C언어로 배우는 프로그래밍 기초

Perfect C 3판

INFINITY BOOKS

제 3 장 자료형과 변수

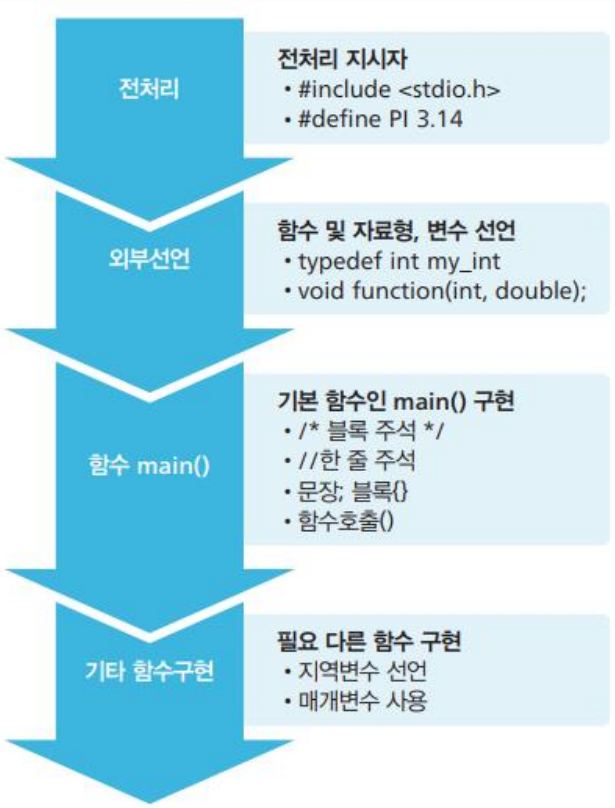
01 프로그래밍 기초
02 자료형과 변수선언
03 기본 자료형
04 상수 표현 방법

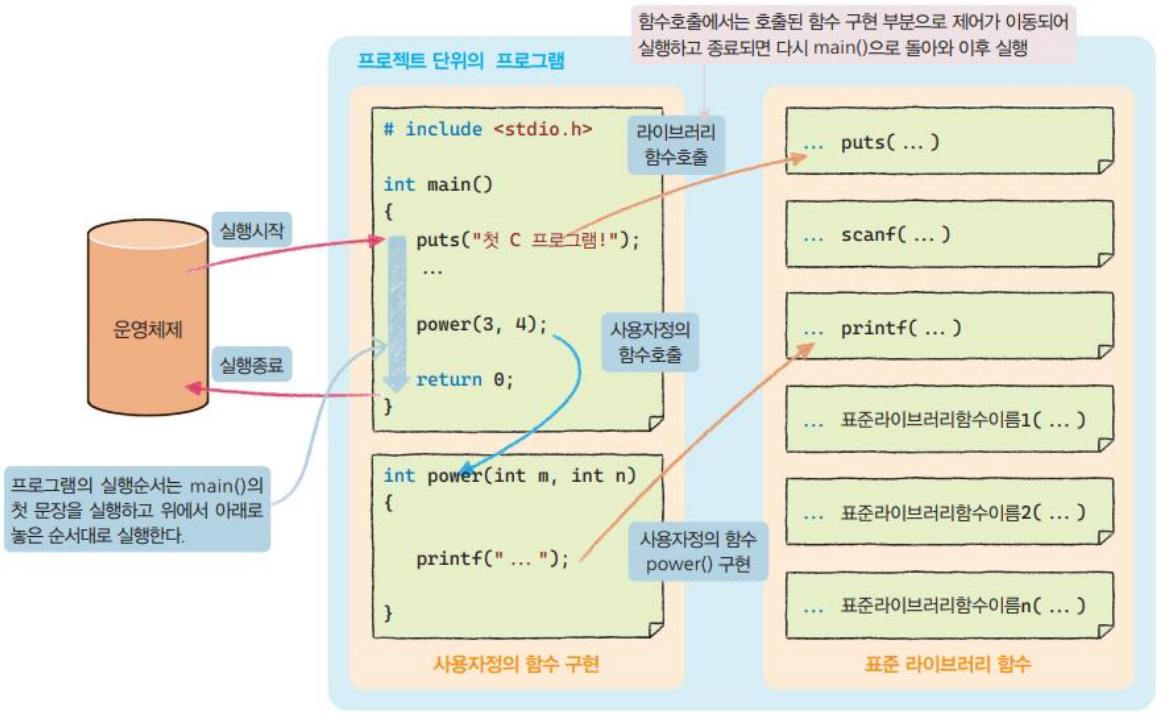
학습목표

- 프로그래밍에 필요한 기본 내용을 설명할 수 있다.
  - C 프로그램의 구조와 실행 순서 과정
  - 키워드와 식별자, 문장과 블록, 그리고 들여쓰기와 주석
- 자료형을 이해하고 변수 선언을 할 수 있다.
  - 자료형과 변수
  - 변수선언과 초기화
- 기본 자료형을 활용할 수 있다.
  - 정수형, 문자형, 부동소수형
- 상수의 개념을 이해하고 상수를 활용할 수 있다.
  - 정수, 문자, 문자열, 실수 등의 리터럴 상수
  - const, 열거형과 매크로를 비롯한 심볼릭 상수

C 프로그램 구조와 프로그램 실행

- **한 프로젝트는 단 하나의 함수 main()과 다른 여러 함수로 구현**
  - 최종적으로 프로젝트 이름과 같은 하나의 실행 파일이 생성
  - C 프로그램은 적어도 main() 함수 하나는 구현되어야 응용 프로그램으로 실행





예약어

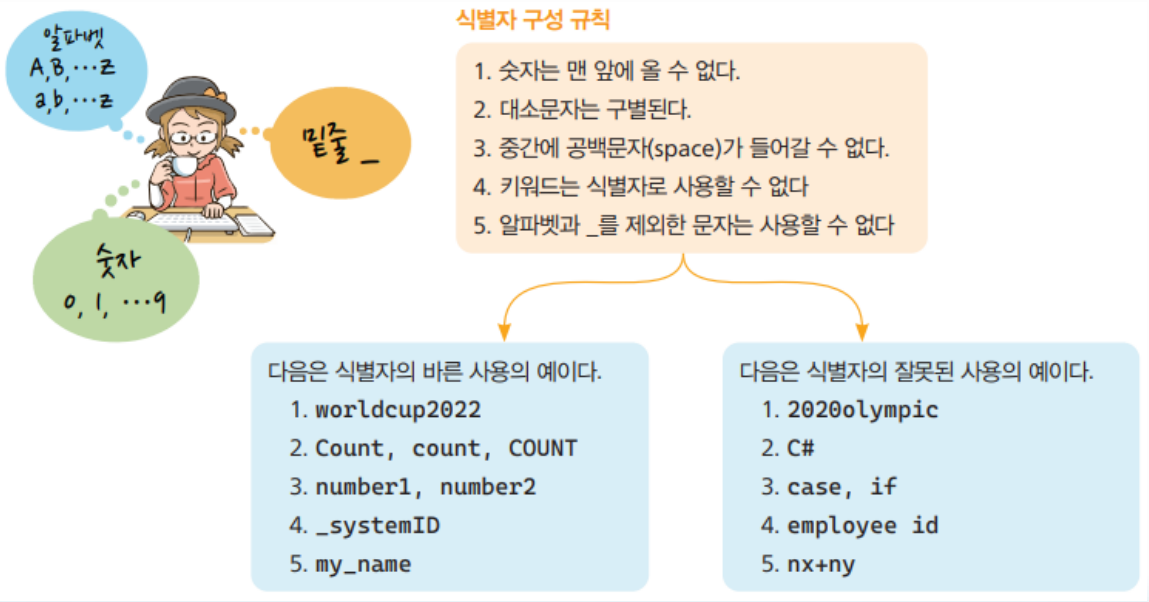
- **문법적으로 고유한 의미를 갖는 예약된 단어**
  - 이 단어들을 다른 용도로 사용해서는 안 된다는 뜻
  - 키워드(keyword)라고도 부름

이러한 단어가 C에서 사용되는 기본 키워드로 문법적인 고유한 의미가 있다.

```
auto      do        goto      signed    unsigned
break     double    if        sizeof    void
case      else      int       static    volatile
char      enum      long      struct    while
const     float     return    typedef
default   for       short     union
```

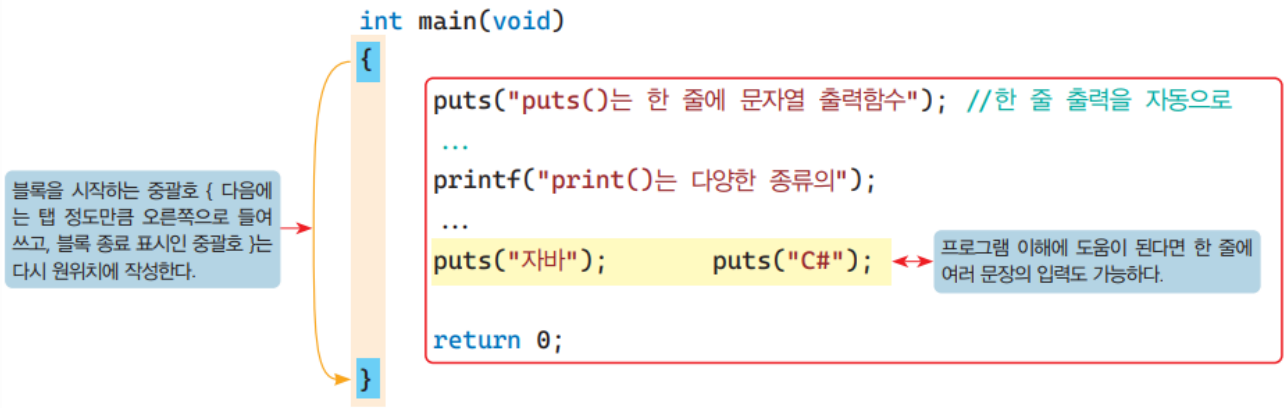
식별자

- **프로그래머가 자기 스스로 정의해 사용하는 단어**
  - 키워드와 비교하여 철자라든지, 대문자, 소문자 등 무엇이라도 달라야 함
    - **대소문자를 모두 구별**
    - **예를 들어, 변수 Count, count, COUNT는 모두 다른 변수**
  - 식별자는 영문자(대소문자 알파벳), 숫자(0 ~ 9), 밑줄(_)로 구성
  - 식별자의 첫 문자로 숫자가 나올 수 없음
  - 프로그램 내부의 일정한 영역에서는 서로 구별되어야 함
  - 키워드는 식별자로 이용 불가능
  - 식별자의 중간에 공백(space)문자 삽입 불가



문장과 블록

- **문장**
  - 프로그래밍 언어에서 컴퓨터에게 명령을 내리는 최소 단위를 문장(statement)
  - 문장은 마지막에 세미콜론 ;으로 종료
- **들여쓰기와 블록**
  - 여러 개의 문장을 묶으면 블록(block)
    - **중괄호(curly brace)로 열고 닫음**
  - 들여쓰기(indentation)
    - **블록 내부에서 문장들을 탭(tab) 키로 한 스텝만큼 오른쪽으로 들여 쓰는 소스 작성**



주석

- 일반 문장과 달리 프로그램 내용에는 전혀 영향을 미치지 않는 설명문
- 한 줄 주석 //
- 블록 주석 /* … */

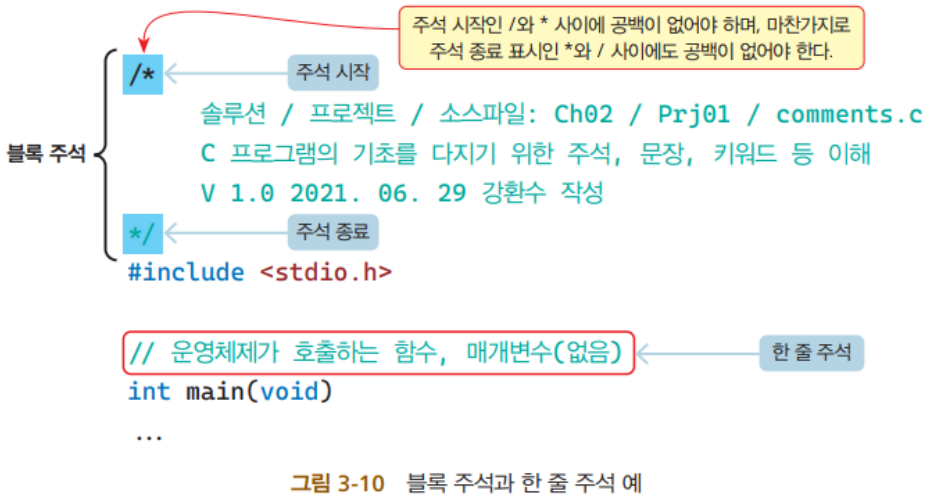


그림 3-10 블록 주석과 한 줄 주석 예

NOTE: [문법오류] 중첩된 블록주석

블록 주석에서 /* … */ 은 주석 안에 다시 주석을 이용하는 중첩된(nested) 주석은 사용할 수 없으니 주의해야 한다. 다음과 같은 중첩된 블록주석에서 컴파일러는 처음으로 나타난 /*을 블록주석 시작으로 인식하며, 이후 처음으로 표시된 */을 종료로 인식하므로 다시 나타난 */에서 컴파일 시간에 문법 오류가 발생한다.

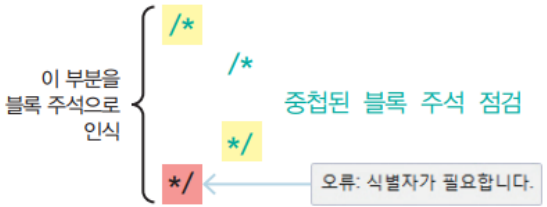


그림 3-11 다중 블록 주석 오류

C 언어 기초를 다지기 위한 주석, 문장, 키워드 등 이해

- **키워드와 식별자 그리고 주석 등을 이해하기 위한 프로젝트**
  - 솔루션과 프로젝트: ch03 / Prj01
  - 소스파일: 01comments.c

실습예제 3-1 | Prj01 | 01comments.c | C 언어 기초를 다지기 위한 주석, 문장, 키워드 등 이해 | 난이도: ★

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: ch03 / Prj01 / 01comments.c
03     C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
04     V 1.0
05  */
06  #include <stdio.h>
07
08  // 운영체제가 호출하는 함수, void로 매개변수 없음을 표시
09  int main(void)
10  {
11      puts("3장 첫 C 프로그램!\n");
12
13      printf("키워드: int void return 등\n");
14      printf("식별자: main puts printf 등\n");
15
16      return 0;
17  }
```

여러 줄에 걸친 블록 주석으로 비주얼 스튜디오에서 주석은 모두 초록색으로 표시

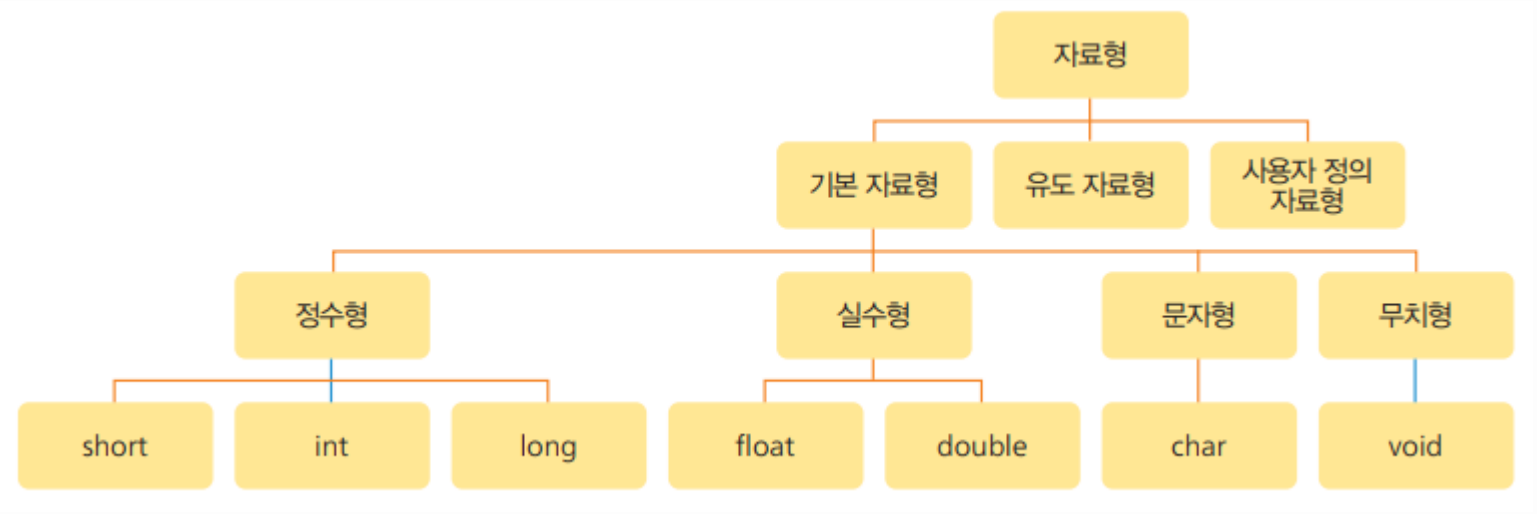
들여쓰기

결과

```
3장 첫 C 프로그램!
키워드: int void return 등
식별자: main puts printf 등
```

자료형 분류

- **자료형(data type)**
  - 프로그래밍 언어에서 자료를 식별하는 종류
- **분류**
  - 기본형(basic types)
    - **다시 정수형, 실수 형, 문자형, void**
  - 유도형(derived types)
  - 사용자 정의형(user defined types)

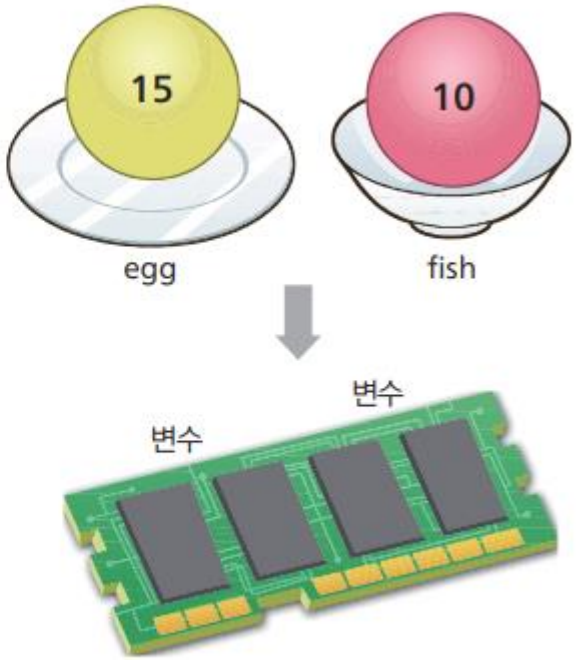


자료공간인 변수

- **변수에는 고유한 이름이 붙여지며**
  - 기억장치인 메모리에 위치
  - 선언된 자료형에 따라 변수의 저장공간 크기와 저장되는 자료 값의 종류가 결정
  - 저장되는 값에 따라 변수 값은 계속 수정
    - **마지막에 저장된 하나의 값만 저장 유지**

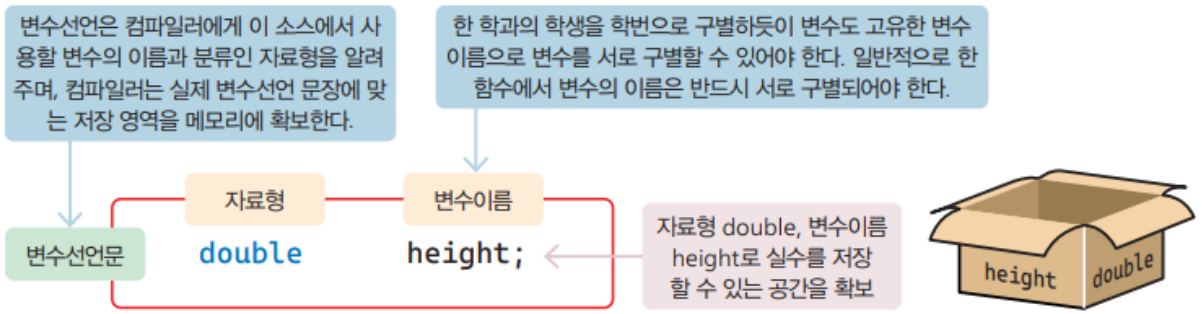
저장공간인 변수의 특징

1. 변수는 자료형을 갖고, 자료형에 따라 공간 크기와 저장될 자료값의 범주가 결정된다.
2. 저장되는 값은 수정할 수 있으며
3. 제일 마지막에 저장된 하나의 값만 유일하게 유지된다.

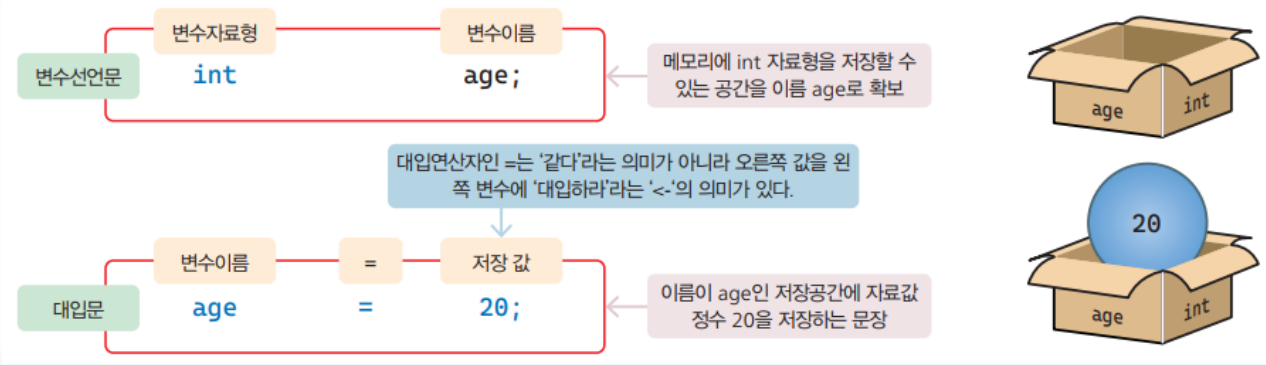


변수 선언과 값 대입

- **컴파일러에게 프로그램에서 사용할 저장 공간인 변수를 알리는 역할**
- **프로그래머 자신에게도 선언한 변수를 사용하겠다는 약속의 의미**
  - 변수는 관습적으로 소문자 사용
  - 하나의 문장으로 세미콜론으로 종료
  - 변수선언 이후에는 지정한 변수이름으로 값을 저장하거나 값을 참조

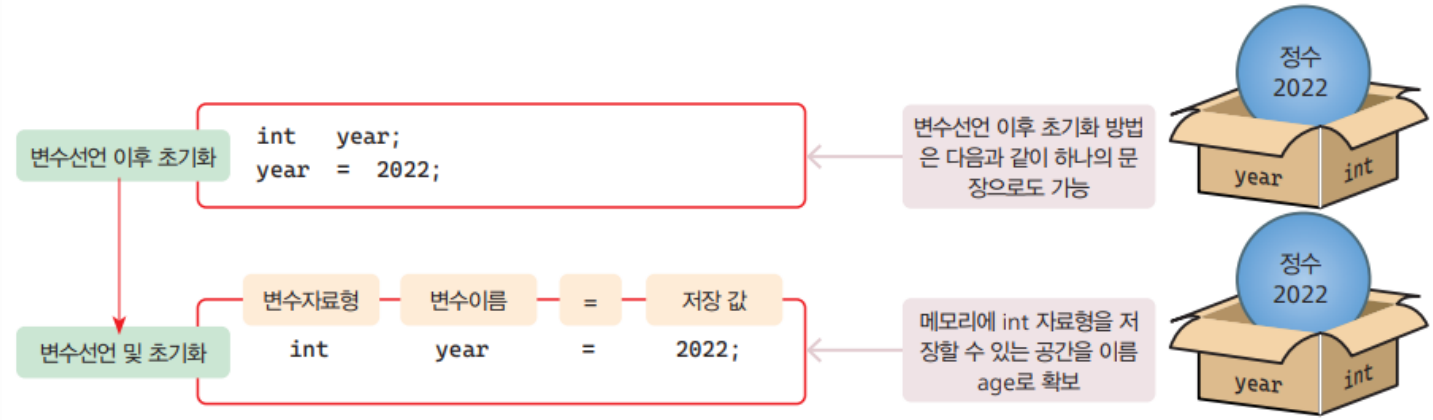


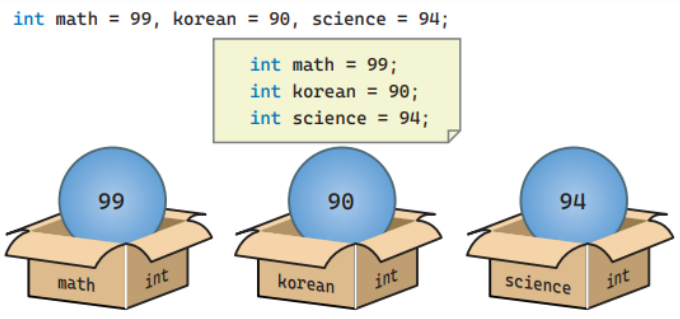
- **저장 값 대입**



변수 초기화

- **변수를 선언만 하고 자료 값에 아무것도 저장하지 않으면**
  - 원치 않는 값이 저장되고 오류가 발생
  - 변수를 선언한 이후에는 반드시 값을 저장





TIP **초기화 되지 않은 지역변수의 저장 값과 오류**

함수 내부에서 선언된 변수를 지역 변수(local variables)라 한다. 초기화 되지 않은 지역 변수는 그 저장 값이 정의되지 않으며(소위 쓰레기 값이라는 부르는 의미 없는 값이 저장), 다른 연산에 참조될 수 없다. 그러므로 초기화 되지 않은 지역 변수의 값을 다른 문장에서 참조하면 C4700 컴파일 오류가 발생한다.

```
int math = 99;
int korean = 90;
int science;
                error C4700: 초기화되지 않은 'science' 지역 변수를 사용했습니다.
int total = math + korean + science;
```

C 언어 기초를 다지기 위한 변수의 선언과 사용

실습예제 3-2 | Prj02 | 02variables.c | C 언어 기초를 다지기 위한 변수의 선언과 사용 | 난이도: ★

```c
/**
 *  소스: 02variables.c
 *  버전: V 1.0
 **/

#include <stdio.h>

int main(void)
{
   int year = 2022;    //선언과 동시에 변수 초기화

   int credits;
   credits = 15;       //선언된 변수에 초기화


   printf("%d년도\n", year);
   printf("이수학점: %d학점\n", credits);

   return 0;
}
```

결과

```
2022년도
이수학점: 15학점
```

변수 선언 위치

- **변수를 사용하기 전에만 변수를 선언**

NOTE: **ANSI C99 표준: 변수 선언 위치 제한 해제**

ANSI C99 이전에는 변수선언은 반드시 블록의 처음에 해야했다. 즉 변수선언의 위치는 선언이 아닌 다른 문장보다 먼저 나와야 하는 제한이 있었으나 ANSI C99에 이르러 이러한 변수선언 위치 제한이 없어졌다. 비주얼 스튜디오도 비주얼 스튜디오 2013 이후에는 이러한 제한이 없어졌다. 그러므로 이제 변수를 사용하기 전에만 변수를 선언하면 된다. 그러므로 위 코드는 다음과 같이 가능하다.

```c
#include <stdio.h>

int main(void)
{
   int year = 2022;    //선언과 동시에 변수 초기화
   printf("%d년도\n", year);

   int credits;
   credits = 15;       //선언된 변수에 초기화
   printf("이수학점: %d학점\n", credits);

   return 0;
}
```

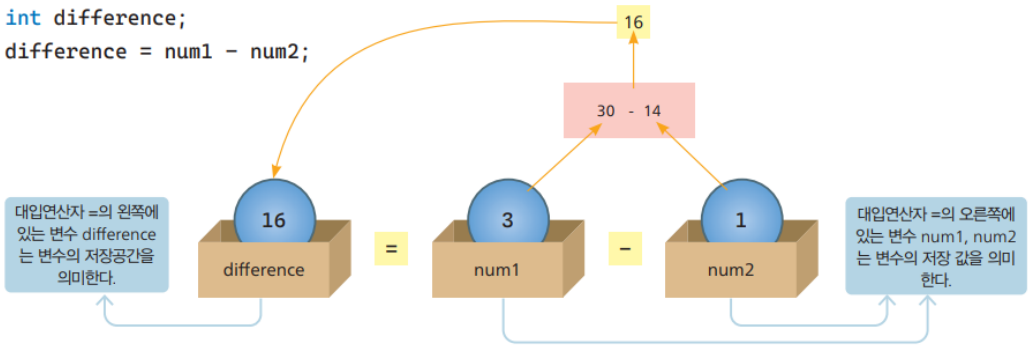
그림 3-25 변수선언 위치는 어디에도 가능

변수의 값을 더하고 빼기

실습예제 3-3 | Prj03 | 03addsub.c | 변수의 l-value와 r-value의 이해와 변수의 값을 더하고 빼기 | 난이도: ★

```c
/**
 *  소스: 03addsub.c
 *  버전: V 1.0
 **/

#include <stdio.h>

int main(void)
{
   int data1 = 20, data2 = 13;

   //대입 연산자의 왼쪽과 오른쪽에서의 변수의 의미 해석
   int diff = data1 - data2;
   int sum = data1 + data2;

   printf("data1: %d, data2: %d\n", data1, data2);
   printf("차: %d, 합: %d\n", diff, sum);

   return 0;
}
```

```c
int num1 = 30, num2 = 14;
int difference;
difference = num1 - num2;
```



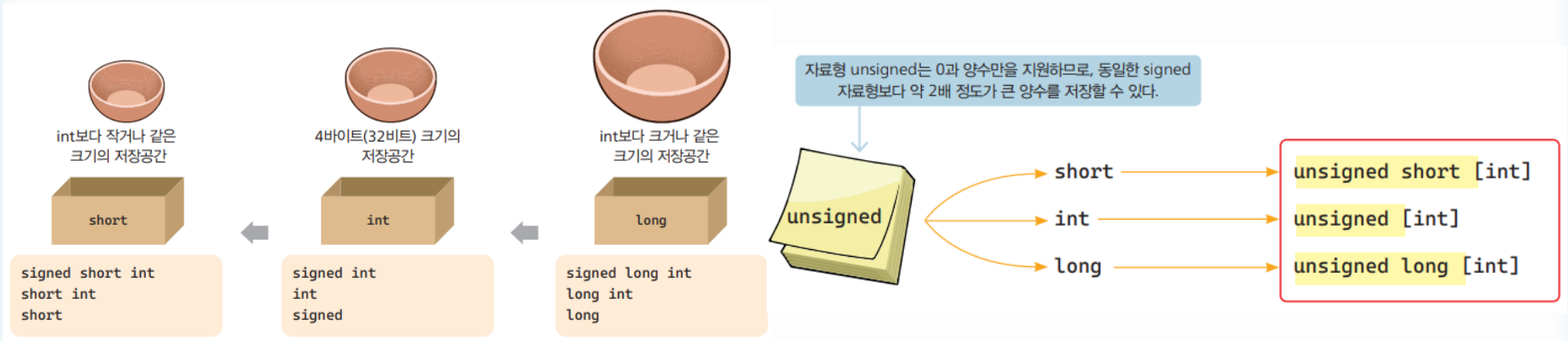
변수 diff는 l-value로 변수 자체를 의미하고, data1과 data2는 r-value로 변수에 저장된 값이 연산에 참여

결과

```
data1: 20, data2: 13
차: 7, 합: 33
```

정수형 int

- **기본 키워드는 int**
  - 365, 1024, 030, 0xF3과 같이 10진수, 8진수, 16진수의 정수가 다양하게 저장
- **short와 long**
  - 정수형 int에서 파생된 자료형
- **signed와 unsigned**
  - 음수, 0, 양수를 모두 지원하는 signed 자료형
  - 0과 양수만을 지원하는 unsigned 자료형



정수형 저장공간

- **비주얼 스튜디오**
  - short는 2바이트
  - int와 long은 모두 4바이트

표 3-1 정수 자료형의 표현 범위

음수지원 여부	자료형	크기	표현 범위
부호가 있는 정수형 signed	signed short	2 바이트	$-32,768(-2^{15}) \sim 32,767(2^{15}-1)$
	signed int	4 바이트	$-2,147,483,648(-2^{31}) \sim 2,147,483,647(2^{31}-1)$
	signed long	4 바이트	$-2,147,483,648(-2^{31}) \sim 2,147,483,647(2^{31}-1)$
부호가 없는 정수형 unsigned	unsigned short	2 바이트	$0 \sim 65,535(2^{16}-1)$
	unsigned int	4 바이트	$0 \sim 4,294,967,295(2^{32}-1)$
	unsigned long	4 바이트	$0 \sim 4,294,967,295(2^{32}-1)$

ANSI C99 표준: 자료형 long long int

NOTE: **ANSI C99 표준: 자료형 long long int**

C99 표준에 따르면 정수형을 다음과 같이 5개로 구분하고 있다. 비주얼 스튜디오에서도 저장공간 크기가 64비트인 long long int형을 지원한다. 또한 비주얼 스튜디오에서 8비트에서 64비트에 이르는 다양한 정수 자료형 키워드로 __int8, __int16, __int32, __int64 등도 지원한다.

표 3-2 C99의 규정과 비주얼 스튜디오의 다양한 정수 자료형

C99 자료형	규정	비주얼 스튜디오 지원 자료형	크기
char	8비트 이상	char, __int8	1바이트(8비트)
short	16비트 이상	short [int], __int16	2바이트(16비트)
int	16비트 이상	int	4바이트(32비트)
long int	32비트 이상	long [int], __int32	4바이트(32비트)
long long int	64비트 이상	long long [int], __int64	8바이트(64비트)

하드웨어의 발전으로 64비트 운영체제에서 64비트의 정수형의 지원은 기본이 되었다. **자료형 long long int는 간단히 long long으로 사용할 수 있으며 8바이트 64비트로 $-9,223,372,036,854,775,808(-2^{63})$에서 $+9,223,372,036,854,775,807(2^{63}-1)$까지 지원**한다. 자료형 long long은 약 922경 정도의 수를 음수와 양수로 지원하니 매우 큰 범위를 지원하며, unsigned long long은 0에서 약 1,844경까지 지원한다. 이제 long 자료형으로 불가능한 지구와 천왕성 간 거리, 해왕성과 태양 간 거리 등, 태양계를 비롯한 우주의 행성 간의 거리를 나타내는 변수로 long long 자료형을 이용하면 가능하다. 다만 long long 자료형은 함수 printf("%lld", …) 처럼 **출력 형식문자를 %lld로 기술**해야 하며, **unsigned long long 형은 %llu를 사용**한다. **자료형 long long은 하나의 키워드인 __int64 로도 지원**한다.

정수 표현을 위한 C 언어의 다양한 자료형

실습예제 3-4 | Prj04 | 04integer.c | 정수 표현을 위한 C 언어의 다양한 자료형 | 난이도: ★

```c
/* 소스: 04integer.c */

#include <stdio.h>

int main(void)
{
   short sVar = 32000;       //-32767에서 32767까지
   int   iVar = -2140000000;    //약 21억 정도까지 저장 가능

   printf("저장 값: %d %d\n", sVar, iVar);

   //C99 이후 추가된 자료형: 64비트의 정수형 지원
   long long dist1 = 2720000000000;//지구와 천왕성 간의 거리(km) 27억 2천
   __int64 dist2 = 4500000000000;    //태양과 해왕성 간의 거리(km) 45억

   printf("지구와 천왕성 간의 거리(km): %lld\n", dist1);
   printf("태양과 해왕성 간의 거리(km): %lld\n", dist2);

   return 0;
}저장 값: 32000 -2140000000
```

자료형 long long, __int64는 64비트로 약 922경까지 저장 가능

long long과 __int64를 출력할 경우, 형식제어문자를 %lld로 기술

결과

```
지구와 천왕성 간의 거리(km): 2720000000000
태양과 해왕성 간의 거리(km): 4500000000000
```

부동소수형 변수의 선언과 활용

- **키워드: float, double, long double 세 가지**
  - double형은 float형보다 표현범위가 같거나 보다 정확
  - long double형은 double형보다 표현범위가 같거나 보다 정확

표 3-4 부동소수형의 표현범위

자료형	크기	정수의 유효자릿수	표현범위
float	4 바이트	6~7	1.175494351E−38F에서 3.402823466E+38F까지
double	8 바이트	15~16	2.2250738585072014E−308에서 1.7976931348623158E+308까지
long double	8 바이트	15~16	2.2250738585072014E−308에서 1.7976931348623158E+308까지

실습예제 3-5 Prj05 05floatdouble.c 부동소수형 변수의 선언과 활용 난이도: ★

```
01  /* 소스: 05floatdouble.c */
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07      float       x = 3.14F;      //float x = 3.14;인 경우, 경고 발생
08      double      y = -3.141592;  //double 저장공간 크기는 float의 2배
09      long double z = 29.74;      //double과 long double은 저장공간이 모두 64비트
10
11      printf("부동소수 값: %f %f %f\n", x, y, z); //모두 %f로 출력 가능
12
13      return 0;
14  }
```

float 형 상수에는 3.14와 같이 반드시 F나 f를 붙이도록

결과

```
저장 값: 3.140000 -3.141592 29.740000
```

문자형 자료형 char

- **문자형 char**
  - char, signed char, unsigned char 세 가지 종류
    - **저장공간 크기는 모두 1바이트**
    - **키워드 signed와 unsigned를 함께 이용 가능**

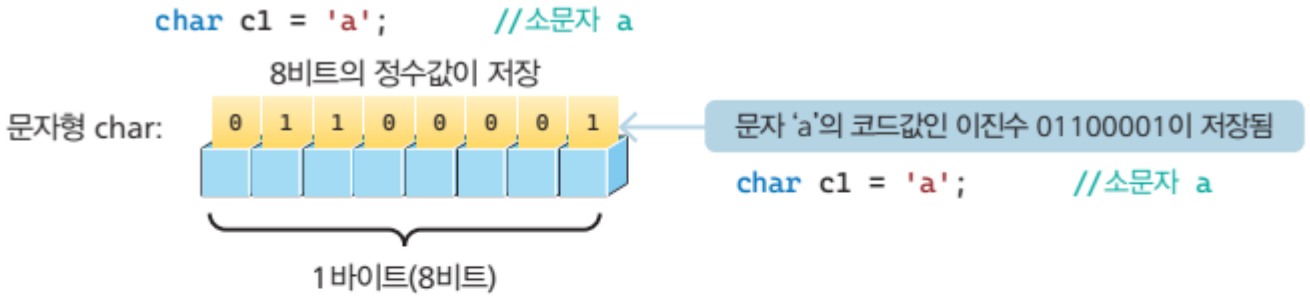


그림 3-36 문자형 자료 값의 표현과 저장공간

표 3-5 문자형의 표현범위

자료형	저장공간 크기	표현범위
char	1 바이트	−128에서 127까지(문자는 실제 0에서 127까지 이용)
signed char	1 바이트	−128에서 127까지
unsigned char	1 바이트	0 에서 255까지

아스키코드

- **아스키코드(ASCII: American Standard Code for Information)**
  - ANSI(American National Standards Institute)에서 제정한 정보 교환용 표준 코드
  - 총 127 개의 문자로 구성
- **문자형 변수의 선언과 이용**

실습예제 3-6 | Prj06 | 06char.c | 문자형 변수의 선언과 이용 | 난이도: ★

```c
/* 소스: 06char.c */

#include <stdio.h>

int main(void)
{
   char c1 = 'a';       //소문자 a
   char c2 = 65;        //대문자 A가 코드 값 65
   char c3 = '\127';    //대문자 W의 8진수 코드 값 127
   char c4 = '\x57';    //대문자 W의 16진수 코드 값 57

   printf("문자 값(문자): %c %c %c %c\n", c1, c2, c3, c4);
   printf("코드 값(번호): %d %d %d %d\n", c1, c2, c3, c4);

   return 0;
}
```

%c는 문자가 출력되며, %d는 문자의 코드 값 십진수가 출력된다.

결과

```
문자 값(문자): a A W W
코드 값(번호): 97 65 87 87
```

표 3-6 아스키코드표

문자	10진	2진	8진	16진	문자	10진	2진	8진	16진	문자	10진	2진	8진	16진	문자	10진	2진	8진	16진
NUL	0	0000 0000	0	0	SPC	32	0010 0000	40	20	@	64	0100 0000	100	40	`	96	0110 0000	140	60
SOH	1	0000 0001	1	1	!	33	0010 0001	41	21	A	65	0100 0001	101	41	a	97	0110 0001	141	61
STX	2	0000 0010	2	2	"	34	0010 0010	42	22	B	66	0100 0010	102	42	b	98	0110 0010	142	62
ETX	3	0000 0011	3	3	#	35	0010 0011	43	23	C	67	0100 0011	103	43	c	99	0110 0011	143	63
EOT	4	0000 0100	4	4	$	36	0010 0100	44	24	D	68	0100 0100	104	44	d	100	0110 0100	144	64
ENQ	5	0000 0101	5	5	%	37	0010 0101	45	25	E	69	0100 0101	105	45	e	101	0110 0101	145	65
ACK	6	0000 0110	6	6	&	38	0010 0110	46	26	F	70	0100 0110	106	46	f	102	0110 0110	146	66
BEL	7	0000 0111	7	7	'	39	0010 0111	47	27	G	71	0100 0111	107	47	g	103	0110 0111	147	67
BS	8	0000 1000	10	8	(	40	0010 1000	50	28	H	72	0100 1000	110	48	h	104	0110 1000	150	68
HT	9	0000 1001	11	9	)	41	0010 1001	51	29	I	73	0100 1001	111	49	i	105	0110 1001	151	69
LF	10	0000 1010	12	0A	*	42	0010 1010	52	2A	J	74	0100 1010	112	4A	j	106	0110 1010	152	6A
VT	11	0000 1011	13	0B	+	43	0010 1011	53	2B	K	75	0100 1011	113	4B	k	107	0110 1011	153	6B
FF	12	0000 1100	14	0C	,	44	0010 1100	54	2C	L	76	0100 1100	114	4C	l	108	0110 1100	154	6C
CR	13	0000 1101	15	0D	-	45	0010 1101	55	2D	M	77	0100 1101	115	4D	m	109	0110 1101	155	6D
SO	14	0000 1110	16	0E	.	46	0010 1110	56	2E	N	78	0100 1110	116	4E	n	110	0110 1110	156	6E
SI	15	0000 1111	17	0F	/	47	0010 1111	57	2F	O	79	0100 1111	117	4F	o	111	0110 1111	157	6F
DLE	16	0001 0000	20	10	0	48	0011 0000	60	30	P	80	0101 0000	120	50	p	112	0111 0000	160	70
DC1	17	0001 0001	21	11	1	49	0011 0001	61	31	Q	81	0101 0001	121	51	q	113	0111 0001	161	71
DC2	18	0001 0010	22	12	2	50	0011 0010	62	32	R	82	0101 0010	122	52	r	114	0111 0010	162	72
DC3	19	0001 0011	23	13	3	51	0011 0011	63	33	S	83	0101 0011	123	53	s	115	0111 0011	163	73
DC4	20	0001 0100	24	14	4	52	0011 0100	64	34	T	84	0101 0100	124	54	t	116	0111 0100	164	74
NAK	21	0001 0101	25	15	5	53	0011 0101	65	35	U	85	0101 0101	125	55	u	117	0111 0101	165	75
SYN	22	0001 0110	26	16	6	54	0011 0110	66	36	V	86	0101 0110	126	56	v	118	0111 0110	166	76
ETB	23	0001 0111	27	17	7	55	0011 0111	67	37	W	87	0101 0111	127	57	w	119	0111 0111	167	77
CAN	24	0001 1000	30	18	8	56	0011 1000	70	38	X	88	0101 1000	130	58	x	120	0111 1000	170	78
EM	25	0001 1001	31	19	9	57	0011 1001	71	39	Y	89	0101 1001	131	59	y	121	0111 1001	171	79
SUB	26	0001 1010	32	1A	:	58	0011 1010	72	3A	Z	90	0101 1010	132	5A	z	122	0111 1010	172	7A
ESC	27	0001 1011	33	1B	;	59	0011 1011	73	3B	[	91	0101 1011	133	5B	{	123	0111 1011	173	7B
FS	28	0001 1100	34	1C	<	60	0011 1100	74	3C	\	92	0101 1100	134	5C	|	124	0111 1100	174	7C
GS	29	0001 1101	35	1D	=	61	0011 1101	75	3D	]	93	0101 1101	135	5D	}	125	0111 1101	175	7D
RS	30	0001 1110	36	1E	>	62	0011 1110	76	3E	^	94	0101 1110	136	5E	~	126	0111 1110	176	7E
US	31	0001 1111	37	1F	?	63	0011 1111	77	3F	_	95	0101 1111	137	5F	DEL	127	0111 1111	177	7F

제어 문자 공백 문자 구두점 숫자 알파벳

자료형의 크기

- **기본 자료형은 long long을 포함하면 모두 14가지**

표 3-7 기본 자료형의 저장공간 크기와 표현범위(자료형에서 []은 생략 가능함)

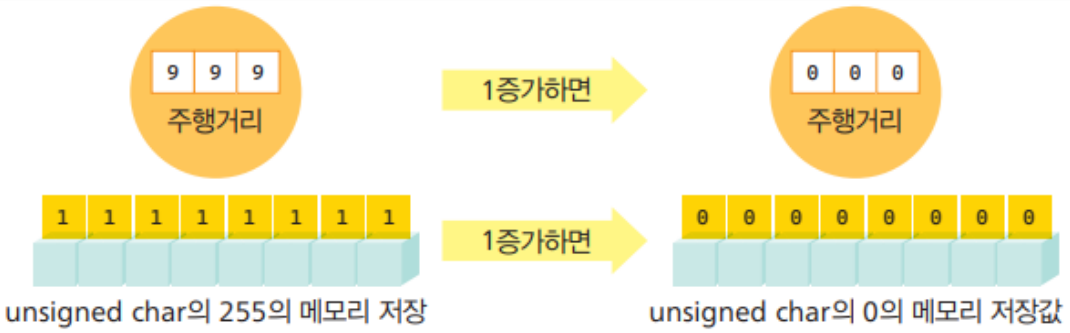
분류	자료형	크기	표현범위
문자형	char	1 바이트	$+128(+2^7) \sim 127(2^7+1)$
	signed char	1 바이트	$+128(+2^7) \sim 127(2^7+1)$
	unsigned char	1 바이트	$0 \sim 255(2^8+1)$
정수형	[signed] short [int]	2 바이트	$+32,768(+2^{15}) \sim 32,767(2^{15}+1)$
	[signed] [int]	4 바이트	$+2,147,483,648(+2^{31}) \sim 2,147,483,647(2^{31}+1)$
	[signed] long [int]	4 바이트	$+2,147,483,648(+2^{31}) \sim 2,147,483,647(2^{31}+1)$
	[signed] long long [int]	8 바이트	$9,223,372,036,854,775,808(+2^{63}) \sim 9,223,372,036,854,775,807(2^{63}+1)$
	unsigned short [int]	2 바이트	$0 \sim 65,535(2^{16}+1)$
	unsigned [int]	4 바이트	$0 \sim 4,294,967,295(2^{32}+1)$
	unsigned long [int]	4 바이트	$0 \sim 4,294,967,295(2^{32}+1)$
	[unsigned] long long [int]	8 바이트	$0 \sim 18,446,744,073,709,551,615(2^{64}+1)$
부동소수형	float	4 바이트	대략 $10^{+38} \sim 10^{38}$
	double	8 바이트	대략 $10^{+308} \sim 10^{308}$
	long double	8 바이트	대략 $10^{+308} \sim 10^{308}$

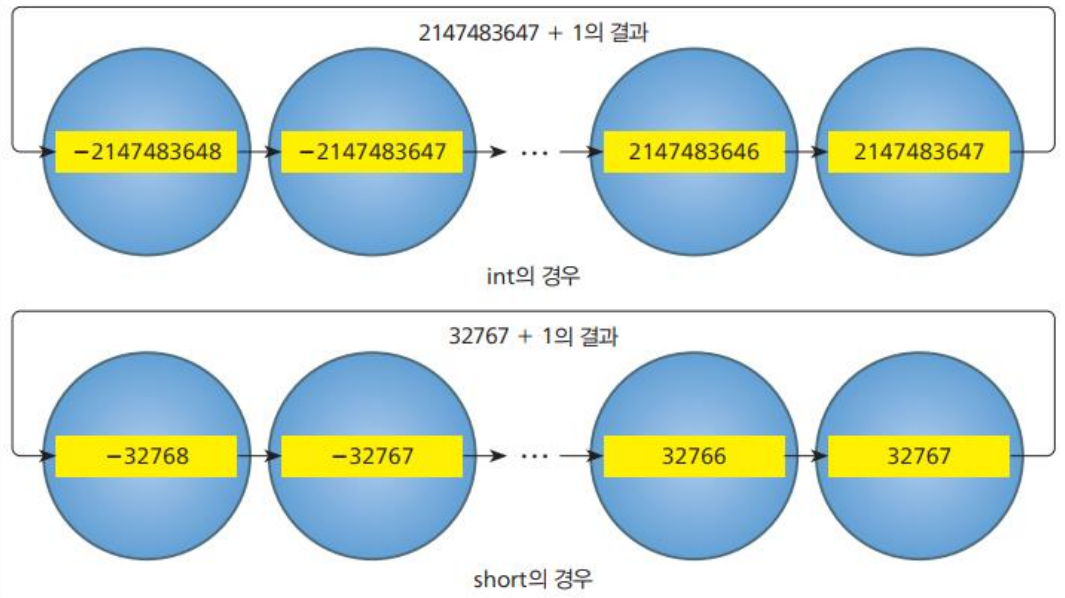
- **연산자 sizeof**
  - 자료형, 변수, 상수의 저장공간 크기를 바이트 단위 반환

```
sizeof(char)  // sizeof (자료형키워드), 괄호가 반드시 필요
sizeof 3.14   // sizeof 상수, sizeof (상수) 모두 가능
sizeof n      // sizeof 변수, sizeof (변수) 모두 가능
```

오버플로와 언더플로

- **오버플로(overflow) 또는 언더플로(underflow)가 발생**
  - 자료형의 범주에서 벗어난 값을 저장





변수의 크기와 오버플로

실습예제 3-7 | Prj07 | 07sizeflow.c | 변수의 크기와 오버플로 | 난이도: ★★

```c
/* 소스: 07sizeflow.c */

#include <stdio.h>

int main(void)
{
   printf("     자료형 : 크기(바이트)\n");
   printf("       char : %d\n", sizeof(char));
   printf("        int : %d %d\n", sizeof(int), sizeof(200));
   printf("  long long : %d %d\n", sizeof(long long), sizeof(900LL));
   printf("      float : %d %d\n", sizeof(float), sizeof 3.14F);
   printf("long double : %d %d\n", sizeof(long double), sizeof 3.24L);

   short s = 32767;
   printf("%d\n", s);
   s = s + 1;
   printf("%d\n", s);    //오버플로 발생

   return 0;
}
```

연산 32767 + 1의 결과는 32768이나
자료형 short에서 32768은 오버플로가 발생하므로 −32768이 저장됨

결과

```
자료형 : 크기(바이트)
       char : 1
        int : 4 4
  long long : 8 8
      float : 4 4
long double : 8 8
32767
-32768
```

LAB 문자 '#'과 대한민국 인구, 세계 인구의 출력

- 문자 '#'을 변수 ch에 저장하여 문자와 코드 번호를 출력
- 변수 pop1과 pop2를 적정한 자료형으로 선언하여 인구를 저장, 출력

Lab 3-1 lab1intchar.c 문자 '#'과 대한민국 인구, 세계 인구의 출력 난이도: ★

```
01  #include <stdio.h>
02
03  int main(void)
04  {
05      int ch = '#';
06      printf("%d\n", ch); //십진 코드 값 출력
07      printf("%c\n", ch); //문자 출력
08
09      int pop1 = 51800000;          //대한민국 인구 약 5100만
10      long long pop2 = 7716600000;//전 세계 인구 약 77억
11      printf("%d %lld\n", pop1, pop2);
12
13      return 0;
14  }
```

정답

```
07  printf("%c\n", ch);          //문자 출력
09  int pop1 = 51800000;         //대한민국 인구 약 5100만
```

상수의 종류와 표현 방법

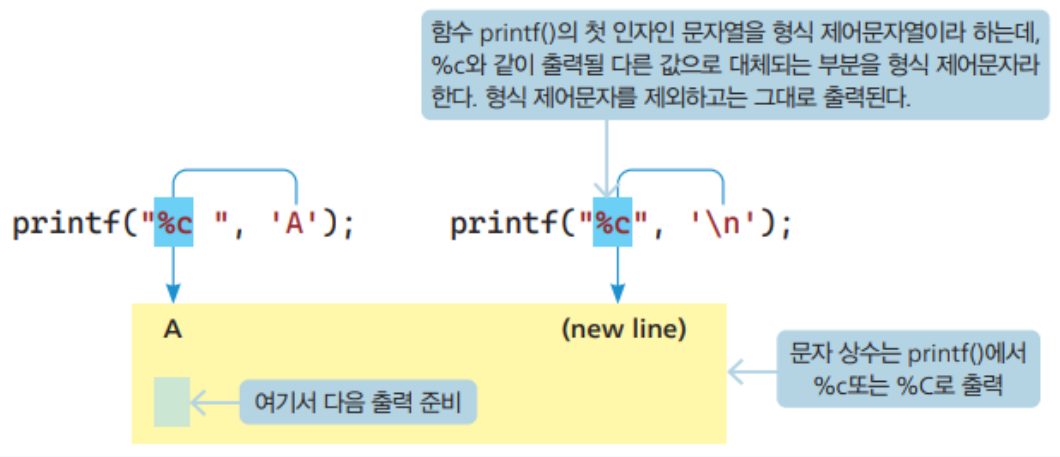
- **상수(constant)**
  - 리터럴 상수(literal constant)
    - **이름 없이 있는 그대로 표현한 자료 값이나**
  - 심볼릭 상수 (symbolic constant)
    - **이름이 있으나 정해진 하나의 값 만으로 사용되는 자료 값**
      - **const 상수(const constant)**
      - **매크로 상수(macro constant)**
      - **열거형 상수(enumeration constant)**

표 3-8 상수의 종류

구분	표현 방법	설명	예
리터럴 상수 (이름이 없는 상수)	정수형 실수형 문자 문자열 상수	다양한 상수를 있는 그대로 기술	32, 025, 0xf3, 10u, 100L, 30LL 3.2F, 3.15E3, 'A', '\n', '\0', '\24', '\x2f' "C 언어", "프로그래밍 언어\n"
심볼릭 상수 (이름이 있는 상수)	const 상수	키워드 const를 이용한 변수 선언과 같으며, 수정할 수 없는 변수 이름으로 상수 정의	const double PI = 3.141592;
	매크로 상수	전처리기 명령어 #define으로 다양한 형태를 정의	#define PI 3.141592
	열거형 상수	정수 상수 목록 정의	enum bool {FALSE, TRUE};

문자 상수 표현

- **문자 상수**
  - 문자 하나의 앞 뒤에 작은따옴표(single quote)를 넣어 표현
  - ₩ddd
    - **세 자리까지의 8진수 코드 값**
      - **문자 'A' 는 '₩141'**
  - ₩xhh
    - **₩x뒤에 한 자리에서 두 자리까지의 16진수 코드 값을 이용**
      - **'₩x61'**
- **함수 printf()**
  - 형식 제어 문자열(format control string)
    - **%c의 c는 문자 character**



이스케이프 시퀀스

- 역슬래 쉬 ₩와 문자의 조합으로 표현
  - ₩n
    - 새로운 줄(new line)

표 3-9 이스케이프 문자

제어문자 이름	영문 표현	코드값(10진수)	\ddd(8진수)	제어문자 표현	의미
널문자	NULL	0	\000	\0	아스키코드 0번
경고	BEL(Bell)	7	\007	\a	경고음이 울림
백스페이스	BS(Back Space)	8	\010	\b	커서를 한 문자 뒤로 이동
수평탭	HT(Horizontal Tab)	9	\011	\t	커서를 수평으로 다음 탭만큼 이동
개행문자	LF(Line Feed)	10	\012	\n	커서를 다음 줄로 이동
수직탭	VT(Vertical Tab)	11	\013	\v	수직으로 이동하여 탭만큼 이동
폼피드	FF(Form Feed)	12	\014	\f	새 페이지의 처음으로 이동
캐리지 리턴	CR(Carriage Return)	13	\015	\r	커서를 현재 줄의 처음으로 이동
큰따옴표	Double quote	34	\042	\"	" 문자
작은따옴표	Single quote	39	\047	\'	' 문자
역슬래쉬	Backslash	92	\134	\\	\ 문자

이스케이프 문자 등, 다양한 문자 리터럴의 표현

실습예제 3-8 | Prj08 | 08charliteral.c | 이스케이프 문자를 비롯해서 다양한 문자 리터럴의 표현 | 난이도: ★

```
01  /* 소스: 08charliteral.c */
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07      printf("%Cava", 'J');
08
09      char sq = '\'';                 //작은따옴표
10      printf("%c\7\n", '\a');         //알람 문자를 2번 출력하고 공백 줄
11      printf("%c자바 언어'\n", sq);    //문자열 내부에서는 '(작은따옴표) 그대로 사용 가능
12
13      //문자열 내부에서는 "(큰따옴표) 반드시 \"로 사용
14      printf("\"C언어\" 정말 재미있다!\n");
15
16      return 0;
17  }
```

Java 출력

경고음 소리가 출력되며, 뒤 이은 \7도 \a와 같으므로 경고음이 2번 울린다.

결과

```
Java
'자바 언어'
"C언어" 정말 재미있다!
```

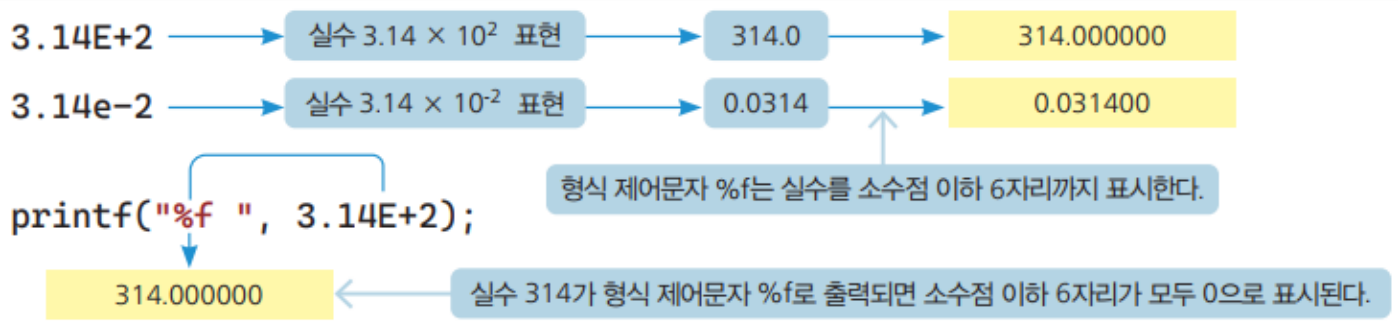
정수와 실수 리터럴 상수

- **l, L, ll, LL, u, U**
  - 정수형 리터럴 상수 표현:
    - **정수 뒤에 l 또는 L을 붙이면 long int**
- **2진수와 16진수 표현 방식**

표 3-10 8진수와 16진수의 상수 표현

표현 방법	의미	예
숫자 앞에 0(zero) 표시	8진수	06, 020, 030
숫자 앞에 0(zero)과 문자 x 조합인 0x또는 0X 앞에 표시	16진수	0xA3, 0x4, 0X5D

- **지수표현 방식**



리터럴 상수

- **실수형 리터럴 상수**
  - float, double, long double의 자료형
    - **일반 소수는 double 유형, float 상수는 숫자 뒤에 f나 F를 붙임**

표 3-11 리터럴 상수 접미어

구분	표현 방법	접미어	예
정수형 상수	unsinged (int)	u U	1000000u 24563U
	unsigned long	ul UL	87252987ul 10000000UL
	unsigned long long	ull Ull	10000000000000000ull 24500000000000000ULL
	long	l L	−20987l 76528876L
	long long	ll LL	10000000000000000ll −24500000000000000LL
실수형 상수	float	f F	3.14f 3354.9876F
	long double	l L	4356.9876l 5634.984276L

정수형과 실수형 리터럴 상수의 다양한 표현

실습예제 3-9 | Prj09 | 09numliteral.c | 정수형과 실수형 리터럴 상수의 다양한 표현 | 난이도: ★

```c
/* 소스: 09numliteral.c */

#include <stdio.h>

int main(void)
{
    printf("%d, %d\n", 010, 015);      //8진수
    printf("%d, %d\n", 10, 15);        //10진수
    printf("%d, %d\n", 0X1a, 0x15);    //16진수

    printf("%f, ", 2.71828);
    printf("%f, ", 2.71828E+2);
    printf("%f\n", 2.71828e-2);

    return 0;
}
```

부동소수의 10의 지수승 표현으로 부동소수는 printf()에서 %f로 출력

결과

```
8, 13
10, 15
26, 21
2.718280, 271.828000, 0.027183
```

심볼릭 const 상수

- **키워드 const**
  - 초기 값을 수정할 수 없으며, 이름이 있는 심볼릭 상수(constant number)

실습예제 3-10 | Prj10 | 10const.c | 키워드 const를 사용한 상수 선언 | 난이도: ★

```
01  /* 소스: 10const.c */
02
03  #include <stdio.h>
04
05  int main(void)
06  {
07      //키워드 const로 상수 만들기
08      double const e = 2.718281;    //오일러 수
09
10      //e = 2.71828;
11      printf("오일러 수 %f\n", e);
12
13      return 0;
14  }
```

상수는 수정할 수 없으므로 주석을 빼면 컴파일 오류가 발생

결과

```
오일러 수 2.718281
```

열거형 상수

- **키워드 enum**
- **정수형 상수 목록 집합을 정의하는 자료형**
  - 목록 첫 상수의 기본 값이 0이며
    - **다음부터 1씩 증가하는 방식으로 상수 값이 자동으로 부여**
  - 상수 값을 지정한 상수는 그 값으로
  - 따로 지정되지 않은 첫 번째 상수는 0이며,
    - **중간 상수는 앞의 상수보다 1씩 증가한 상수 값으로 정의**

```
enum SHAPE { POINT, LINE, TRI = 3, RECT, OCTA = 8, CIRCLE };
```

정수형 상수 목록

POINT	0	LINE	1	TRI	3	RECT	4	OCTA	8	CIRCLE	9

```
enum boolean  {FALSE, TRUE};
enum city  {SEOUL, INCHEON, DAEGU, PUSAN};
enum OS  {WINDOW, OSX = 3, ANDROID, IOS = 7, LINUX};
enum pl {c = 1972, cpp = 1983, java = 1995, csharp = 2000};
```

enum의 열거형 상수

실습예제 3-11 | Prj11 | 11enum.c | enum의 열거형 상수 | 난이도: ★★

```c
/* 소스: 11enum.c */

#include <stdio.h>

int main(void)
{
    //키워드 enum으로 열거형 정수 상수 목록 만들기
    enum DAY { SUN, MON, TUE, WED, THU, FRI, SAT };
    printf("%d %d\n", SUN, THU); //0 4

    //상수 목록에서 특정한 정수 지정 가능
    enum SHAPE { POINT, LINE, TRI = 3, RECT, OCTA = 8, CIRCLE };
    printf("LINE: %d, RECT: %d, CIRCLE: %d\n", LINE, RECT, CIRCLE);

    enum pl { c = 1972, cpp = 1983, java = 1995, csharp = 2000 };
    printf("c: %d, cpp: %d, java: %d\n", c, cpp, java);

    return 0;
}
```

상수 SUN은 0에서부터 순차적으로 1씩 증가되어 지정되며, 상수 THU은 4로, SAT는 6으로 지정

결과

```
0 4
LINE: 1, RECT: 4, CIRCLE: 9
c: 1972, cpp: 1983, java: 1995
```

전처리기 지시자 #define

- **매크로 상수(macro constant)**
  - 전처리 지시자 #define
    - **매크로 상수를 정의하는 지시자**
    - **심볼릭 상수로 주로 대문자 이름으로 정의**
- **헤더파일 limits.h, float.h에 정의**
  - 문자형과 정수형의 최대 최소 상수
  - 부동소수형의 최대 최소 상수.

분류	헤더파일	자료형	관련 상수이름
문자형	limits.h	char	CHAR_MIM, CHAR_MAX
		signed char	SCHAR_MIM, SCHAR_MAX
		unsigned char	UCHAR_MAX
정수형	limits.h	[signed] short [int]	SHRT_MIM, SHRT_MAX
		[signed] [int]	INT_MIM, INT_MAX
		[signed] long [int]	LONG_MIM, LONG_MAX
		[signed] long long [int]	LLONG_MIM, LLONG_MAX
		unsigned short [int]	USHRT_MAX
		unsigned [int]	UINT_MAX
		unsigned long [int]	ULONG_MAX
		unsigned long long [int]	ULLONG_MAX
부동소수형	float.h	float	FLT_MIM, FLT_MAX
		double	DBL_MIN, DBL_MAX
		long double	LDBL_MIN, LDBL_MAX

LAB 부동소수형 최대 최소 매크로 상수 출력

- 헤더파일 float.h에 매크로의 최대 최소 상수를 출력

Lab 3-2 | lab2minmaxfloat.c | 부동소수형 최대 최소 매크로 상수 출력 | 난이도: ★

```
01  #include <stdio.h>
02  #include <float.h>        //부동소수형 상수가 정의된 헤더파일 삽입
03
04  int main(void)
05  {
06      printf("float 범위: %e %e\n",          , FLT_MAX);
07      printf("double 범위: %e %e\n", DBL_MIN,          );
08      printf("long double 범위: %e %e\n", LDBL_MIN, LDBL_MAX);
09
10      return 0;
11  }
```

정답

```
06  printf("float 범위: %e %e\n", FLT_MIN, FLT_MAX);
07  printf("double 범위: %e %e\n", DBL_MIN, DBL_MAX);
```

감사합니다.