

## C - 변수, 연산자



# Visual Studio 2022



# 변수(Variable)

## ● 변수란?

- 프로그램 내부에서 사용하는 데이터를 저장해두는 메모리 공간
- 한 순간에 한 개의 값을 저장한다. (배열-여러 개 저장)

## ● 변수의 선언 및 사용

- 자료형 변수이름;
- 자료형 변수이름 = 초기값;

변수 선언문

```
char ch;
```

```
int year = 2019;
```

```
double rate = 0.75;
```

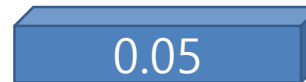
이름

ch

year

rate

공간(값)



자료형

char

int

double



# 변수의 선언과 사용

## ● 변수 사용 예제

```
//정수형 변수 선언 및 초기화
int n1 = 10;
int n2 = 20;

//출력
printf("두 수의 합: %d\n", n1 + n2);
printf("두 수의 차: %d\n", n1 - n2);

//실수형 변수 선언
double rateOfBirth = 0.75;

printf("대한민국의 2024년 출산율은 %.2f명입니다.\n", rateOfBirth);
```

```
두 수의 합: 30
두 수의 차: -10
대한민국의 2024년 출산율은 0.75명입니다.
그 호텔의 서비스는 A등급이다.
그 과일의 이름은 사과이다.
```



# 변수의 선언과 사용

- 변수 사용 예제

```
//문자형 변수 선언
char grade = 'A';

printf("그 호텔의 서비스는 %c등급이다.\n", grade);

//문자열 변수 선언
char nameOfFruit[] = "사과"; //배열 자료구조 사용

printf("그 과일의 이름은 %s이다.\n", nameOfFruit);

//변수 이름 작성시 오류
//int 2n = 5; //숫자로 시작 안됨
//int ag e = 11; //공백 불가
//int class = 3; //예약어는 사용 불가
```



# 자료형(data type)

## ● 자료형이란?

- 데이터를 저장하는 공간의 유형이다.
- 사용할 데이터의 종류에 따라 메모리 공간을 적절하게 설정해 준다.

자료형		용량 (bytes)	주요 용도	범위
정수형	<b>char</b>	1	문자 또는 작은 정수 표현	-128~127
	<b>short</b>	2	정수 표현	-32768~32767
	<b>int</b>	4	큰 범위의 정수 표현	-2147483648~2147483647
	<b>long</b>	8	큰 범위의 정수 표현	$-2^{63} \sim (2^{63}-1)$
실수형	<b>float</b>	4	실수 표현	$10^{-38} \sim 10^{38}$
	<b>double</b>	8	정밀한 실수 표현	$10^{-380} \sim 10^{380}$

- 정수형 양수 표현범위를 2배로 늘릴 때는 자료형 앞에 **unsigned**를 붙일수 있는데. 이 경우 동일한 공간으로 0을 포함한 양수만을 표현하게 된다.
- 예 : char -128~127 -----> unsigned char는 0~255 범위 표현



# 자료형(Data Type)

- 변수의 메모리 주소와 자료형의 크기

```
//정수형 변수 선언 및 초기화
int n1 = 10;
int n2 = 20;

//실수형 변수 선언
double rateOfBirth = 0.75;

//문자형 변수 선언
char grade = 'A';

//문자열 변수 선언
char nameOfFruit[] = "사과";

printf("==== 변수의 값과 메모리 주소 =====\n");
printf("%d\t %x\n", n1, &n1);
printf("%.2lf\t %x\n", rateOfBirth, &rateOfBirth);
printf("%c\t %x\n", grade, &grade);
printf("%s\t %x\n", nameOfFruit, &nameOfFruit[0]);
```



# 자료형(Data Type)

- 변수의 메모리 주소와 자료형의 크기

```
printf("\n==== 자료형의 크기 ==== \n");
printf("int형: %dByte\n", sizeof(n1));
printf("double형: %dByte\n", sizeof(rateOfBirth));
printf("char형: %dByte\n", sizeof(grade));
printf("문자열형: %dByte\n", sizeof(nameOfFruit));

return 0;
```

```
== 변수 값과 메모리 주소 ==
10      a411f524
0.75    a411f568
A        a411f584
사과     a411f5a4

==== 자료형의 크기 ====
int형: 4Byte
double형: 8Byte
char형: 1Byte
문자열형: 5Byte
```



# 자료형(Data Type)

## • 자료형의 범위

```
/*  
    'A' - 아스키 코드값(65), char형 1Byte = 8bit  
    int : -128 ~ 127  
    unsigned int : 0 ~ 255  
    unsigned 형은 음수를 저장할 수 없고 양수 범위가 2배로 늘어남  
*/  
printf("==== char 자료형 ====\\n");  
char ch = 'A';  
printf("%c %d\\n", ch, ch);  
  
char value1 = -128;  
printf("%d\\n", value1);  
  
char value2 = 128; //범위를 초과하여 overflow 발생  
printf("%d\\n", value2);  
  
unsigned char value3 = -128;  
printf("%d\\n", value3);
```





# 자료형(Data Type)

## • 자료형의 범위

```
/*
    int형 4B = 32bit
    -21억 ~ 21억
    더 큰 정수
    long 4B(windows), 8B(macOS)
    long long 8B
*/
printf("==== int 자료형 =====\n");
int iNum = 2100000000;
printf("%d\n", iNum);

int iNum2 = 2200000000; //범위를 초과하여 overflow 발생
printf("%d\n", iNum2);

printf("==== long 자료형 =====\n");
long lNum = 2200000000L; //overflow 발생
printf("%ld\n", lNum);

long long llNum = 2200000000L;
printf("%lld\n", llNum);
```



# 자료형(Data Type)

## • 자료형의 범위

```
/*  
    float - 4B, 소수 6자리 표현  
    double - 8B, 소수 15자리 표현  
    정밀도를 표현  
*/  
printf("=== float와 double 자료형 ===\n");  
float fNum = 0.1234567F; //오류  
printf("%.6f\n", fNum);  
  
double dNum = 0.1234567890123456; //오류  
printf("%.15lf\n", dNum);
```

```
===== char 자료형 =====  
A 65  
-128  
-128  
128  
===== int 자료형 =====  
2100000000  
-2094967296  
===== long 자료형 =====  
-2094967296  
2200000000  
=== float와 double 자료형 ===  
0.123457  
0.123456789012346
```



# 문자 자료형

## ■ 문자 자료형(char, 배열)

- 문자 1개를 표현할때 홑따옴표(' ')로 감싸준다.
- 아스키 코드(ASCII코드) – 각 문자에 따른 특정한 숫자 값(코드 값)을 부여  
영문자, 숫자 만 표현 가능
- 유니 코드 – 한글, 중국어등 아스키 코드로 표현할 수 없는 문자를 2바이트 이상의 크기로 표현할수 있는 표준 코드

<http://www.unicode.org/charts/PDF/UAC00.pdf>



# 아스키 코드

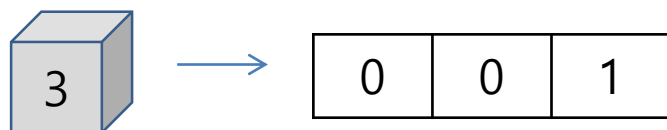
- 아스키 코드(ASCII Code)

아스키 코드는 미국 [ANSI](#)에서 표준화한 정보교환용 7비트 부호체계이다.

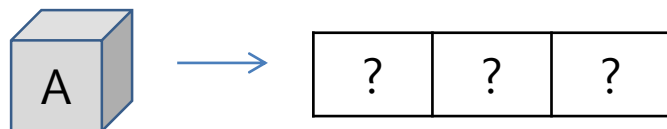
000(0x00)부터 127(0x7F)까지 총 128개의 부호가 사용된다.

영문 키보드로 입력할 수 있는 모든 기호들이 할당되어 있는 부호 체계이다.

10진수를 2진수로 변환



문자를 2진수로 변환



저장하는 문자에 해당하는 숫자를 지정하고 메모리에 저장할때는 그 숫자를 비트 단위로 바꾸어 저장

# 아스키 코드

## ● 아스키 코드(ASCII Code)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>

# 아스키 코드 vs 유니코드

- 유니 코드(Uni code)

전 세계의 모든 문자를 다루도록 설계된 표준 문자 전산 처리 방식이다.  
유니코드를 사용하면 **한글과 간체자, 아랍 문자** 등을 통일된 환경에서 깨뜨리지 않고 사용할 수 있다.

초창기에는 문자 코드는 ASCII의 로마자 위주 코드였고, 1바이트의 남은 공간에 각 나라가 자국 문자를 할당하였다.

이런 상황에서 다른 국가에 이메일을 보냈더니 글자가 깨졌던 것. 이에 따라 **2~3바이트의** 넉넉한 공간에 세상의 모든 문자를 할당한 결과물이다.

**현재는 유니코드가 아스키 코드를 포함하며, 표준이 되었다.**



# 아스키 코드 vs 유니코드

- 유니 코드(Uni code)

	Hangul Syllables																ACFF
	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF	
0	가 AC00	감 AC10	갸 AC20	갹 AC30	갈 AC40	각 AC50	갸 AC60	거 AC70	검 AC80	겐 AC90	갸 ACA0	결 ACB0	격 ACC0	겉 ACD0	고 ACE0	곰 ACF0	
1	각 AC01	갑 AC11	갸 AC21	갹 AC31	갸 AC41	갸 AC51	갸 AC61	걱 AC71	겁 AC81	갸 AC91	갸 ACA1	겉 ACB1	겉 ACC1	겉 ACD1	곡 ACE1	곰 ACF1	
2	갸 AC02	갸 AC12	갸 AC22	갸 AC32	갸 AC42	갸 AC52	갸 AC62	거 AC72	겉 AC82	갸 AC92	갸 ACA2	겉 ACB2	겉 ACC2	겉 ACD2	곡 ACE2	곰 ACF2	
3	갸 AC03	갸 AC13	갸 AC23	갸 AC33	갸 AC43	갸 AC53	갸 AC63	거 AC73	겉 AC83	갸 AC93	갸 ACA3	겉 ACB3	겉 ACC3	겉 ACD3	곡 ACE3	곰 ACF3	
4	간 AC04	갸 AC14	갸 AC24	갸 AC34	갸 AC44	갸 AC54	갸 AC64	건 AC74	갸 AC84	갸 AC94	갸 ACA4	겉 ACB4	겉 ACC4	겉 ACD4	곤 ACE4	곰 ACF4	
5	갸 AC05	갸 AC15	갸 AC25	갸 AC35	갸 AC45	갸 AC55	갸 AC65	겉 AC75	겉 AC85	갸 AC95	갸 ACA5	겉 ACB5	겉 ACC5	겉 ACD5	곡 ACE5	곰 ACF5	
6	갸 AC06	갸 AC16	갸 AC26	갸 AC36	갸 AC46	갸 AC56	갸 AC66	겉 AC76	겉 AC86	갸 AC96	갸 ACA6	겉 ACB6	겉 ACC6	겉 ACD6	곡 ACE6	곰 ACF6	
7	간 AC07	갸 AC17	갸 AC27	갸 AC37	갸 AC47	갸 AC57	갸 AC67	건 AC77	갸 AC87	갸 AC97	갸 ACA7	겉 ACB7	겉 ACC7	겉 ACD7	곤 ACE7	곰 ACF7	
8	갈 AC08	각 AC18	갸 AC28	갸 AC38	갸 AC48	갸 AC58	갸 AC68	겉 AC78	겉 AC88	갸 AC98	갸 ACA8	겉 ACB8	겉 ACC8	겉 ACD8	골 ACE8	곰 ACF8	

# 문자 자료형

## ■ 문자 자료형(char, 배열)

```
//숫자 표기 - 아스키 코드값
char ch = '0';

printf("%c %d\n", ch, ch);
printf("%c %d\n", ch+1, ch+1);
printf("%c %d\n", ch+2, ch+2);

//한글은 배열로 저장
char han[] = "가";
char uniCode[] = "\uAC00";

printf("%s\n", han);
printf("%s\n", uniCode);
```





# 컴퓨터에서 데이터 표현하기

- 비트(binary digit)

컴퓨터가 표현하는 데이터의 최소 단위로 2진수 하나의 값을 저장할 수 있는 메모리의 크기

컴퓨터는 0과 1로만 데이터를 저장함(0-> 신호꺼짐, 1-> 신호켜짐)

- 비트로 표현할 수 있는 수의 범위

비트수	표현할 수 있는 범위(십진수)	
1bit	0, 1(0~1)	$2^1$
2bit	00, 01, 10, 11(0~3)	$2^2$
3bit	000, 001, 010, 011, 100, 101, 110, 111(0~7)	$2^3$



# 10진수를 2진수로 바꾸기

## ■ 진수 표현

10진수	2진수	16진수	10진수	2진수	16진수
1	00000001	1	9	00001001	9
2	00000010	2	10	00001010	A
3	00000011	3	11	00001011	B
4	00000100	4	12	00001100	C
5	00000101	5	13	00001101	D
6	00000110	6	14	00001110	E
7	00000111	7	15	00001111	F
8	00010000	8	16	000010000	10

자리 올림 발생



# 부호 있는 수를 표현하는 방법

- 음의 정수는 어떻게 표현할까?

- 정수의 가장 왼쪽에 존재하는 비트는 부호비트입니다.

(양의 정수는 0, 음의 정수는 1을 붙인다.)

- 음수를 만드는 방법은 2의 보수를 취한다.(1의 보수는 0과 1을 반대로 바꿈)

두 수를 더  
하면 0이 됨  
100000000  
맨 앞 1은 제  
거됨

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

10진수 5

1의 보수를 취한다.

1	1	1	1	1	0	1	0
---	---	---	---	---	---	---	---

1을 더한다.

+ 1

1	1	1	1	1	0	1	1
---	---	---	---	---	---	---	---

10진수 -5

-1은 11111111 (0은 00000000)

-2는 11111110(1을 뺀다)

-3은 11111101(1을 뺀다)

-4는 11111100(1을 뺀다)

-5는 11111011(1을 뺀다)



# 형 변환(Type Conversion)

## ● 묵시적 형 변환(자동)

1) 작은 자료형에서 큰 자료형으로 변환

덜 정밀한 수에서 더 정밀한 수로 대입되는 경우

예) `int iNum = 20;`

`float fNum = iNum;`

2) 연산 중에 자동 변환되는 경우

예) `int num1=100; // 정수`

`double num2=3.14; // 실수`

`printf("%lf %n", num1+num2); // 정수 + 실수`

## ● 명시적 형 변환(강제)

1) 큰 자료형에서 작은 자료형으로 변환

변환 자료형을 명시해야 하고(괄호사용), 자료의 손실이 발생

예) `double dNum = 12.34;`

`int iNum = (int)dNum;`



# 형 변환(Type Conversion)

```
//묵시적 형변환(자동 형변환)
int iNum = 20;
float fNum = iNum; //큰자료형 = 작은 자료형

printf("%d\n", iNum);
printf("%f\n", fNum); //20.0
printf("%f\n", iNum + fNum); //40.0

//명시적 형변환(강제 형변환) - cast(캐스트)라 함.
double dNum = 2.54;
int iNum2 = (int)dNum; //작은 자료형 = 큰 자료형

printf("%d\n", iNum2); //2

//연산
dNum = 1.2;
fNum = 0.9;

iNum = (int)dNum + (int)fNum;
printf("%d\n", iNum); //1

iNum = (int)(dNum + fNum);
printf("%d\n", iNum); //2
```



# 항과 연산자

## ■ 항(operand)

- 연산에 사용되는 값

## ■ 연산자(operator)

- 연산에 사용되는 기호

예)  $3 + 7$  (3과 7은 항, '+'는 연산자)



## ■ 항의 개수에 따른 연산자 구분

연산자	설명	연산 예
단항 연산자	항이 한 개인 연산자	$++num$
이항 연산자	항이 두 개인 연산자	$num1 + num2$
삼항 연산자	항이 세 개인 연산자	$(5 > 3) ? 1 : 0$




# 대입 및 부호 연산자

## ■ 대입 연산자

- 변수에 값을 대입하는 연산자
- 연산의 결과를 변수에 대입
- 우선 순위가 가장 낮은 연산자
- 왼쪽 변수(lvalue)에 오른쪽 값(rvalue)를 대입

total = num1 + num2





# 대입 연산자 연습문제

## 변수 값 교환하기

변수 blue에 1이 저장되어 있고, red에 2가 저장되어 있을때 새로운 변수 yellow를 사용하여 값을 교환해 보세요

```
=== 교환전 ===  
blue = 1, red = 2  
=== 교환후 ===  
blue = 2, red = 1
```

```
int blue = 1;  
int red = 2;  
int yellow;  
  
printf("=== 교환전 ===");  
printf("blue = %d, red = %d\n", blue, red);  
  
//변수 바꾸기  
yellow = blue;  
blue = red;  
red = yellow;  
  
printf("=== 교환전 ===");  
printf("blue = %d, red = %d\n", blue, red);
```



# 산술 및 증감 연산자

## ■ 산술 연산자

연산자	기 능	연산 예
+	두 항을 더합니다.	5+3
-	앞 항에서 뒤 항을 뺍니다.	5-3
*	두 항을 곱합니다.	5*3
/	앞 항에서 뒤 항을 나누어 몫을 구합니다.	5/3
%	앞 항에서 뒤 항을 나누어 나머지를 구합니다.	5%3

연산자	기 능	연산 예
++	항의 값에 1을 더합니다.	val = ++num; // num = num+1; val = num++;
--	항의 값에서 1을 뺍니다.	val = --num // num = num-1; val = num--;



# 산술 및 증감 연산자

## ■ 산술 연산자

```
int a = 99;
int b = 2;
printf(" a + b의 결과 : %d\n", a + b);
printf(" a - b의 결과 : %d\n", a - b);
printf(" a * b의 결과 : %d\n", a * b);
printf(" a / b의 결과 : %d\n", a / b);
printf(" a %% b의 결과 : %d\n", a % b);

printf("a++의 값은 : %d\n", a++); //99
printf("a의 값은 : %d\n", a);     //100

printf("++a의 값은 : %d\n", ++a); //101
printf("a의 값은 : %d\n", a);     //101
```



# 비교 및 논리 연산자

## ■ 관계(비교) 연산자

연산의 결과가 참(1), 거짓(0)으로 반환됨

연산자	기 능	연산 예
>	왼쪽 항이 크면 참을, 아니면 거짓을 반환합니다.	num > 3;
<	왼쪽 항이 작으면 참, 아니면 거짓을 반환합니다.	num < 3;
>=	왼쪽 항이 크거나 같으면 참, 아니면 거짓을 반환합니다.	num >= 3;
<=	왼쪽 항이 작거나 같으면 참, 아니면 거짓을 반환합니다.	num <= 3;
==	두 개의 항 값이 같으면 참, 아니면 거짓을 반환합니다.	num == 3;
!=	두 개의 항 값이 다르면 참, 아니면 거짓을 반환합니다.	num != 3



# 비교 및 논리 연산자

## ■ 논리 연산자 / 조건 연산자

연산자	기 능	연산 예
&& (논리 곱)	두 항이 모두 참인 경우에만 결과 값이 참 입니다.	(7<3) && (5>2)
 (논리 합)	두 항중 하나의 항만 참이면 결과 값이 참 입니다.	(7>3)    (5<2)
! (부정)	참은 거짓으로, 거짓은 참으로 바꿉니다.	!(7>3)

연산자	기 능	연산 예
조건식?결과1:결과2;	조건식이 참이면 결과1, 조건식이 거짓이면 결과2가 선택됩니다.	int num = (5>3)?10:20;



# 비교 및 논리 연산자

```
//1 - 참, 0 - 거짓
printf("10 > 5 의 값은 %d입니다.\n", 10 > 5); //1
printf("10 < 5 의 값은 %d입니다.\n", 10 < 5); //0
printf("10 == 10의 값은 %d입니다.\n", 10 == 10); //1
printf("10 != 10 의 값은 %d입니다.\n", 10 != 10); //0
```

//논리 연산

```
int a = 5, b = 3, c = 2;
```

```
printf("0 && 0 의 값은 %d입니다.\n", (a < b) && (b < c)); //0
printf("0 && 1 의 값은 %d입니다.\n", (a < b) && (b > c)); //0
printf("1 && 1 의 값은 %d입니다.\n", (a > b) && (b > c)); //1
```

```
printf("0 || 0 의 값은 %d입니다.\n", (a < b) && (b < c)); //0
printf("0 || 1 의 값은 %d입니다.\n", (a < b) && (b > c)); //1
printf("1 || 1 의 값은 %d입니다.\n", (a > b) && (b > c)); //1
```

```
printf("!0 의 값은 %d입니다.\n", !(a < b)); //1
printf("!1 의 값은 %d입니다.\n", !(b > c)); //0
```



# 조건 연산자

## ■ 조건 연산자

```
int value;
value = (3 > 4) ? 10 : 20;
printf("결과값: %d\n", value);

int fatherAge = 44;
int motherAge = 46;
char result;

result = (fatherAge > motherAge) ? 'T' : 'F';
printf("결과값: %c\n", result);

int x = -5;
int result2;

result2 = (x < 0) ? -x : x; //절대값
printf("결과값: %d\n", result2);
```



# 복합대입 및 조건 연산자

## ■ 복합대입 연산자

연산자	기 능	연산 예
+=	두 항의 값을 더해서 왼쪽 항에 대입합니다.	num += 2; num=num+2
-=	왼쪽 항에서 오른쪽 항을 빼서 그 값을 왼쪽 항에 대입합니다.	num -= 2; num=num-2
*=	두 항의 값을 곱해서 왼쪽 항에 대입합니다.	num *= 2; num=num*2
/=	왼쪽 항을 오른쪽 항으로 나누어 그 몫을 왼쪽 항에 대입합니다.	num /= 2; num=num
%=	왼쪽 항을 오른쪽 항으로 나누어 그 나머지를 왼쪽 항에 대입합니다.	num %= 2; num=num%2





# 복합대입 및 조건 연산자

- 복합대입 연산자

```
int val = 10;

val += 3; //val = val + 1
printf("%d\n", val);

val -= 3; //val = val - 1
printf("%d\n", val);

val *= 3; //val = val * 1
printf("%d\n", val);

val /= 3; //val = val / 1
printf("%d\n", val);
```



# 비트 연산자

## ■ 비트 연산자

연산자	기 능	연산 예
&	$a \& b$	1 & 1 -> 1을 반환, 그 외는 0
	$a   b$	0   0 -> 0을 반환, 그 외는 1
~	$\sim a$	a가 1이면 0, 0이면 1을 반환
<<	$a << 2$	a를 2비트 만큼 왼쪽으로 이동
>>	$a >> 3$	a를 2비트 만큼 오른쪽으로 이동



# 비트 연산자

## ■ 비트 논리연산자

```
int num1 = 5;  
int num2 = 10;  
int result = num1 & num2;
```



```
num1 : 0 0 0 0 0 1 0 1  
& num2 : 0 0 0 0 1 0 1 0  
-----  
result : 0 0 0 0 0 0 0 0
```

## ■ 비트 이동 연산자

```
int num = 5;  
num << 2;
```



```
num      : 0 0 0 0 0 1 0 1  
num << 2 : 0 0 0 1 0 1 0 0
```



# 비트 연산자

## ■ 비트 연산자

```
//비트 논리 연산자
int num1 = 5;    //00000101
int num2 = 10;   //00001010
int result = num1 & num2; //00000000
printf("result = %d\n", result);

result = num1 | num2;    //00001111
printf("result = %d\n", result);

//비트 이동 연산자
int num3 = 2;    //00000010 -> 10진수 2
printf("result = %d\n", num3 << 1); //00000100 -> 10진수 4
printf("result = %d\n", num3 << 2); //00001000 -> 10진수 8
printf("result = %d\n", num3);
printf("result = %d\n", num3 >> 1); //00000001 -> 10진수 1
```



# 연산자 우선 순위

## ■ 연산자 우선 순위

우선순위	형태	연산자
1	일차식	( ) [ ]
2	단항	++ -- !
3	산술	% * / + -
4	비트이동	<< >>
5	관계	< > == !=
6	비트 논리	&   ~
7	논리	&&    !
8	조건	? :
9	대입	= += -= *=



# 상수(constant)

- 상수(constant)

- 한번 설정해 두면 그 프로그램이 종료 될 때까지 변경될 수 없는 값
- 상수 만드는 방법 – 1) 매크로 상수, 2) **const** 키워드 사용

**#define** 상수이름 상수값

**const** 자료형 상수이름 = 상수값;

**#define** PI 3.1415

**const** double PI = 3.1415;

*//PI = 3.14 (변경할 수 없다.)*



# 상수(constant)

- 상수(constant)

```
#include <stdio.h>
#define PI 3.1415 //매크로 상수

int main()
{
    /*
        상수
        1. const 키워드 사용
        2. 매크로 상수
    */
    const int MIN_NUM = 1;
    const int MAX_NUM = 100;

    //MIN_NUM = 200; //변경(재할당) 불가

    printf("%d\n", MIN_NUM);
    printf("%d\n", MAX_NUM);
}
```



# 상수(constant)

- 상수(constant)

```
//원의 넓이 계산하기
int radius = 10;
double area;

area = PI * radius * radius;

printf("원의 넓이: %.21f\n", area);

return 0;
}
```

```
1
100
원의 넓이 : 314.15
```





# 키보드로 데이터 입력 받기

## ■ 데이터 입력 처리

### ✓ 숫자 자료형

**scanf\_s(입력 형식, 데이터 저장 변수)**

ex) int n;

scanf\_s("%d", &n);

### ✓ 문자 자료형

**scanf\_s(입력 형식, 데이터 저장 변수, 데이터의 크기)**

ex) char str[20]

scanf\_s("%s", str, sizeof(str));



# 키보드로 데이터 입력 받기

## ■ 데이터 입력 처리

```
int iNum;
float fNum;
char str[40];

//입력시 변수에 주소 연산자(&) 붙임
printf("정수 입력: ");
scanf_s("%d", &iNum);

printf("입력된 정수: %d\n", iNum);
printf("입력된 정수의 주소: %x\n", &iNum);

printf("실수 입력: ");
scanf_s("%f", &fNum);

printf("입력된 실수: %.f\n", fNum);
printf("입력된 실수의 주소: %x\n", &fNum);
```

```
정수 입력: 9
입력된 정수: 9
입력된 정수의 주소: e88ff824
실수 입력: 2.54
입력된 실수: 2.540000
입력된 실수의 주소: e88ff844
문자열 입력: 대한민국
입력된 문자열: 대한민국
입력된 문자열의 주소: e88ff868
```



# 구속[球速]의 단위 변환 프로그램



메이저리그는 점점 더 빠른 구속을 추구하고 있다. 올 시즌 메이저리그 포심 패스트볼 평균 구속은 시속 93.2마일(150.0km)에 달한다. 이제는 100마일(160.9km)이 넘는 공도 어렵지 않게 볼 수 있게 됐다.

투수에게 있어 구속이 가장 중요한 요소는 아니다. 구종, 제구, 구위 등 수 많은 요소들이 어우러져야 비로소 뛰어난 투구를 할 수 있다. 하지만 같은 조건이라면 당연히 구속이 빠를수록 유리하다. 구속이 빠를수록 타자들이 공에 대처할 수 있는 물리적인 시간이 줄어들기 때문이다.



# 구속(球速)의 단위 변환 프로그램

```
#include <stdio.h>
#define RATE_KPH_MPH 1.6093

/*
 * KPH(킬로미터)를 MPH(마일)로 변환
 */

int main()
{
    int kph;
    double mph;

    printf("당신의 구속을 입력하세요[KPH]: ");
    scanf_s("%d", &kph);

    mph = kph / RATE_KPH_MPH; //마일로 환산

    printf("당신의 구속은 %.21f[MPH]입니다.\n", mph);

    return 0;
}
```



# 실습 문제 1 – 자료형 변환

-----

변수 두 개를 선언해서 10과 2.0을 대입하고, 두 변수의 사칙연산의 결과를 정수로 출력해 보세요. (파일 이름 : Calculator.c)

-----

👉 실행 결과

```
합 : 12  
차 : 8  
곱 : 20  
나누기 : 5
```



## 실습 문제 2 – 산술 연산

-----

빵 10개를 3명이 나눠 가질 경우 각자의 몫과 남은 빵의 개수를 구하시오.  
(파일 이름 : Bread.c)

-----

👉 실행 결과

```
각자의 몫 : 3  
남은빵의 개수 : 1
```

