와 같거나 큰 값인지 확인 EX) n1 >= n2;
==	n10이 n2와 같은 값인지 확인 EX) n1 == n2;
!=	n10이 n2와 다른 값인지 확인 EX) n1 != n2;

비교 연산자의 종류

12_compare.c

연산자 우선 순위

연산자	우선 순위
(), [], ., –〉, foo++, foo––	1
++foo, ––foo, +foo, –foo, ~, !	2
*, /, %	3
+, –	4
〈〈, 〉〉, 〉〉〉	5
〈, 〉, 〈=, 〉=	6
==, !=	7
&	8
^	9
|	10
&&	11
`	12
?:	13
=, +=, –=, *=, /=, %= 등	14

□ 연산자의 우선 순위를 외우려고 하기 보다 필요에 따라 ()를 이용해 우선 순위를 변경하는 것이 효율적일 수 있음

13_compare_exam.c

□ 우선 순위와 괄호 사용하기

1. 아래 수식이 출력되도록 괄호를 사용해 수식을 완성하기

2–1. printf("%d", 10 + 20 * 3); // 출력값: 90

2–2. printf("%d", 90 / 10 + 5); // 출력값: 6

2–3. printf("%d", 2 + 3 * 4 – 5); // 출력값: –5

2–4. printf("%d", 50 – 10 + 5 * 2); // 출력값: 30

2–5. printf("%d", 8 + 12 / 2 * 3); // 출력값: 10

연산자	연산자의 기능
&&	A와 B 모두 1이면 연산 결과는 1 (논리 AND) EX) A && B;
||	A와 B 둘 중 하나라도 1이면 연산 결과는 1 (논리 OR) EX) A || B;
!	A가 1일 때는 0, A가 0일 때는 1 (논리 NOT) EX) !A;

논리 연산자의 종류

연산자	연산자의 기능
++ (prefix)	피연산자에 저장된 값을 1 증가 시킴 EX) int foo = ++bar;
-- (prefix)	피연산자에 저장된 값을 1 감소 시킴 EX) int foo = --bar;
++ (postfix)	피연산자에 저장된 값을 1 증가 시킴 EX) int foo = bar++;
-- (postfix)	피연산자에 저장된 값을 1 감소 시킴 EX) int foo = bar--;

증감연산자는 붙는 위치에 따라 작동 방식에 차이점이 있다.

증감 연산자를 이용한 예제

◈ 15_prefix_postfix.c

15_prefix_postfix.c

```c
#include <stdio.h>

int main()
{
    int num = 7;

    // 전위 증가 연산자 사용: 값이 증가한 후에 대입
    int result1 = ++num; // result1 = 8, num = 8
    printf("전위 증가 연산자 사용:\n");
    printf("result1 = %d, num = %d\n", result1, num);

    // 후위 증가 연산자 사용: 값을 대입한 후에 증가
    int result2 = num++; // result2 = 8, num = 9
    printf("후위 증가 연산자 사용:\n");
    printf("result2 = %d, num = %d\n", result2, num);

    return 0;
}
```

증감 연산자가 prefix, postfix로 쓰일 때 차이점 알기

★ 다른 연산자와 함께 사용하는 연산식에서
증감 연산자는 위치에 따라 결과가 다르게 나오므로 주의

□ prefix는 증감 연산자가 반영되고 연산을 수행

□ postfix는 연산을 수행하고 증감 연산자가 반영

prefix와 postfix로 사용될 때의 값을 예측해 보세요.

□ 콘솔창에 출력하는 방식 익히기

1. 15_example.c 만들고 main() 함수 생성
2. 콘솔에 “안녕하세요? 저는 OO입니다.” 이름 넣어서 출력
3. 개행 후 “저의 취미는 OO입니다.” 출력하기
4. 개행 후 “3.14 + 3.56 = 6.7” 출력하기

   이 때, 결과 값은 직접 입력하거나 더하기 연산자를 활용

□ 정수형 변수 생성 후 활용하기

1. 16_example.c 만들고 main() 함수 생성
2. 정수형 변수로 num1, num2 선언 후 각각 10, 20으로 초기화
3. 정수형 변수로 multiply 선언 후 num1과 num2를 곱한 값으로 초기화
4. 콘솔에 “num1 * num2 = 200” 출력하기

   이 때, 결과 값은 직접 입력하지 말고 곱하기 연산자를 활용

□ 나이를 입력 후 1년 뒤 나이 출력

1. 17_example.c 만들고 main() 함수 생성
2. 나이를 입력 받아 1년 뒤 나이를 계산하기

```
나이를 입력하세요: 18
1년 뒤 당신의 나이는 19살입니다.
```

□ 점수 계산하기

1. 18_example.c 만들고 main() 함수 생성
2. 3개의 과목 점수를 입력 받아 총점과 평균을 구하기
3. 이 때, 평균은 소수점 두 번째 자리로 반올림 하기

```
세 과목의 점수를 입력하세요(예: 85 90 78): 85 90 78
총점은 253점입니다.
평균은 84.33입니다.
```

□ 상품 가격 계산

1. 19_example.c 만들고 main() 함수 생성
2. 아래의 결과를 보고 프로그램을 만들어 보세요.
3. printf() 함수에 %를 출력하는 방법은 %% 를 사용합니다.

```
세 상품의 가격을 입력하세요(예: 10000 20000 15000): 10000 20000 15000
총 가격은 45000원입니다.
10% 할인된 가격은 40500원입니다.
```

```
세 상품의 가격을 입력하세요(예: 10000 20000 15000): 1400 3560 8570
총 가격은 13530원입니다.
10% 할인된 가격은 12177원입니다.
```

THANK YOU

작 성 자 : 이 재 선

2026년