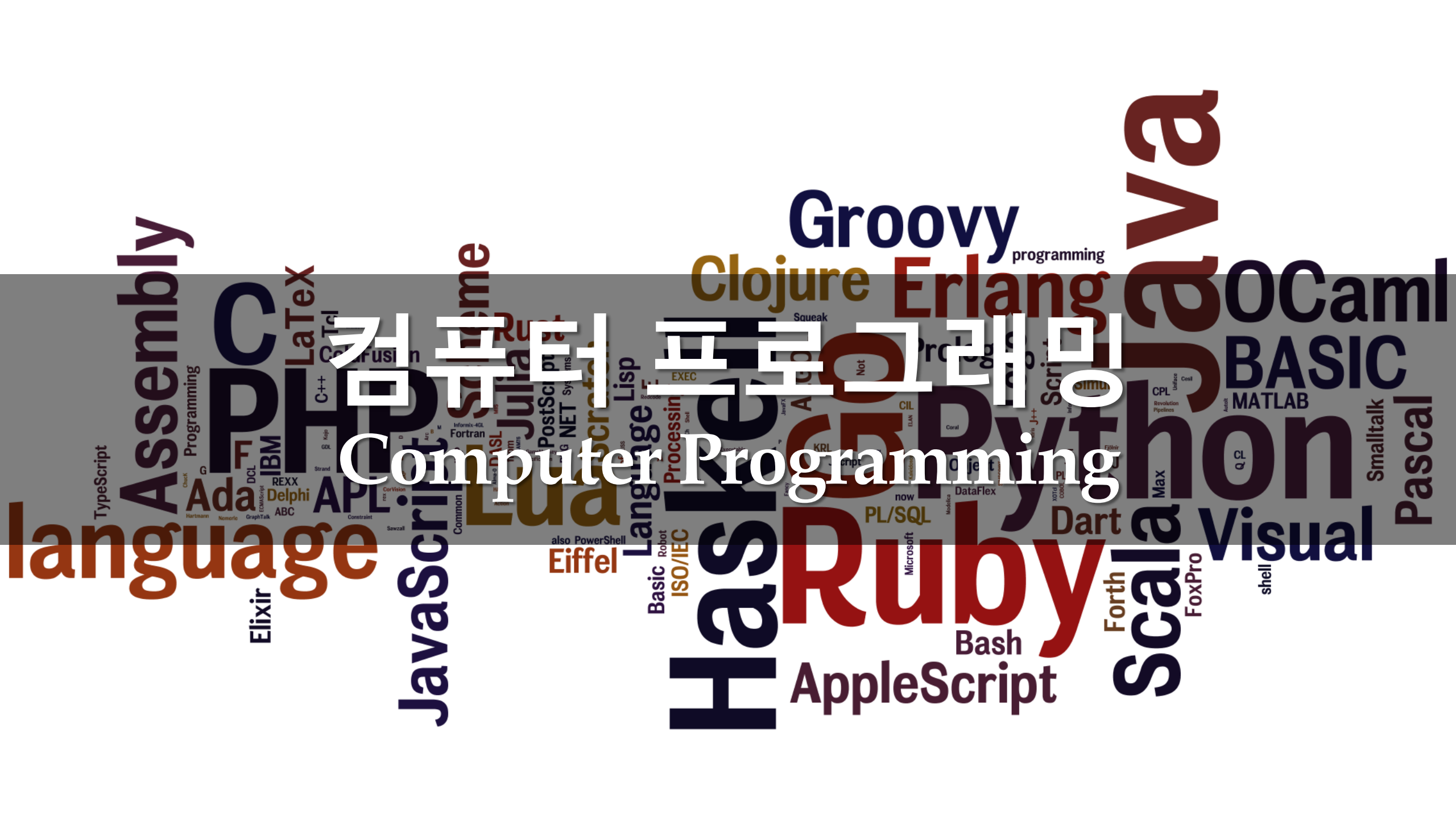
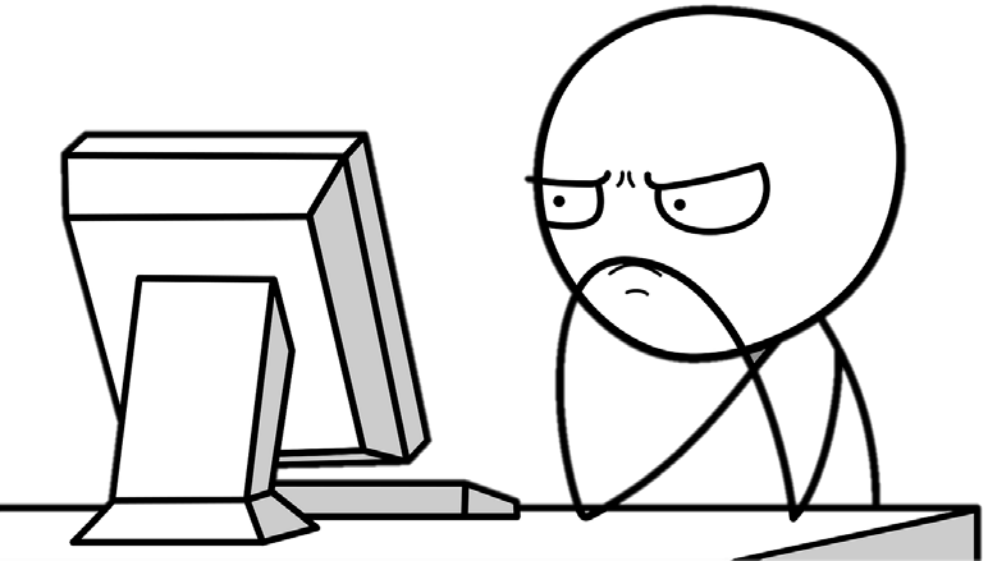

컴퓨터 프로그래밍
Computer Programming

03
자료형과 변수

목차

1. 프로그래밍 기초
2. 자료형과 변수 선언
3. 기본 자료형
4. 상수 표현방법

1. 프로그래밍 기초

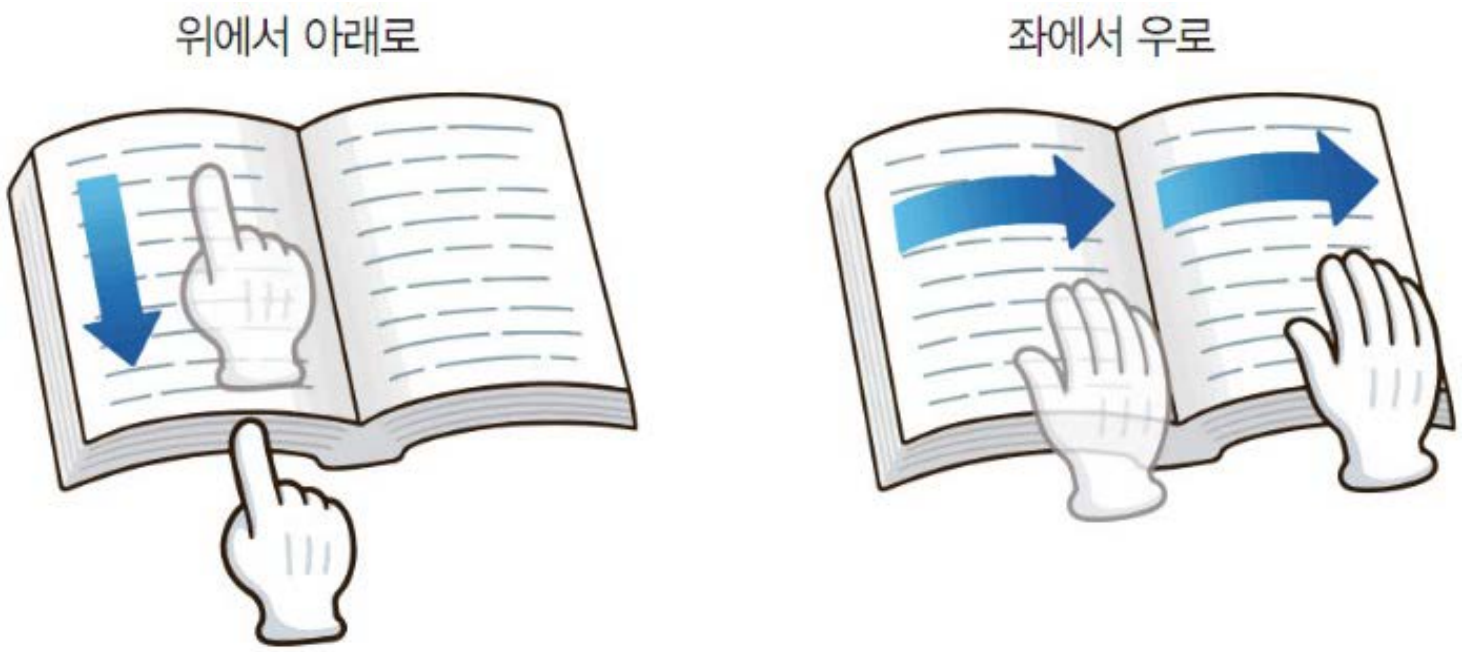


프로그램

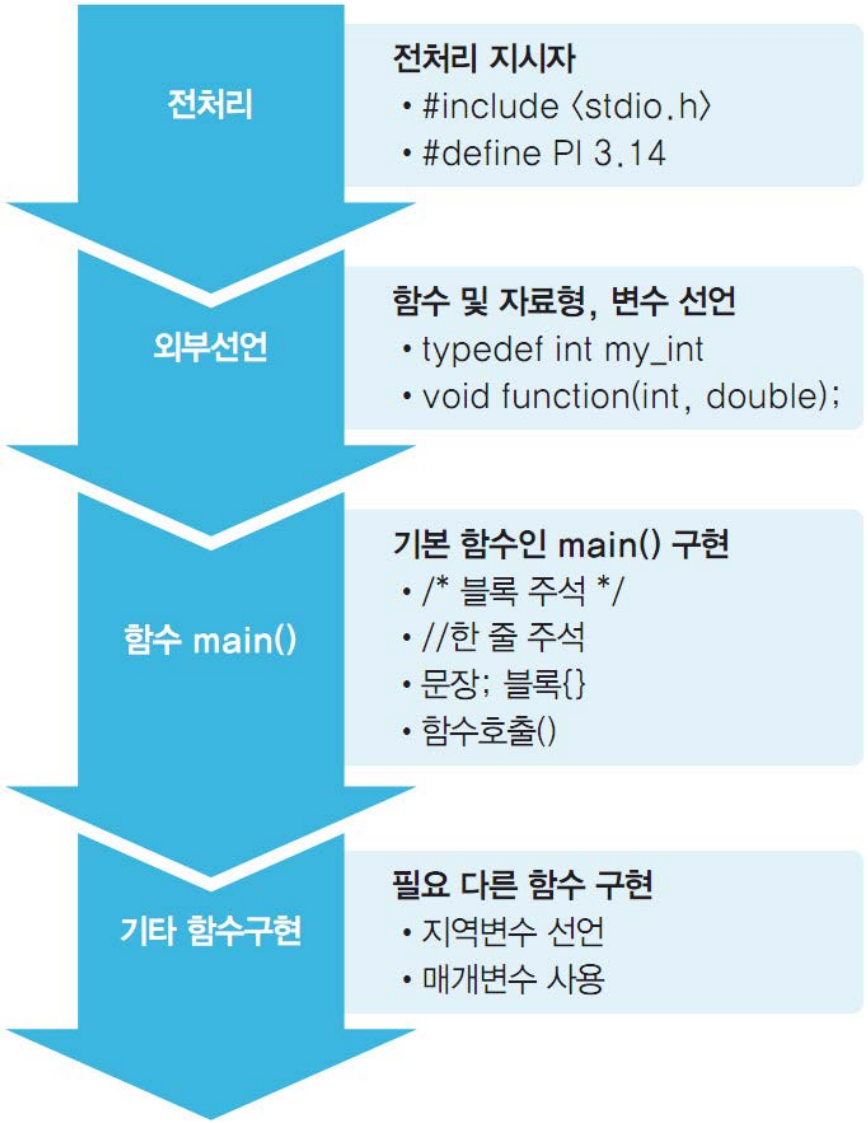
- C 프로그램은 하나 이상의 여러 함수가 모여 한 프로그램으로 구성
- 비주얼 스튜디오
  - 솔루션은 여러 개의 프로젝트로 구성
  - 다시 프로젝트는 여러 소스파일을 포함한 여러 자원resource으로 구성
- 한 프로젝트는 단 하나의 함수 main()과 다른 여러 함수로 구현
- 최종적으로 프로젝트이름으로 하나의 실행 파일이 생성

프로그램 시작과 종료

- 시작된 함수 main() 내부
- 위에서 아래로, 좌에서 우로, 문장이 위치한 순서대로 실행
- 중간에 puts(…)와 power(…)처럼 함수가 호출
  - 호출된 함수로 이동하여 그 함수를 모두 실행한 후 다시 돌아와 그 이후의 문장을 실행

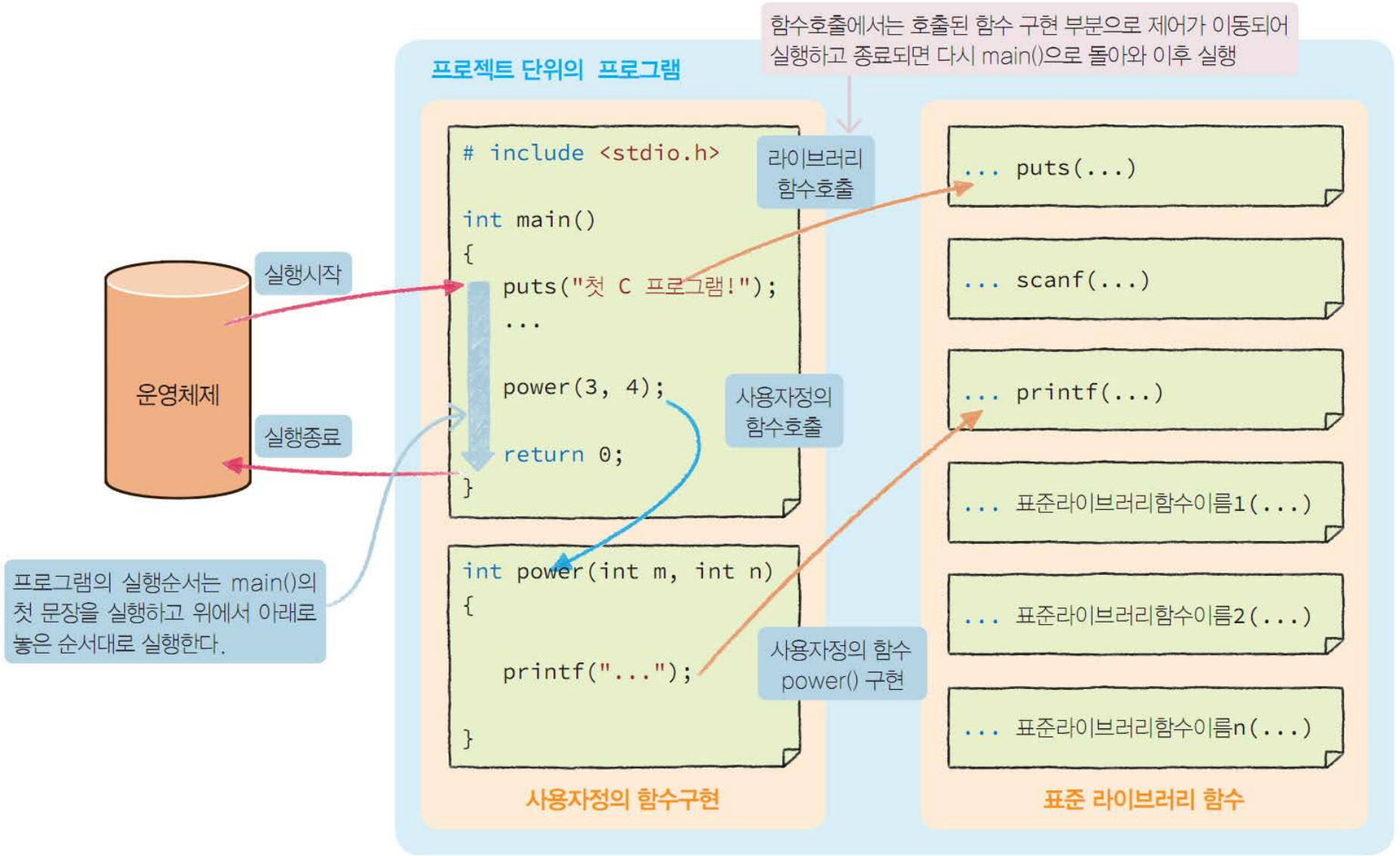


C 프로그램 소스의 구조



- C 프로그램은 적어도 main() 함수 하나는 구현되어야 응용 프로그램으로 실행 가능
  - 함수의 구현은 여러 문장으로 구성
  - 프로그래머가 만든 사용자정의 함수 또는 시스템이 만든 표준 라이브러리 함수 호출이 실행

C 프로그램의 구성과 실행



키워드

- 문법적으로 고유한 의미를 갖는 예약된 단어
- '예약'되었다는 의미
  - 프로그램 코드를 작성하는 사람이 이 단어들을 다른 용도로 사용해서는 안 된다는 뜻
- 예약어reserved word
  - C프로그램에서는 미국표준화위원회ANSI: American National Standard Institute
  - 지정한 32개의 기본적인 단어
- 비주얼 스튜디오 편집기
  - 키워드는 기본적으로 파란색으로 표시

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

식별자identifiers

- 식별자 사용 규칙
  - 구성
    - 영문자(대소문자 알파벳)
    - 숫자(0~9)
    - 밑줄(_)로 구성
  - 식별자의 첫 문자로 숫자가 나올 수 없음
  - 프로그램 내부의 일정한 영역에서는 서로 구별
  - 키워드는 식별자로 이용할 수 없음
  - 식별자는 대소문자를 모두 구별
    - 예를 들어, 변수 Count, count, COUNT는 모두 다른 변수
  - 식별자의 중간에 공백(space)문자가 들어갈 수 없음
- 프로그래머가 자기 마음대로 정의해서 사용하는 단어
  - 변수이름
    - age, year 등
  - 함수이름
    - puts, main, printf 등

식별자identifiers

식별자 구성 규칙

1. 숫자는 맨 앞에 올 수 없다.
2. 대소문자는 구별된다.
3. 중간에 공백문자(space)가 들어갈 수 없다.
4. 키워드는 식별자로 사용할 수 없다
5. 알파벳과 _를 제외한 문자는 사용할 수 없다

다음은 식별자의 바른 사용의 예입니다.

1. worldcup2014.
2. Count, count, COUNT
3. number1, number2
4. _systemID
5. my_name

다음은 식별자의 잘못된 사용의 예입니다.

1. 2012olympic.
2. C#.
3. case, if
4. employee id
5. nx+ny

문장 statement

- 컴퓨터에게 명령을 내리는 최소 단위를 문장statement
- 문장은 마지막에 세미콜론 ;으로 종료
  - 문장 마지막에 ;을 빠뜨리면
  - 컴파일 시간에 문법 오류가 발생

"아버지 가방에 들어가신다." ← 띄어쓰기가 잘못된 문장

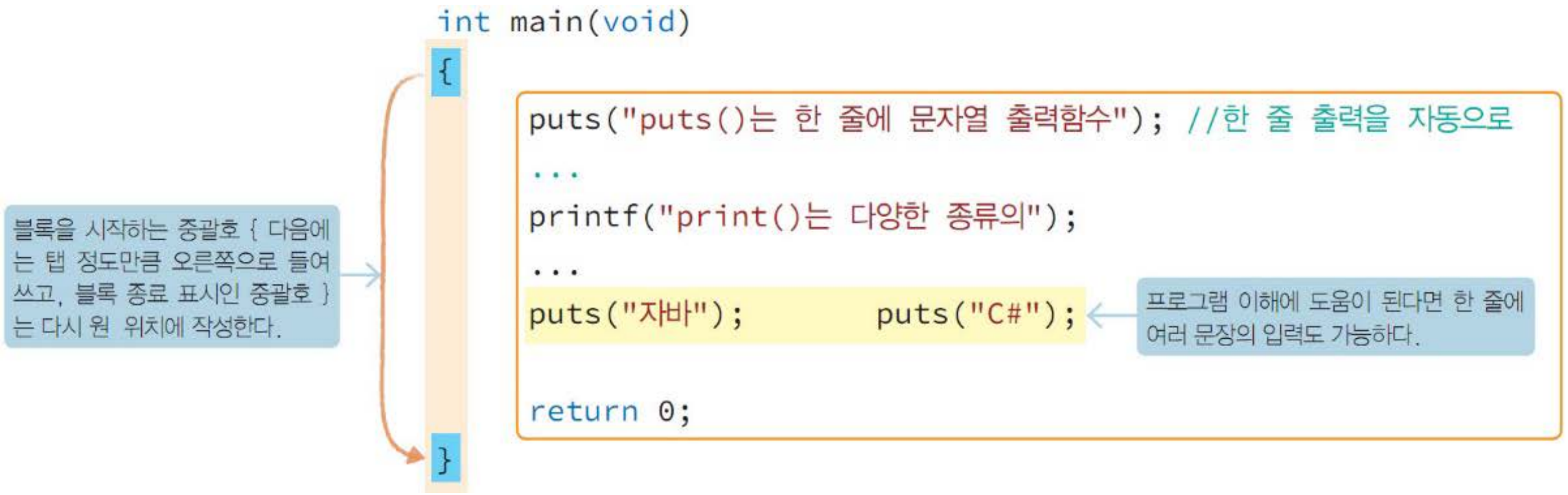
C 언어가 재미있나요 ← ?가 빠져 잘못된 문장

```
puts("C 언어")
```
← 문장 마지막에; 이 없어 문법 오류가 발생

```
print("C 언어");
```
← 함수 이름에서 마지막에 f가 빠져 문법 오류가 발생

블록과 들여쓰기

- 블록
  - 여러 개의 문장을 묶으면 블록block
  - {문장1, 문장2, …} 처럼 중괄호로 열고 닫음
- 들여쓰기
  - 블록 내부에서 문장들을 탭tab 정도만큼 오른쪽으로 들여 쓰는 소스 작성 방식



적절한 소스 구성

- 적절한 줄 구분과 빈 줄 삽입, 그리고 들여쓰기는 프로그램의 이해력을 돕는데 매우 중요한 요소

식별자 구성 규칙

```
#include <stdio.h>

int main(void)
{
    puts("C 언어");

    return 0;
}
```

오른쪽은 왼쪽 소스와 비교하여 상대적으로 이해하기 어려운 소스이다.

식별자 구성 규칙

```
#include <stdio.h>
int main(void) { puts("C 언어");
return 0;
}
```

주석의 정의와 중요성

- 주석comments
  - 문장과 달리 프로그램 내용에는 전혀 영향을 미치지 않는 설명문
- 주석은 매우 중요한 프로그램의 과정
  - 자신을 비롯한 이 소스를 보는 모든 사람이 이해할 수 있도록 도움이 되는 설명을 담고 있어야 함
  - 주석은 개발 시에도 필요하지만 개발 이후에 유지보수 기간에는 더욱 더 중요한 역할
  - 개인이나 팀, 또는 프로젝트에서 주석처리 형식을 통일성 있게 만들어 꼼꼼히 작성할 필요
  - 잘 처리된 주석이란 시각적으로 정돈된 느낌
  - 프로그램의 내용을 적절히 설명

주석의 예제

```c
/*
솔루션 / 프로젝트 / 소스파일: Ch02 /Prj01 / comments.c
C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
V 1.0 2018. 08. 29(수) 이수안 작성
*/


/**
* 소스: comments.c
* 내용: C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
* 버전: V 1.0 2018. 08. 29(수) 이수안 작성
**/

```

```c
/*************************************************************************
소스: comments.c
내용: C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
버전: V 1.0 2018. 08. 29(수) 이수안 작성
*************************************************************************/


/*************************************************************************
// 소스: comments.c
// 내용: C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
// 버전: V 1.0 2018. 08. 29(수) 이수안 작성
*************************************************************************/

```

주석 처리 방법

- 한 줄 주석 //
  - // 이후부터 그 줄의 마지막까지 주석으로 인식
  - 현재 줄의 처음이나, 문장 뒤부터 중간에서의 주석은 주로 한 줄 주석을 이용
    - 구현 방법이나 작동 방식을 설명하는 주석으로 처리
- 블록 주석 /* … */
  - /* … */은 여러 줄에 걸쳐 설명을 사용할 때 이용
  - 주석 시작은 /*로 표시하며, 종료는 */로 표시
  - 프로그램의 처음 부분에는 주로 여러 줄에 걸친 블록 주석을 사용
    - 작성자와 소스의 목적
    - 프로그램의 전체적 구조와 저작권 정보 등 파일 관련 정보
  - 함수의 시작 부분
    - 프로그램의 기능과 함께 매개변수 등을 주석처리
  - 블록주석의 중첩은 오류

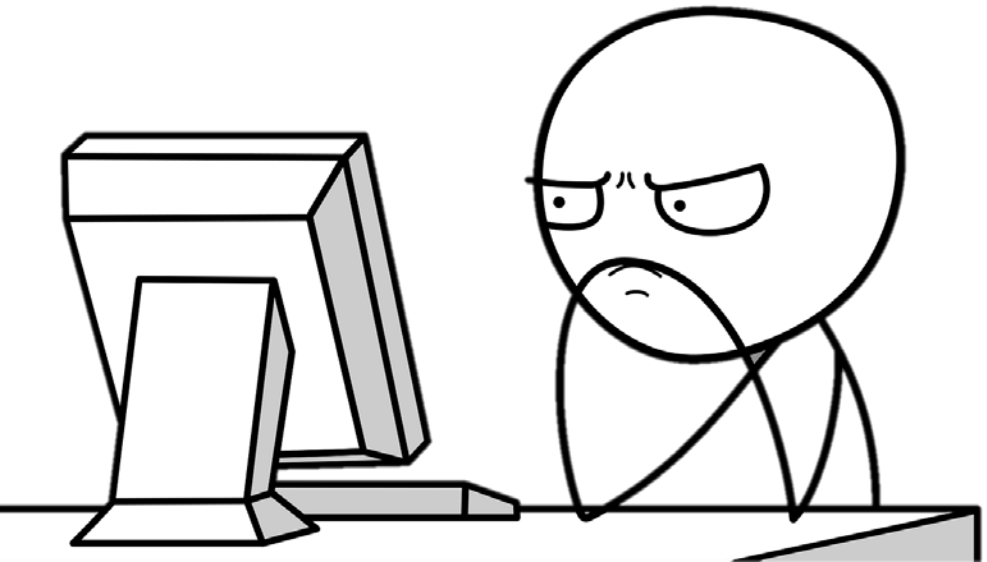
Source Code #01: comments.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch02 / Project01 / comments.c
    C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
    V 1.0 2018.
*/
#include <stdio.h>

// 운영체제가 호출하는 함수, 매개변수(없음)
int main(void)
{
    puts("2장 첫 C 프로그램!\n");

    printf("키워드: int void return 등\n");
    printf("식별자: main puts printf 등\n");
    printf("블록: { ... }\n");

    //인자인 문자열 내부는 //주석도 일반 문자열로 인식
    printf("한 줄 주석: // 이 줄 끝까지 한 줄 주석입니다.\n");
    // /*블록 주석*/도 일반 문자열로 인식
    printf("블록 주석: /* 여러 줄에 걸친\n블록 주석입니다.*/ \n");

    return 0;
}
```

```
2장 첫 C 프로그램!

키워드: int void return 등
식별자: main puts printf 등
블록: { ... }
한 줄 주석: // 이 줄 끝까지 한 줄 주석입니다.
블록 주석: /* 여러 줄에 걸친
블록 주석입니다.*/
```

- 키워드와 식별자 그리고 주석 등을 이해하기 위한 간단한 소스
  - 솔루션과 프로젝트: Ch03 / Project01
  - 소스파일: comments.c
- 한 줄 주석 //에서 시작 표시인 // 이후부터는 어떤 입력도 주석으로 인식
- 한 줄 // 주석은 중복되어도 상관없음
- /* 등이 나타나도 아무 문제가 없음
- 주석은 문자열 내부에서는 단지 문자열이지 주석으로 인식되지 못함
- 문자열에서의 \n
  - 특수문자 '\n'은 새로운 줄(new line)로 이동을 지시하는 문자로 문자열 내부에 사용이 가능

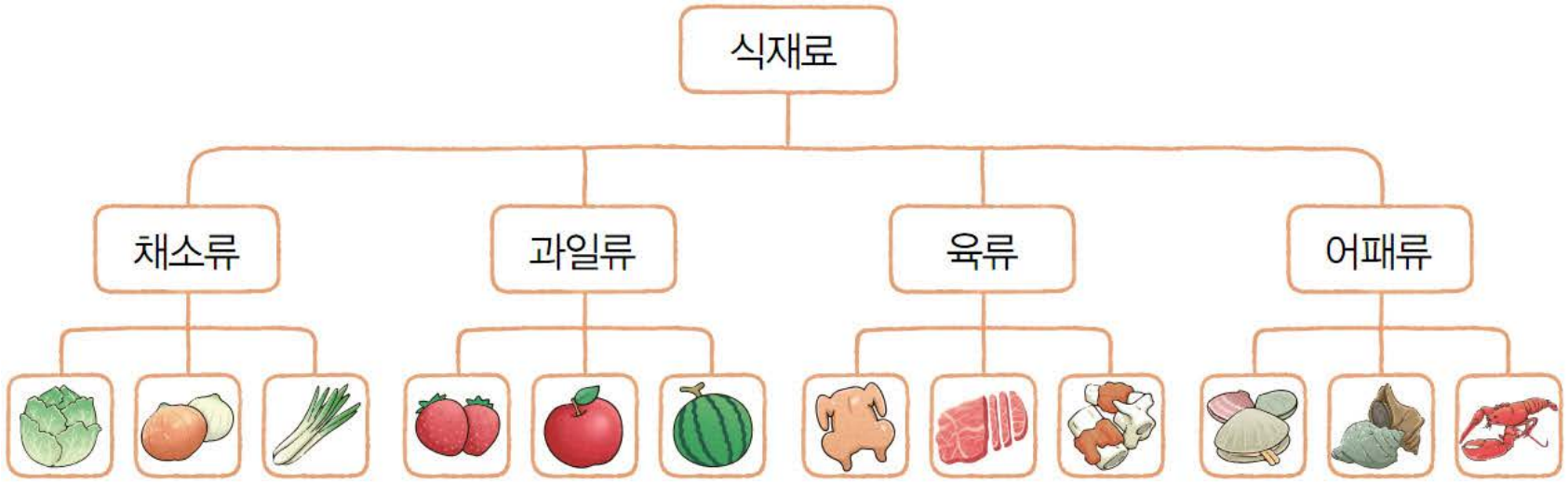
2. 자료형과 변수선언

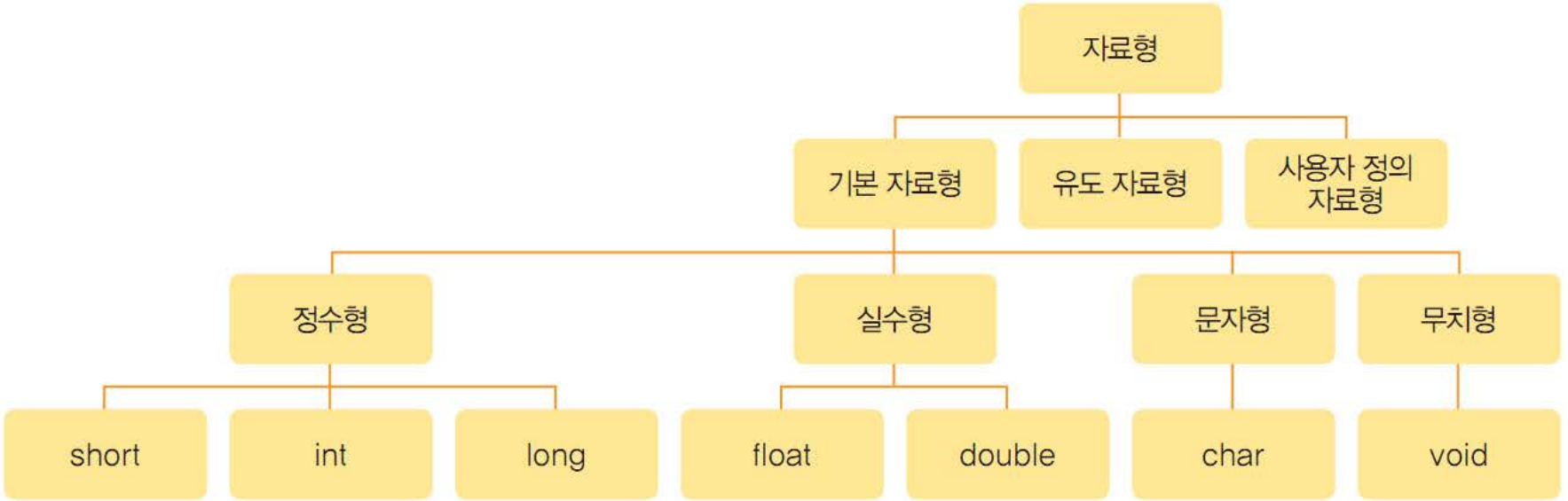


자료형

- 프로그래밍 언어에서 자료를 식별하는 종류
  - 기본형basic types
  - 유도형derived types
  - 사용자 정의형user defined types

자료형


식재료
채소류
과일류
육류
어패류


자료형
기본 자료형
유도 자료형
사용자 정의 자료형
정수형
실수형
문자형
무치형
short
int
long
float
double
char
void

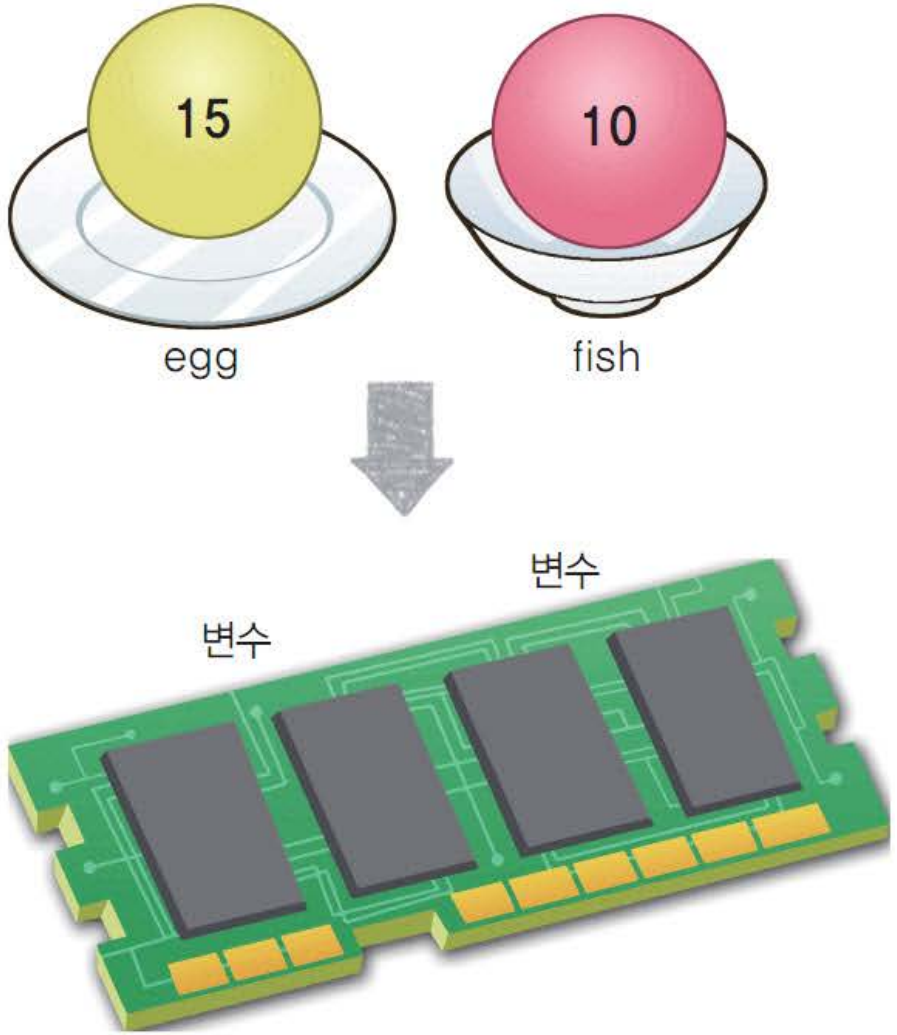
저장공간

- 저장 공간을 변수variables라 부름
- 변수에는 고유한 이름이 주어짐
- 물리적으로 기억장치인 메모리에 위치
- 변수에 여러 값을 저장 가능
  - 저장되는 값에 따라 변수값은 바뀔 수 있음
  - 마지막에 저장된 하나의 값만 저장 유지

저장공간

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.



변수선언

- 그릇을 변수라고 한다면 그릇에 이름을 붙여 준비하는 것을 변수선언
- 컴파일러에게 프로그램에서 사용할 저장 공간인 변수를 알리는 역할
- 자신에게도 변수를 사용하겠다는 약속 의미
- 변수는 고유한 이름이 붙여지고, 자료값이 저장되는 영역

변수선언

자료형을 지정한 후 변수이름을 나열

- int, double, float와 같이 자료형 키워드를 사용
- 변수이름은 관습적으로 소문자를 이용
  - 사용 목적에 알맞은 이름으로 특정한 영역에서 중복되지 않도록
  - 변수선언도 하나의 문장이므로 세미콜론으로 종료
- 변수선언 이후에는 정해진 변수이름으로 값을 저장하거나 값을 참조 가능

자료형을 지정한 후 변수이름을 나열

변수선언은 컴파일러에게 이 소스에서 사용할 변수의 이름과 분류인 자료형을 알려주며, 컴파일러는 실제 변수선언 문장에 맞는 저장 영역을 메모리에 확보한다.

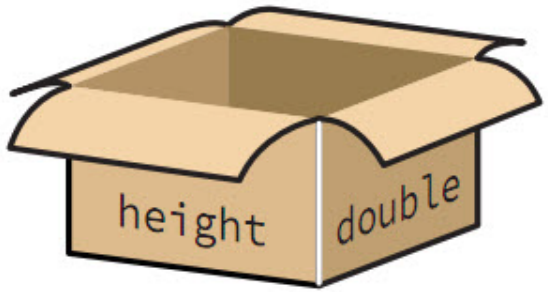
한 학과의 학생을 학번으로 구별하듯이 변수도 고유한 변수 이름으로 변수를 서로 구별할 수 있어야 한다. 일반적으로 한 함수에서 변수의 이름은 반드시 서로 구별되어야 한다.

변수선언문

자료형 변수이름

```
double height;
```

자료형 double, 변수이름 height로 실수를 저장할 수 있는 공간을 확보



Source Code #02: var.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project2 / var.c
    C 프로그램의 기초를 다지기 변수선언 이해
    V 1.0 2018.
*/

 #include <stdio.h>

int main(void)
{
    int snum;        //변수 선언
    int credits;

    snum = 20183021;//값 지정
    credits = 18;

    printf("학번: %d\n", snum);
    printf("신청학점: %d\n", credits);

    return 0;
}

```

```
학번: 20183021
신청학점: 18
```

- 정수형 int 변수 snum 선언
- 정수형 int 변수 credits 선언
- 변수 snum에 학번 저장
- 변수 credits에 신청한 학점 저장
- 각각 변수 snum과 credits에 저장된 값을 출력

대입문

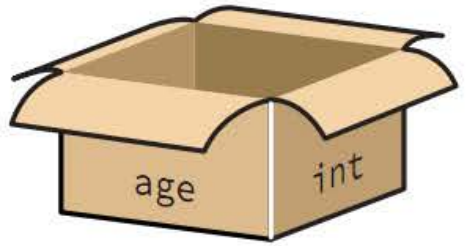
- 원하는 자료값을 선언된 변수에 저장
- 대입연산자assignment operator표시인 '='를 사용
- 오른쪽에 위치한 값을 이미 선언된 왼쪽 변수에 저장한다라는 의미
- 대입문assignment statement
  - 변수명 age를 int 형으로 선언한 후, 변수 age에 20을 저장하는 문장
  - 가장 마지막에 저장된 값만이 남음

대입문

변수선언문

변수자료형 `int` 변수이름 `age;`

메모리에 int 자료형을 저장할 수 있는 공간을 이름 age로 확보

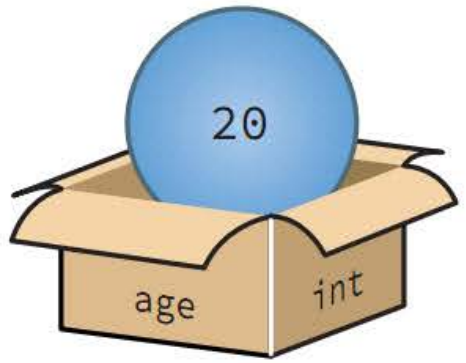


대입 연산자인 =는 '같다'라는 의미가 아니라 오른쪽 값을 왼쪽 변수에 '대입하라'라는 '<-'의 의미가 있다.

대입문

변수이름 `age` = 저장 값 `20;`

이름이 age인 저장공간에 자료값 정수 20을 저장하는 문장



Source Code #03: sum.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project3 / sum.c
    변수 초기화 이해
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    int math = 99;      //선언과 동시에 변수 초기화
    int korean = 90;

    int science;
    science = 94;       //선언된 변수에 초기화

    //더하기 기호인 +를 사용하여 총합을 변수 total에 선언하면서 저장
    int total = math + korean + science;

    printf("수학: %d\n", math);
    printf("국어: %d\n", korean);
    printf("과학: %d\n", science);
    printf("총점 %d\n", total);

    return 0;
}
```

```
수학: 99
국어: 90
과학: 94
총점 283
```

- 변수 math, korean, science, total 선언과 사용

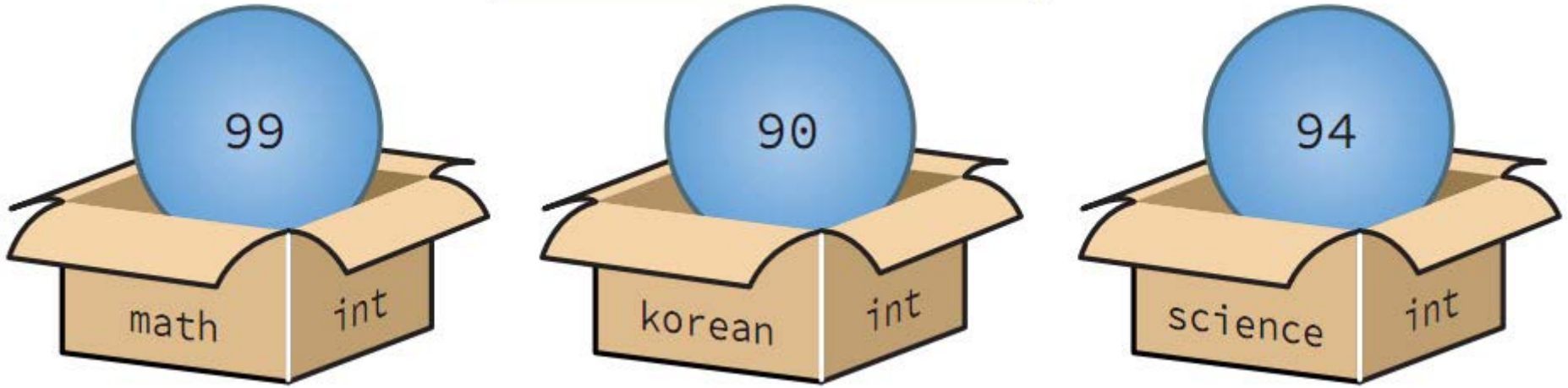
초기값 저장

- 변수를 선언하면서 변수명 이후에 대입연산자 =와 수식이나 값이 오면 바로 지정한 값으로 초기값이 저장
- 오류가 발생
  - 변수를 선언만 하고 자료값이 아무것도 저장하지 않으면 원치 않는 값이 저장
  - 초기값이 없는 변수를 사용하면 오류
  - 변수를 선언한 이후에는 반드시 값을 저장

변수 초기화

```
int math = 99, korean = 90, science = 94;
```

```
int math = 99;
int korean = 90;
int science = 94;
```



대입에서 l-value와 r-value

- 대입연산자 =의 왼쪽에 위치하는 변수를 lvalue 또는 l-value라 하며
  - l-value는 반드시 수정이 가능한 하나의 변수이어야 함
  - r-value는 l-value에 저장할 자료값을 반환하는 표현식
  - 21 = 20 + 1과 문장은 오류가 발생

```
21 = 20 + 1;
```

(int)21

식이 수정할 수 있는 lvalue여야 합니다.

초기화 되지 않은 지역변수의 저장값과 오류

- 함수 내부에서 선언된 변수를 지역 변수local variables
- 초기화 되지 않은 지역 변수는 그 저장 값이 정의되지 않음
  - 소위 쓰레기값이라고 부르는 의미 없는 값이 저장
- 초기화 되지 않은 지역 변수를 다른 문장에서 사용하면
  - C4700 컴파일 오류가 발생

```
int math = 99;
int korean = 90;
int science;

int total = math + korean + science;
```

```
error C4700: 초기화되지 않은 'science' 지역 변수를 사용했습니다.
```

Source Code #04: subtraction.c

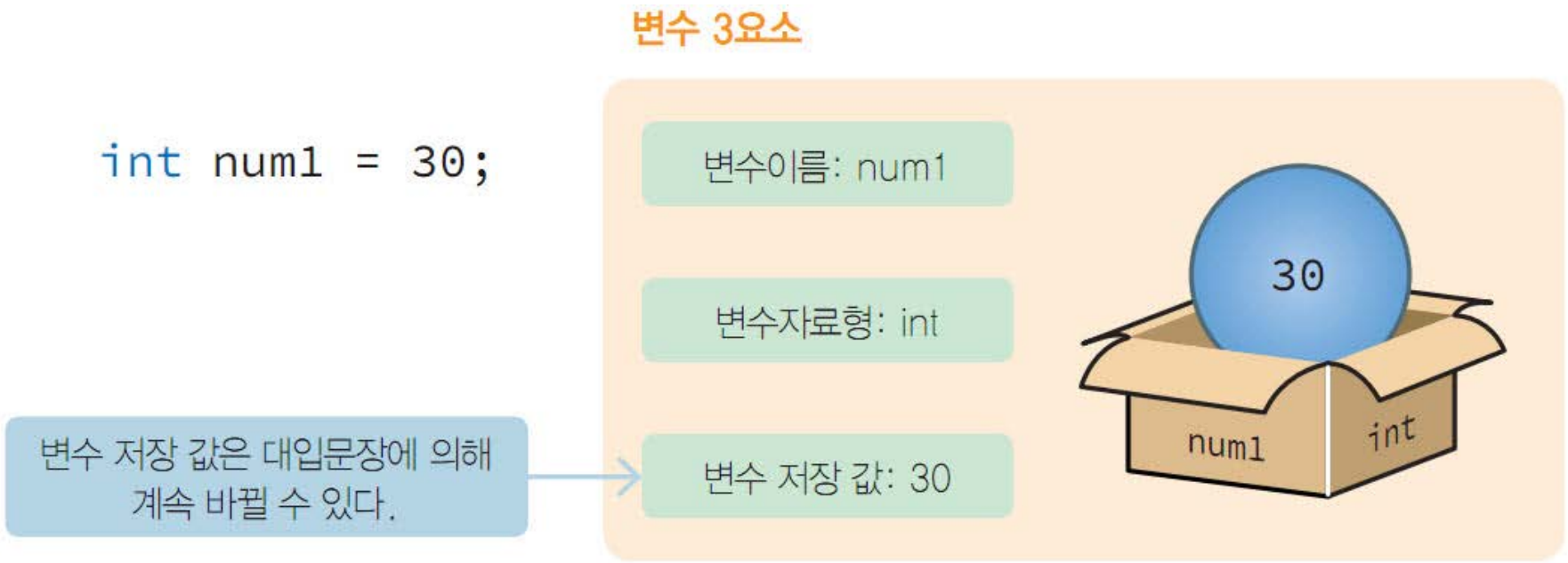
```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project4 / subtraction.c
    변수의 저장공간 자체와 변수에 저장된 값을 의미
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    int num1 = 30, num2 = 14;
    int difference;

    //대입 연산자의 왼쪽과 오른쪽에서의 변수의 의미 해석
    difference = num1 - num2;

    printf("num1: %d, num2: %d\n", num1, num2);
    printf("num1 - num2 의 결과: %d\n", difference);

    return 0;
}
```

```
num1: 30, num2: 14
num1 - num2 의 결과: 16
```

- 변수 num1, num2 사용
- 변수 difference에 차를 저장

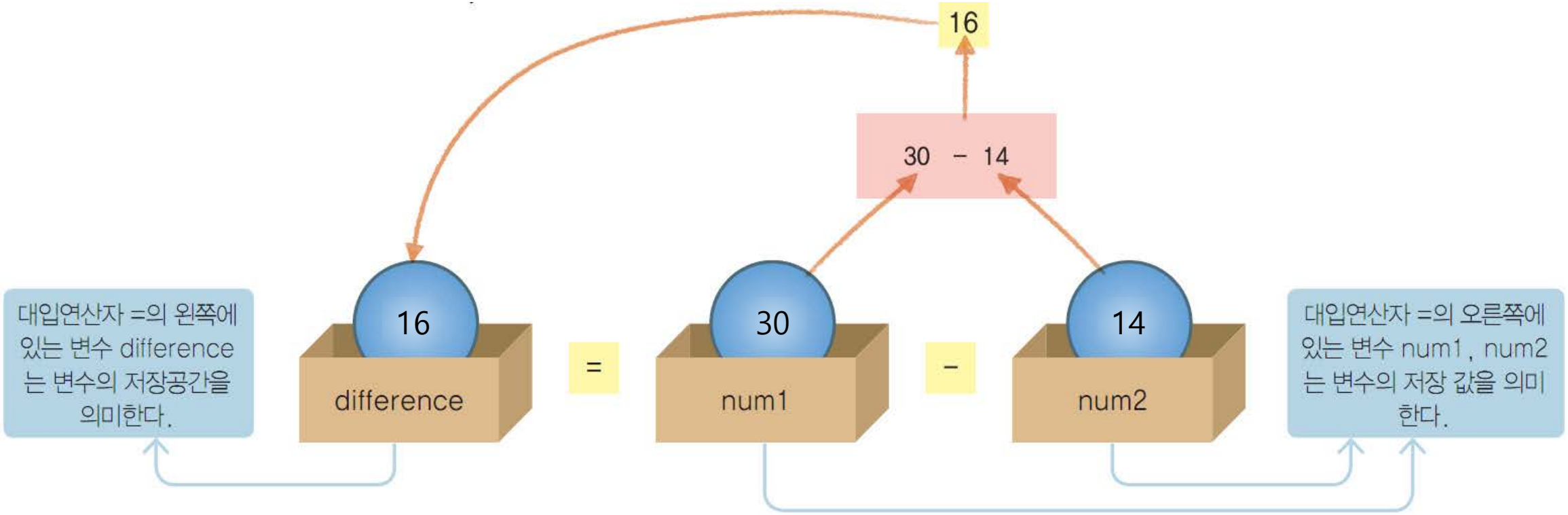
변수의 3요소

- 변수이름, 변수의 자료형, 변수 저장값



=의 왼쪽과 오른쪽

- 변수의 의미는 저장공간 자체와 저장공간에 저장된 값으로 나뉨
  - 대입 연산자 =의 왼쪽에 위치한 변수는 저장공간 자체의 사용을 의미
  - 대입 연산자 =의 오른쪽에 위치한 변수는 저장 값의 사용을 의미

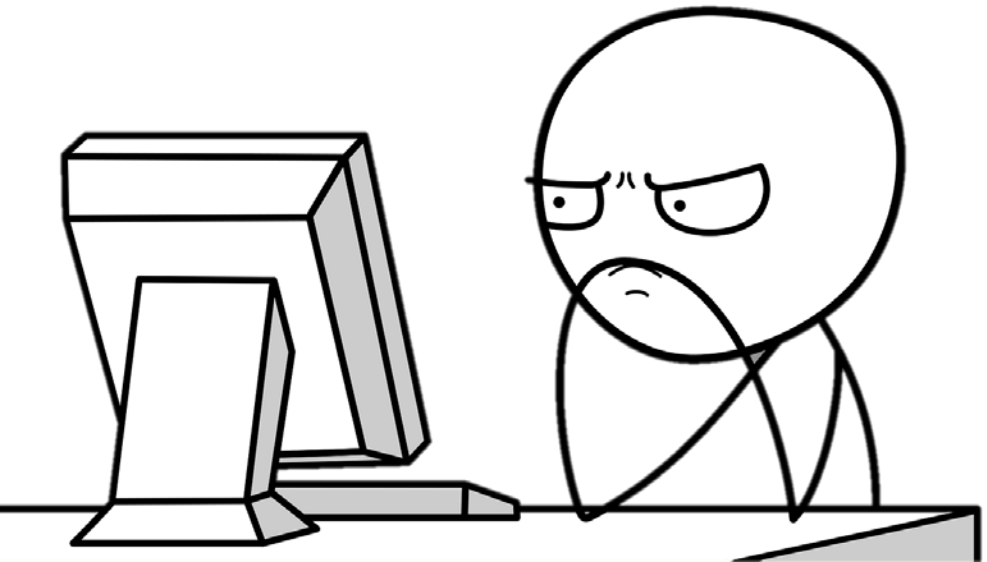


Lab #01: 두 정수의 합, 두 부동소수의 차 출력

- 두 정수의 합과 두 실수의 차가 출력되는 프로그램
  - 정수를 위한 자료형은 int로, 실수를 위한 자료형은 double로 이용
  - 합을 위한 연산자 +, 두 실수의 차를 위한 연산자 –와 결과 저장을 위한 변수 difference
- 결과
  - 합: 73
  - 차: -7.003000

```c
// basictype.c: 두 정수의 합, 두 실수의 차 출력

#include <stdio.h>

int main(void)
{
    int a = 30, b = 43; //두 정수 선언과 초기 값 대입
    int sum;            //두 정수의 합을 저장할 변수 선언
    ▨▨▨▨▨▨▨▨;          //두 정수의 합 구하기

    double x = 38.342, y = 45.345;  //두 실수 선언과 초기 값 대입
    double difference;              //두 실수의 차을 저장할 변수 선언
    ▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨;              //두 실수의 차 구하기

    printf("합: %d\n",▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨);   //두 정수의 합 출력
    printf("차: %f\n",▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨);   //두 실수의 차 출력

    return 0;
}
```

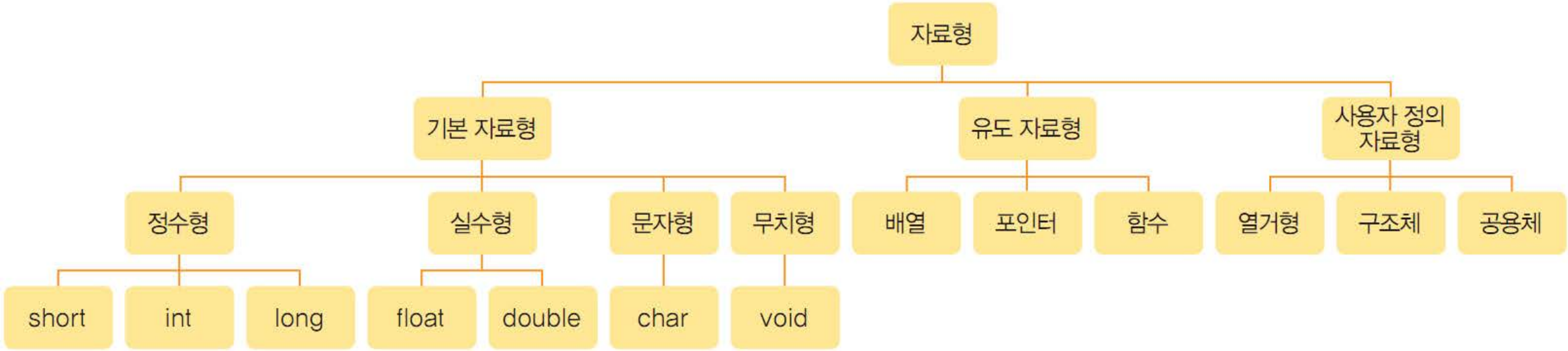
3. 기본 자료형



자료형

- C의 자료형
  - 기본형basic data types, 유도형derived data types, 사용자 정의형user defined data types
- 기본이 되는 자료형
  - 다시 정수형, 부동소수형, 문자형, 무치형
  - 무치형 자료형: void
    - 아무런 자료형도 지정하지 않은 자료형
    - 함수의 인자 위치에 놓이면 '인자가 없다'라는 의미로 사용
    - 함수의 반환값에 놓으면 '반환값이 없다'라는 의미
- 유도형
  - 배열array, 포인터pointer, 함수function 등으로 구성
- 사용자정의형
  - 기본형과 유도형을 이용하여 프로그래머가 다시 만드는 자료형
  - 열거형enumeration
  - 구조체structure
  - 공용체union

자료형


자료형
기본 자료형
유도 자료형
사용자 정의 자료형
정수형
실수형
문자형
무치형
배열
포인터
함수
열거형
구조체
공용체
short
int
long
float
double
char
void

정수형 int

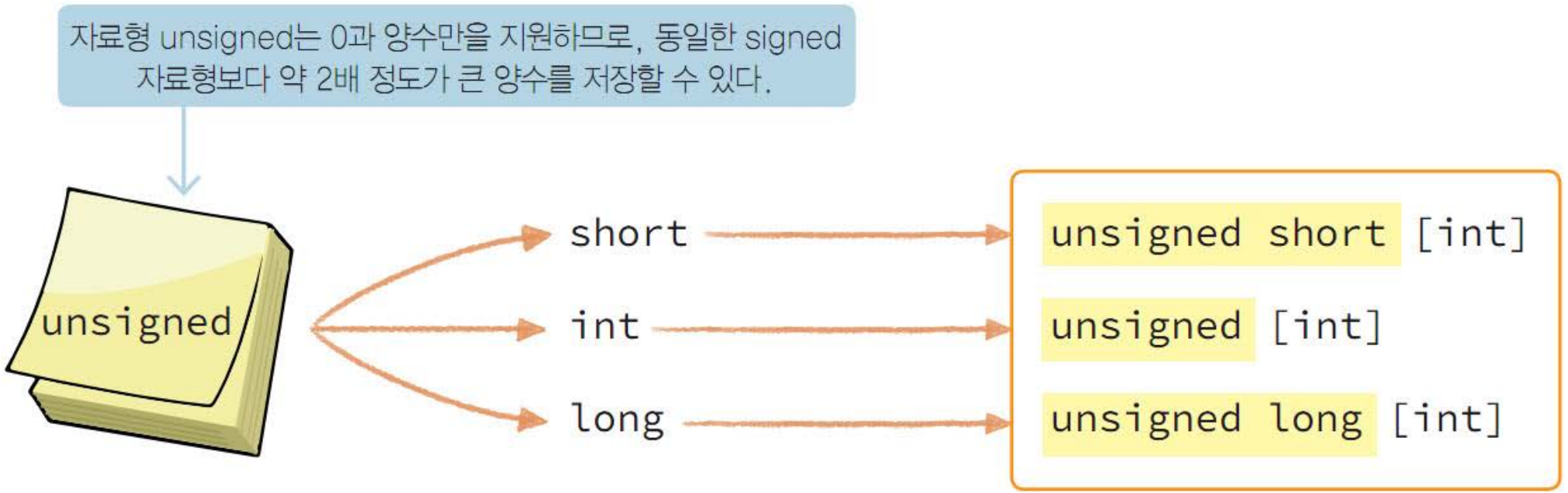
- 정수형(integer types)의 기본 키워드: int
  - 십진수, 팔진수, 십육진수의 정수가 다양하게 저장
  - 파생된 자료형: short와 long
  - short, short int
    - int보다 작거나 같고
    - short에 너무 큰 값을 저장한다면 아예 저장이 되지 않으며
  - long, long int
    - int보다 크거나 같고 long에 너무 작은 값을 저장한다면 그 만큼 자원의 낭비
    - 정수형 short, int, long 모두 양수, 0, 음수를 모두 표현

[부호가 있는] signed 키워드

- 정수형 자료형 키워드 앞에 표시 가능
- signed 키워드는 생략 가능
- signed int와 int는 같은 자료형

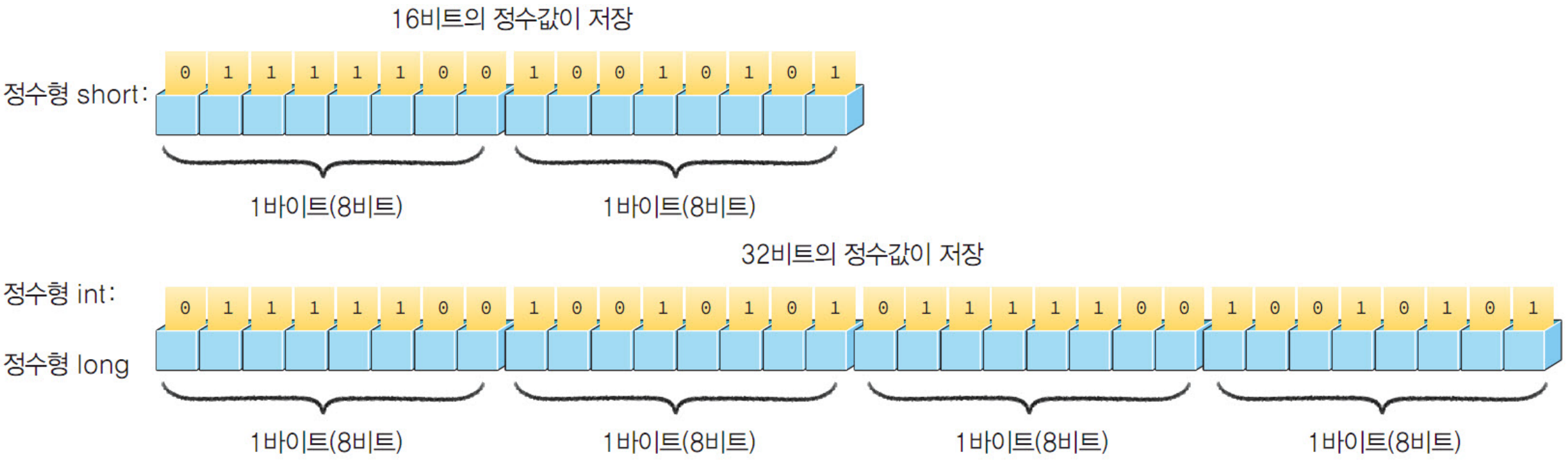
키워드 unsigned

- 0과 양수만을 처리
- short, int, long 앞에 표시
- unsigned int, int는 동일



정수형 저장공간

- 비주얼 스튜디오
  - short는 2바이트
  - int, long은 모두 4바이트
  - int는 short보다 표현 범위가 넓으며 long과는 동일



정수형 저장공간

- 저장공간 크기 n비트인 singed
  - -2n-1에서 2n-1-1까지 유효
- 저장공간 크기 n비트인 unsinged
  - 0에서 2n-1까지 유효

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	$0 \sim 4,294,967,295(2^{32}-1)$

Source Code #05: integer.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project05 / integer.c
    정수형 자료형 변수의 선언과 활용
    V 1.0 2018.
*/

#include <stdio.h>

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

    unsigned short int  usVar = 65000;      //0에서 65535까지 저장 가능
    unsigned int        uiVar = 4280000000; //약 0에서 42억 정도까지 저장 가능

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

    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
저장 값: 65000 4280000000
지구와 천왕성 간의 거리(km): 2720000000000
태양과 해왕성 간의 거리(km): 4500000000000
```

- 21억보다 큰 정수의 사용

C99 표준

- 1999년에 제정한 C99 표준에 따르면 정수형을 다음과 같이 5개로 구분

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)

자료형 long long int

- 간단히 long long으로 사용할 수 있으며
- 약 922경 정도의 수를 음수와 양수로 지원
- unsigned long long은 0에서 약 1,844경까지 지원

행성	태양에서 행성까지의 실제 거리
수성	5,800 km
금성	1억 km
지구	1억 5천만 km
화성	2억 3천만 km
목성	7억 8천만 km
토성	14억 3천만 km
천왕성	28억 7천만 km
해왕성	45억 km
명왕성	60억 km
엘로힘행성	지구에서 9조 km

Source Code #06: float.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch02 / Project06 / float.c
    부동소수형 변수의 선언과 활용
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    float       x = 3.14F;      //float x = 3.14;인 경우, 경고 발생
    double      y = -3.141592;  //double 저장공간 크기는 float의 2배
    long double z = 180000000.0;//double과 long double은 저장공간이 모두 64비트

    printf("저장 값: %f %f %f\n", x, y, z);

    return 0;
}
```

```
저장 값: 3.140000 -3.141592 180000000.000000
```

- 실수를 위한 저장 공간 사용

부동소수형 3가지

- 키워드는 float, double, long double 세 가지
- 비주얼 스튜디오
  - float는 4바이트이며, double과 long double은 모두 8바이트

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

Source Code #07: char.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project07 / char.c
    문자형 변수의 선언과 이용
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    char c1 = 'a';      //소문자 a
    char c2 = 65;       //대문자 A가 코드 값 65
    char c3 = '\132';   //대문자 Z의 8진수 코드 값 132
    char c4 = '\x5A';   //대문자 Z의 16진수 코드 값 5A

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

    return 0;
}
```

```
저장 값(문자): a A Z Z
저장 값(정수): 97 65 90 90
```

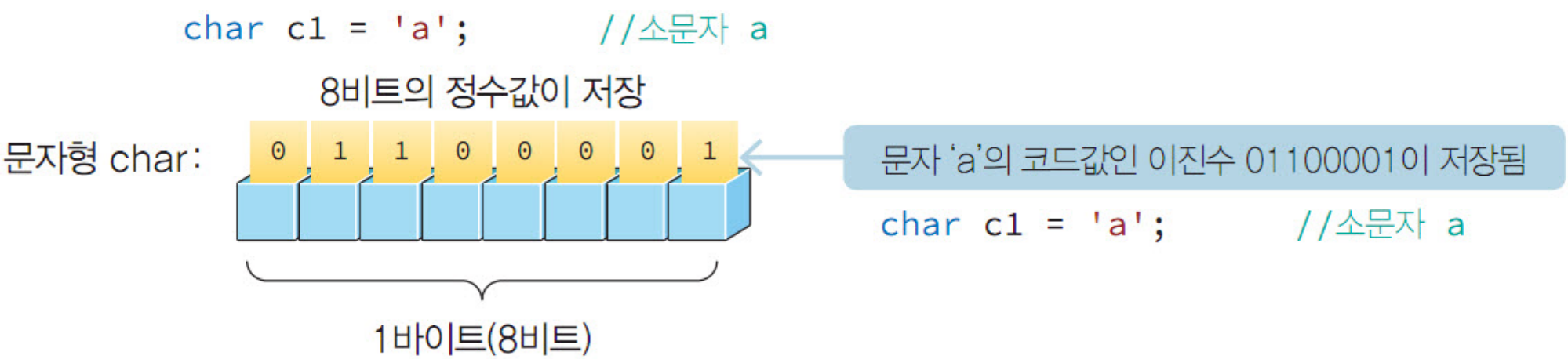
- 문자 저장 공간 사용

문자형 char

- char, signed char, unsigned char 세 가지 종류
  - 문자형 저장공간 크기는 모두 1바이트
  - 키워드 signed와 unsigned를 함께 이용 가능
- 비주얼 스튜디오
  - char는 signed char와 같으나, 컴파일러에 따라 다를 수 있음

문자형 자료형 저장 방법

- 'a'와 같이 문자 상수를 이용하거나, 정수를 직접 저장
- 문자 코드값 저장
  - '\ddd'와 같이 세 자리의 팔진수로,
  - '\xhh'와 같이 두 자리의 십육진수로 표현



아스키 코드

- C 언어에서 문자형 자료공간에 저장되는 값
  - 실제로 정수값이며
  - 아스키 코드 표에 의한 값
- 아스키 코드
  - ASCII: American Standard Code for Information
  - ANSI(American National Standards Institute)에서 제정한 정보 교환용 표준 코드
  - 총 127 개의 문자로 구성
  - 소문자 'a'
    - 16진수로 61
    - 이진수로는 1100001
    - 십진수로 97

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

Source Code #08: size.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch02 / Project08 / size.c
    연산자 sizeof를 이용한 저장공간 크기 출력
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    printf("     자료형 : 크기(바이트)\n");
    printf("       char : %d %d\n", sizeof(char), sizeof(unsigned char));
    printf("      short : %d %d\n", sizeof(short), sizeof(unsigned short));
    printf("        int : %d %d\n", sizeof(int), sizeof(200));
    printf("       long : %d %d\n", sizeof(long), sizeof(300L));
    printf("  long long : %d %d\n", sizeof(long long), sizeof(900LL));
    printf("      float : %d %d\n", sizeof(float), sizeof 3.14F);
    printf("     double : %d %d\n", sizeof(double), sizeof 3.14);
    printf("long double : %d %d\n", sizeof(long double), sizeof 3.24L);

    return 0;
}
```

```
     자료형 : 크기(바이트)
       char : 1 1
      short : 2 2
        int : 4 4
       long : 4 4
  long long : 8 8
      float : 4 4
     double : 8 8
long double : 8 8
```

- 연산자 sizeof 사용

문자형 char

- 기본 자료형은 long long을 포함하면 모두 14가지

분류	자료형	크기	표현범위
문자형	char	1 바이트	−128(−27) ~ 127(27−1)
	signed char	1 바이트	−128(−27) ~ 127(27−1)
	unsigned char	1 바이트	0 ~ 255(28−1)
정수형	[signed] short [int]	2 바이트	−32,768(−215) ~ 32,767(215−1)
	[signed] [int]	4 바이트	−2,147,483,648(−231) ~ 2,147,483,647(231−1)
	[signed] long [int]	4 바이트	−2,147,483,648(−231) ~ 2,147,483,647(231−1)
	[signed] long long [int]	8 바이트	9,223,372,036,854,775,808(−263) ~ 9,223,372,036,854,775,807(263−1)
정수형	unsigned short [int]	2 바이트	0 ~ 65,535(216−1)
	unsigned [int]	4 바이트	0 ~ 4,294,967,295(232−1)
	unsigned long [int]	4 바이트	0 ~ 4,294,967,295(232−1)
	[unsigned] long long [int]	8 바이트	0 ~ 18,446,744,073,709,551,615(264−1)
부동소수형	float	4 바이트	대략 10^{-38} ~ 10^{38}
	double	8 바이트	대략 10^{-308} ~ 10^{308}
	long double	8 바이트	대략 10^{-308} ~ 10^{308}

연산자 sizeof

- 자료형, 변수, 상수의 저장공간 크기를 바이트 단위 반환
- 자료형 키워드로 직접 저장공간 크기를 알려면 자료형 키워드에 괄호가 반드시 필요

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

Source Code #09: overflow.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project09 / overflow.c
    오버플로와 언더플로의 발생
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    unsigned char   uc = 255 + 1;
    short           s = 32767 + 1;
    float           min = 1.175E-50;
    float           max = 3.403E39;

    printf("%u\n", uc);     //오버플로 발생
    printf("%d\n", s);      //오버플로 발생
    printf("%e\n", min);    //언더플로 발생
    printf("%f\n", max);    //오버플로 발생

    return 0;
}
```

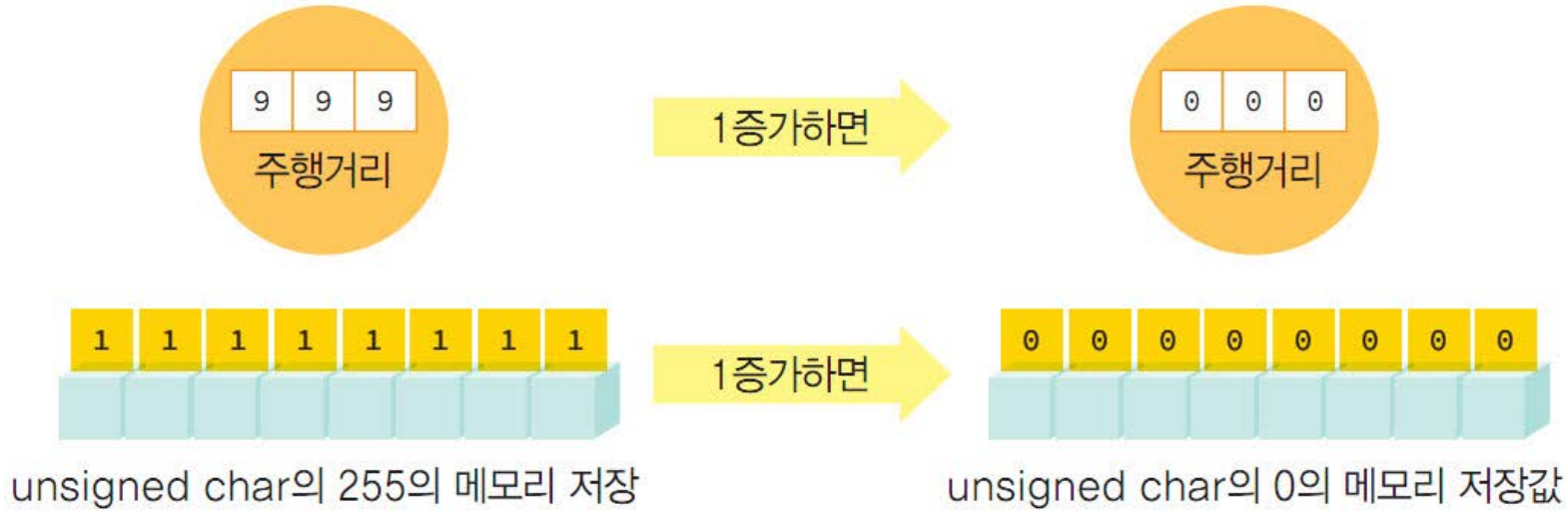
```
0
-32768
0.000000e+00
inf
```

- 자료형의 범주에서 벗어난 값을 저장
- 오버플로overflow / 언더플로underflow가 발생

```
Woverflow.c(11): warning C4305: '초기화 중': 'int'에서 'unsigned char'(으)로 잘립니다.
Woverflow.c(13): warning C4305: '초기화 중': 'double'에서 'float'(으)로 잘립니다.
Woverflow.c(14): warning C4056: 부동 소수점 상수 산술 연산에서 오버플로가 발생했습니다.
Woverflow.c(14): warning C4756: 상수 산술 연산에서 오버플로가 발생했습니다.
```

오버플로와 언더플로

- 오버플로
  - 자료형 unsigned char
    - 8비트로 0에서 255까지 저장 가능
    - 만일 256을 저장하면 0으로 저장
  - 정수의 순환
    - 정수형 자료형에서 최대값+1은 오버플로로 인해 최소값이 저장
    - 마찬가지로 최소값-1은 최대값
- 언더플로
  - 실수형 float 변수에 정밀도가 매우 자세한 수를 저장하면 언더플로underflow가 발생
  - 0이 저장



Lab #02: 아스키 코드값 126 문자 '~'의 다양한 출력

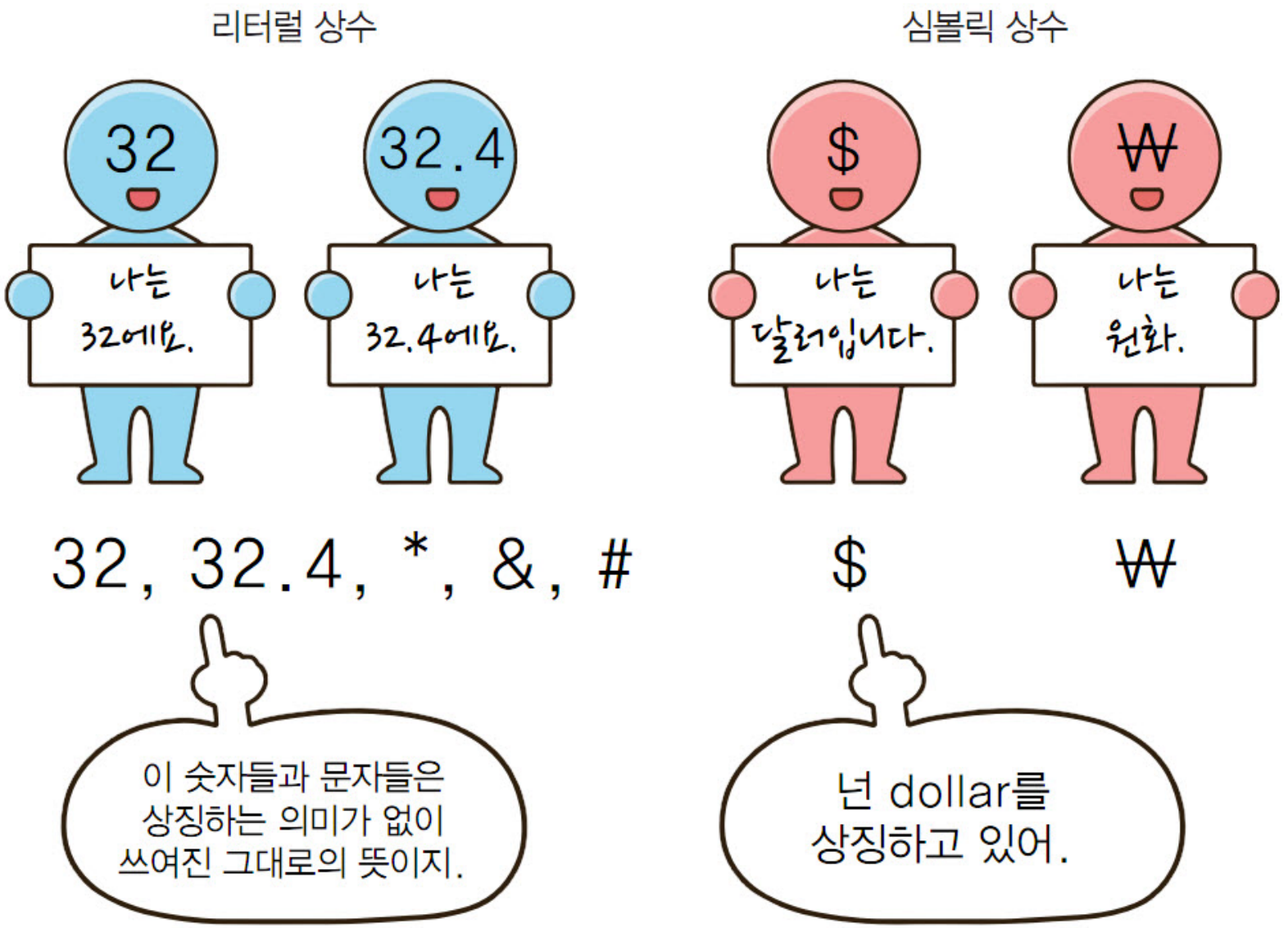
- 다음 결과로 출력되는 프로그램을 작성
  - 문자 '~'의 코드값
    - 십진수 126
    - 팔진수 176
    - 십육진수 72
  - 출력을 위한 함수 print()에서 %d로 정수를, %c로 문자를 출력

```c
// intchar.c: 아스키 토드 값 126 문자 '~'의 다양한 출력

#include <stdio.h>

int main(void)
{
    int ch = 126;

    printf("%d\n", ch); //십진 코드 값 출력
    printf("%o\n", ch); //문자 출력
    printf("%c\n", '\176'); //문자 출력
    printf("%c\n", '\x7e'); //문자 출력

    return 0;
}
```

상수의 종류와 표현 방법

- 상수constant
  - 이름 없이 있는 그대로 표현한 자료값
  - 우린 생활에서 숫자 32, 32.4, 문자 *, &, # 그리고 문자열 "Hello World!"등을 사용
  - 이름이 있으나 정해진 하나의 값만으로 사용되는 자료값
- 리터럴 상수
  - 소스에 그대로 표현해 의미가 전달되는 다양한 자료값
  - 10, 24.3과 같은 수, "C는 흥미롭습니다."와 같은 문자열
- 심볼릭 상수
  - 변수처럼 이름을 갖는 상수
  - 심볼릭 상수를 표현하는 방법, 세 가지
  - const 상수const constant
  - 매크로 상수macro constant
  - 열거형 상수enumeration constant

상수의 종류와 표현 방법


리터럴 상수
심볼릭 상수
32
32.4
$
₩
나는 32에요.
나는 32.4에요.
나는 달러입니다.
나는 원화.
32, 32.4, *, &, #
$
₩
이 숫자들과 문자들은 상징하는 의미가 없이 쓰여진 그대로의 뜻이지.
넌 dollar를 상징하고 있어.

상수의 종류와 표현 방법

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};

정수와 실수, 문자와 문자열

- 리터럴 상수
  - 정수, 실수
  - 문자, 문자열 상수
- 문자 상수 표현
  - 문자 하나의 앞 뒤에 작은따옴표single quote를 넣어 표현
  - \ddd
    - 팔진수 코드값을 이용
  - \xhh
    - 십육진수 코드값을 이용
  - 코드값이 97인 문자 'A'
    - '\141'와 '\x61'로 표현
- 함수 printf()
  - 문자 상수를 출력하려면 %c 또는 %C 사용
  - %c의 c는 문자 character를 의미

정수와 실수, 문자와 문자열

하나 하나의 문자가 모여 문자열이 구성되며, |도 문자이며 프로그램에서는 'I'와 같이 작은따옴표로 둘러싸야 한다.

일상생활에서는 문장을 표현하는 문자열을 그대로 사용하나 프로그램에서는 "문자열 문장"과 같이 큰 따옴표를 둘러싸야 한다.

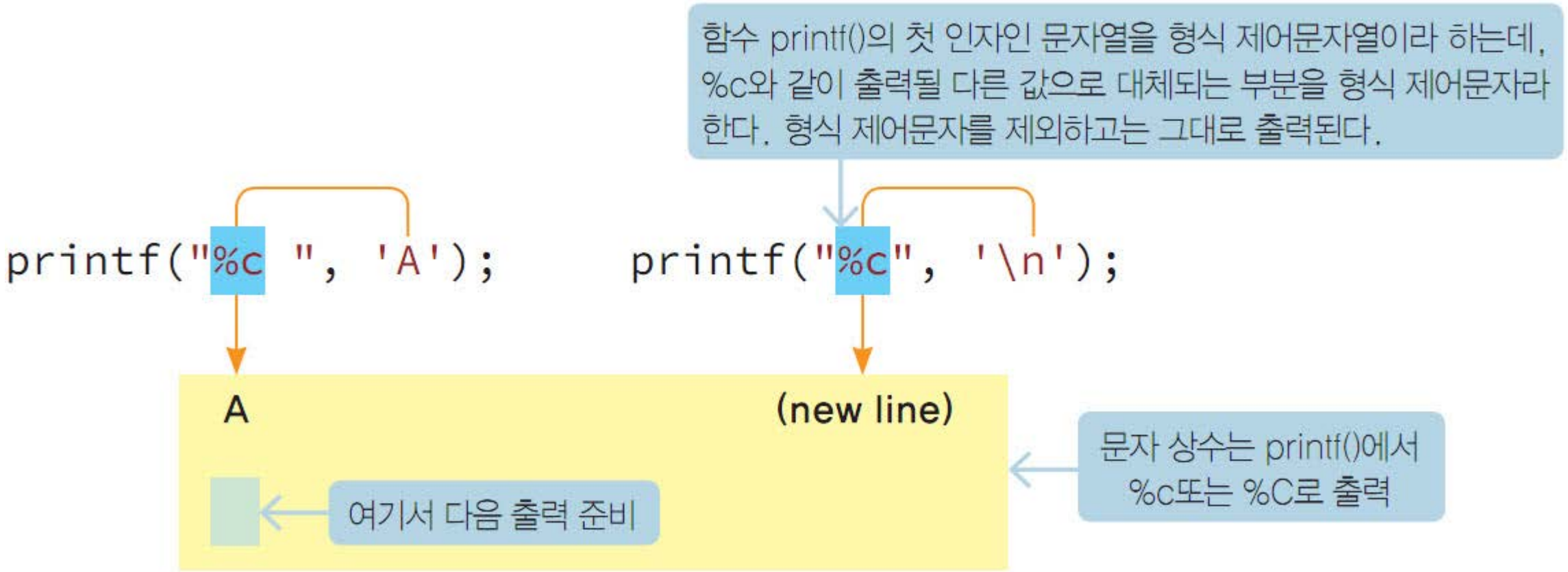
정수 상수: 정수를 그대로 표현 가능하다.
- 십진수: 4, 5, 20
- 8진수: 01, 013
- 16진수: 0X1f, 0x1d

실수 상수: 실수를 그대로 표현 가능하다.
- 소수점 실수: 2.5, 3.1 3.0
- 지수표현 방식: 2.47E3, 12.82e-3

문자 상수: 한 문자를 작은따옴표로 묶는 표현
- 일반문자: 'b', 'C'
- 특수문자: '\n', '\'

문자열 상수: 일련의 여러 문자를 큰따옴표로 표현
- "C++", "Objective-C"

정수와 실수, 문자와 문자열



Source Code #10: charliteral.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project10 / charliteral.c
    이스케이프 문자를 비롯해서 다양한 문자 리터럴의 표현
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    printf("%Casic", 'B');      printf("%c", '\n');

    char sq = '\'';                 //작은따옴표

    printf("BCPL\tB\tC\tJava\n");   //문자열 내부에서 \t(탭) 문자 사용
    printf("%c\7\n", '\a');         //알람 문자를 2번 출력하고 공백 줄
    printf("%c자바언어'\n", sq);    //문자열 내부에서는 '(작은따옴표) 그대로 사용 가능

    //문자열 내부에서는"(큰따옴표) 반드시 \"로 사용
    printf("\"C언어\" 정말 재미있다!\n");

    return 0;
}
```

```
Basic
BCPL    B       C       Java

'자바언어'
"C언어" 정말 재미있다!
```

- 문자 리터럴의 표현과 출력

이스케이프 문자

- \n와 같이 역슬래쉬 \와 문자의 조합으로 표현하는 문자
  - '\n'이 새로운 줄(new line)을 의미하는 대표적인 이스케이프 문자
  - 문자열에도 사용 가능
- 이스케이프 문자는 제어문자, 특수문자 또는 확장문자라고도 부름

이스케이프 문자

제어문자 이름	영문 표현	코드값 (십진수)	\ddd (팔진수)	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

정수형 리터럴 상수의 다양한 형태 100L, 20U, 5000UL

- 정수 뒤에 l 또는 L을 붙이면 long int
- u 또는 U는 unsigned int
- ul 또는 UL은 unsigned long
- long long형은 LL, ll과 ULL, ull

unsigned int형 상수
30u, 6754U, 34566549u

int형 상수
-3, 40, +8000, -1234

unsigned long형 상수
300000UL, 23456ul, 7834ul

long long형 상수
367853345643LL, -2345LL

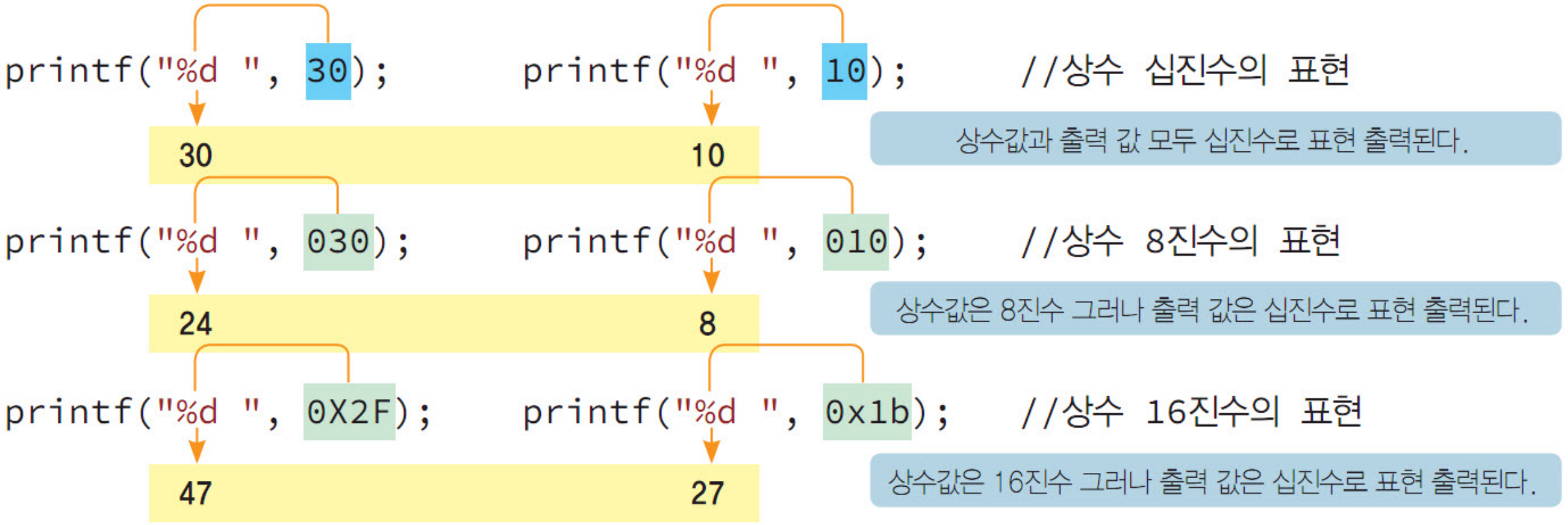
unsigned long long형 상수
4200000ULL, 232356456ull,834ul

long long형 상수
367853345643LL, -2345LL

2진수와 16진수 표현방식

- 상수의 정수표현은 10진수로 인식
- 숫자 0을 정수 앞에 놓으면 8진수octal number로 인식
- 숫자 0과 알파벳으로 0x, 0X를 숫자 앞, 16진수hexadecimal number로 인식
  - 십육진수는 0에서 9까지의 수와 알파벳 a, b, c, d, e, f(대소문자 모두 가능)
- 함수 printf()에서 정수를 출력
  - %d의 사용
  - d는 십진수라는 decimal

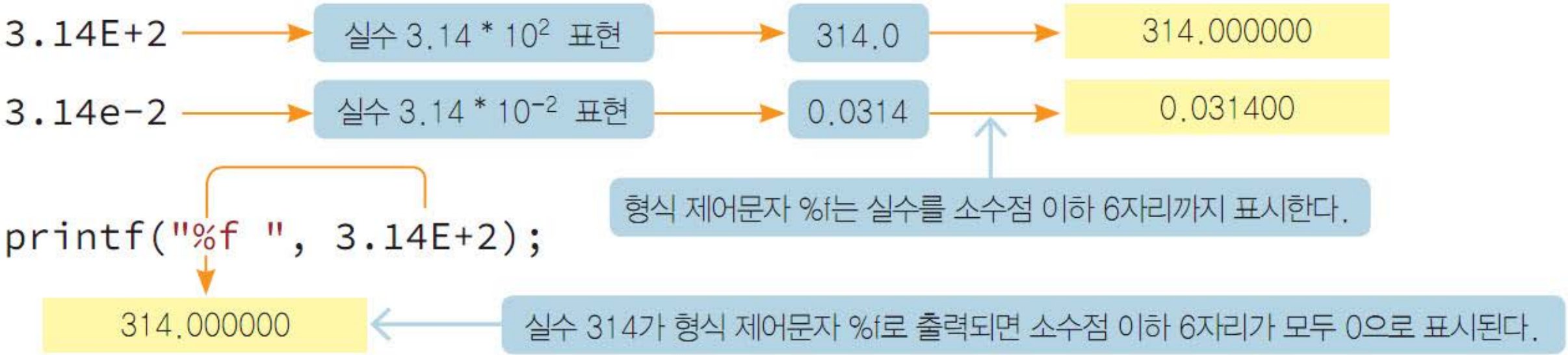
8진수와 16진수의 상수 표현과 출력



실수 리터럴 상수

- 지수표현 방식: 3.14E+2는 3.14*102
- 함수 printf()에서 지수표현 방식과 함께 일반 실수를 출력
  - %f의 형식 제어문자를 사용, f는 실수를 의미하는 float에서 나온 f
  - 형식 제어문자 %f로 출력되는 실수는 소수점 6자리까지 출력
- 실수형 상수: float, double, long double
- 소수는 double 유형
- float 상수: 숫자 뒤에 f나 F를 붙임
- long double 상수: 숫자 뒤에L 또는 l을 붙여 표시

실수 리터럴 상수


3.14E+2
실수 3.14 * 10^2 표현
314.0
314.000000
3.14e-2
실수 3.14 * 10^-2 표현
0.0314
0.031400
형식 제어문자 %f는 실수를 소수점 이하 6자리까지 표시한다.
printf("%f ", 3.14E+2);
314.000000
실수 314가 형식 제어문자 %f로 출력되면 소수점 이하 6자리가 모두 0으로 표시된다.

실수 리터럴 상수

float형 상수
3.4F, 3.45E3f, 3.0F, 46.7e-2F

3은 정수형 상수이나 3.0은 double형 상수이다.

```
float var = 3.14F;
```

double형 상수
3.4, 3.45E3, 3.0, 46.7e-2

long double형 상수
3.4L, 3.45E3L, 3.0L, 46.7e-2L

심볼릭 const 상수

- 키워드 const
  - 변수로는 선언되지만 일반 변수와는 달리 초기값을 수정할 수 없으며
  - 이름이 있는 심볼릭 상수constant number
  - 상수는 변수선언 시 반드시 초기값을 저장
  - 상수는 다른 변수와 구별하기 위해 관례적으로 모두 대문자로 선언

```
//키워드 const로 상수 만들기

const double RATE = 0.03;    //연이자율 3%
int deposit = 800000;

RATE = 0.032;    //수정이 불가능
```

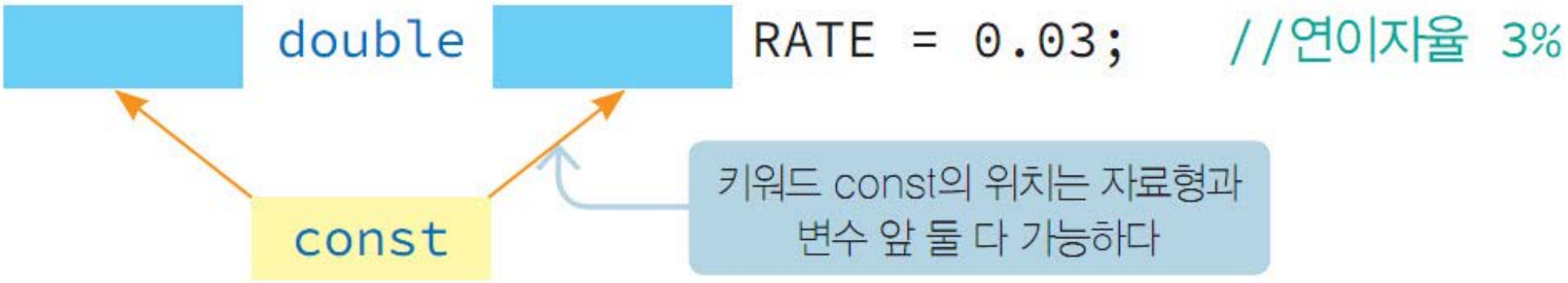
대입 문장의 왼쪽은 수정 가능한 변수이어야 하나 수정이 불가능한 심볼릭 상수이므로 오류가 발생한다

```
error C2166: l-value가 const 개체를 지정합니다.
```

```
const double RATE = (0.029999999999999999)
연이자율 3%
오류: 식이 수정할 수 있는 lvalue여야 합니다.
```

심볼릭 const 상수

- 변수 RATE
  - 상수로 선언하는 구문
  - 선언 이후 저장값을 수정
    - 대입 문장에서 컴파일 오류 C2166이 발생
    - 이자율을 3%에서 3.2%로 수정하려면 const가 있는 선언문에서 직접 0.03을 0.032로 수정



Source Code #11: const.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project11 / const.c
    키워드 const를 사용한 상수 선언
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    //키워드 const로 상수 만들기
    const double RATE = 0.03;   //연이자율 3%
    int deposit = 800000;

    //RATE = 0.032; //수정이 불가능
    printf("이자율: %f\n", RATE);
    printf("계좌 잔고: %d\n", deposit);
    printf("이자액: %f\n", deposit * RATE);

    //문자열을 변수에 저장
    char* str = "좋은 C 언어 입문서"; //char *str, char * str 모두 가능
    char* const title = "진보된 C 언어"; //title에 다른 문자열 상수 저장이 불가

    str = "최근 가장 좋은 C 언어 입문서";
    //title = "C 언어 스케치"; //수정 불가능

    printf("\n%s: %s\n", str, title); //문자열 변수 출력

    return 0;
}
```

- 키워드 const 사용한 심볼릭 상수 이용

문자열 리터럴

- char* 변수에 저장
  - *는 포인터(pointer)라는 의미의 문자
  - 변수 str에는 대입한 문자열에서 첫 문자의 주소(address)가 저장되는 변수
- 변수 title에서 다른 문자열로 대체할 수 없도록 상수로 만들려면
  - 반드시 title 앞에 const를 삽입
  - 만일 char* 앞에 const를 삽입하면 문법적으로 다른 의미

```
//문자열을 변수에 저장

char* str = "좋은 C 언어 입문서";      //char *str, char * str 모두 가능
char* const title = "진보된 C 언어"; //title에 다른 문자열 상수 저장이 불가
```

변수 title에 다른 문자열로 대체할 수 없도록 하려면
이 위치에 키워드 const를 삽입해야 한다.

Source Code #12: enum.c

```c
/*
    솔루션 / 프로젝트 / 소스파일: Ch03 / Project12 / enum.c
    키워드 enum으로 만드는 열거형 정수 상수 목록
    V 1.0 2018.
*/

#include <stdio.h>

int main(void)
{
    //키워드 enum으로 열거형 정수 상수 목록 만들기
    enum DAY { SUN, MON, TUE, WED, THU, FRI, SAT };
    printf("일요일 상수: %d\n", SUN); //0
    printf("수요일 상수: %d\n", WED); //3

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

    return 0;
}
```

- Enum을 사용한 정수 상수의 이용

열거형 상수의 이용

- 열거형 : 정수형 상수 목록 집합을 정의하는 자료형
  - 열거형 상수에서 목록 첫 상수의 기본값이 0
  - 다음부터 1씩 증가하는 방식으로 상수값이 자동으로 부여

```
//키워드 enum으로 열거형 정수상수 목록 만들기
enum DAY {SUN, MON, TUE, WED, THU, FRI, SAT};
          0    1    2    3    4    5    6

enum 열거형태그명 {열거형상수1, 열거형상수2, 열거형상수3, ... };
```

열거 상수 목록으로 순서대로 0, 1, 2 등으로 정의된다.

열거형 상수의 이용

- 상수 목록에서 특정한 정수 지정 가능
  - 상수값을 지정한 상수는 그 값으로, 따로 지정되지 않은 첫 번째 상수는 0이며, 중간 상수는 앞의 상수보다 1씩 증가한 상수값으로 지정

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

정수형 상수 목록

매크로 상수

- 전처리기 지시자 #define
  - 매크로 상수macro constant를 정의하는 지시자
  - 주로 대문자 이름으로 정의
  - 전처리기preprocessor: 매크로 상수를 모두 #define 지시자에서 정의된 문자열로 대체replace

```
#define KPOP 50000000          //정수 매크로 상수
#define PI 3.14                //실수 매크로 상수
```

매크로 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

매크로 상수

```
#define SHRT_MIN      (-32768)                    /* minimum (signed) short value */
#define SHRT_MAX      32767                       /* maximum (signed) short value */
#define USHRT_MAX     0xffff                      /* maximum unsigned short value */
#define INT_MIN       (-2147483647 - 1)           /* minimum (signed) int value */
#define INT_MAX       2147483647                  /* maximum (signed) int value */
#define UINT_MAX      0xffffffff                  /* maximum unsigned int value */
#define LONG_MIN      (-2147483647L - 1)          /* minimum (signed) long value */
#define LONG_MAX      2147483647L                 /* maximum (signed) long value */
#define ULONG_MAX     0xffffffffUL                /* maximum unsigned long value */
#define LLONG_MAX     9223372036854775807i64      /* maximum signed long long int value */
#define LLONG_MIN     (-9223372036854775807i64 - 1)  /* minimum signed long long int value */
#define ULLONG_MAX    0xffffffffffffffffui64      /* maximum unsigned long long int value */
```

Lab #03: 부동소수형 최대 최소 매크로 상수 출력

- 헤더파일 float.h에 정의된 최대 최소 상수를 출력하는 프로그램 작성
  - 자료형 float의 최대 최소 매크로 상수: FLT_MIN, FLT_MAX
    - 자료형 double의 최대 최소 매크로 상수: DBL_MIN, DBL_MAX
    - 위 상수를 참고로 자료형 long double의 최대 최소 매크로 상수 출력
  - 출력을 위한 함수 printf()에서 %e로 부동소수형 출력
- 결과
  - float 범위: 1.175494e-38 3.402823e+38
  - double 범위: 1.175494e-38 3.402823e+38
  - long double 범위: 2.225074e-308 1.797693e+308

```c
// minmaxfloat.c: 부동소수형 세가지의 최대 최소 상수 출력

#include <stdio.h>
#include <float.h>      //부동소수형 상수가 정의된 헤더파일 삽입

int main(void)
{
    printf("float 범위: %e %e\n", [redacted]);
    printf("double 범위: %e %e\n", [redacted]);
    printf("long double 범위: %e %e\n", [redacted]);

    return 0;
}
```


I
PROGRAMMING