제03장

자료형과 변수

Perfect C

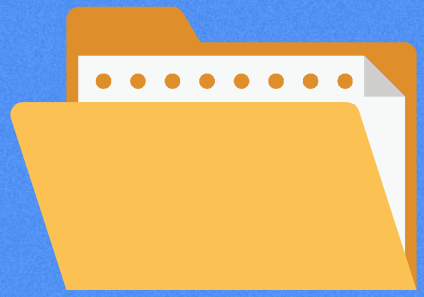
C언어로 배우는 프로그래밍 기초

단원 목표

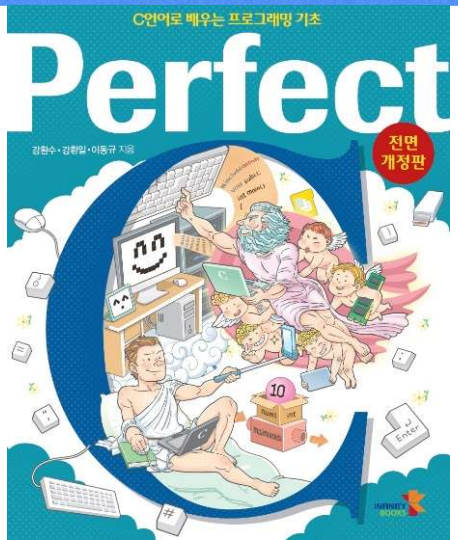
학습목표

- **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

내용 목차



01. 프로그래밍 기초



C 프로그램 구조와 프로그램 실행(1)

프로그램 구조

프로그램

- C 프로그램은 하나 이상의 여러 함수가 모여 한 프로그램으로 구성
- 비주얼 스튜디오
  - 솔루션은 여러 개의 프로젝트로 구성
  - 다시 프로젝트는 여러 소스파일을 포함한 여러 자원(resource)으로 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
- puts(...) 등 함수 호출
  - 호출된 함수로 이동하여 그 함수를 모두 실행한 후 다시 돌아와 그 이후의 문장을 실행

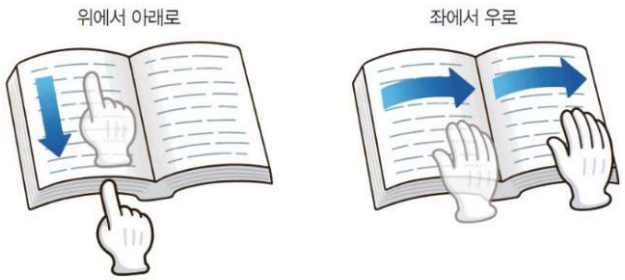


그림 3-3 소설을 읽듯이 C 프로그램의 문장을 실행

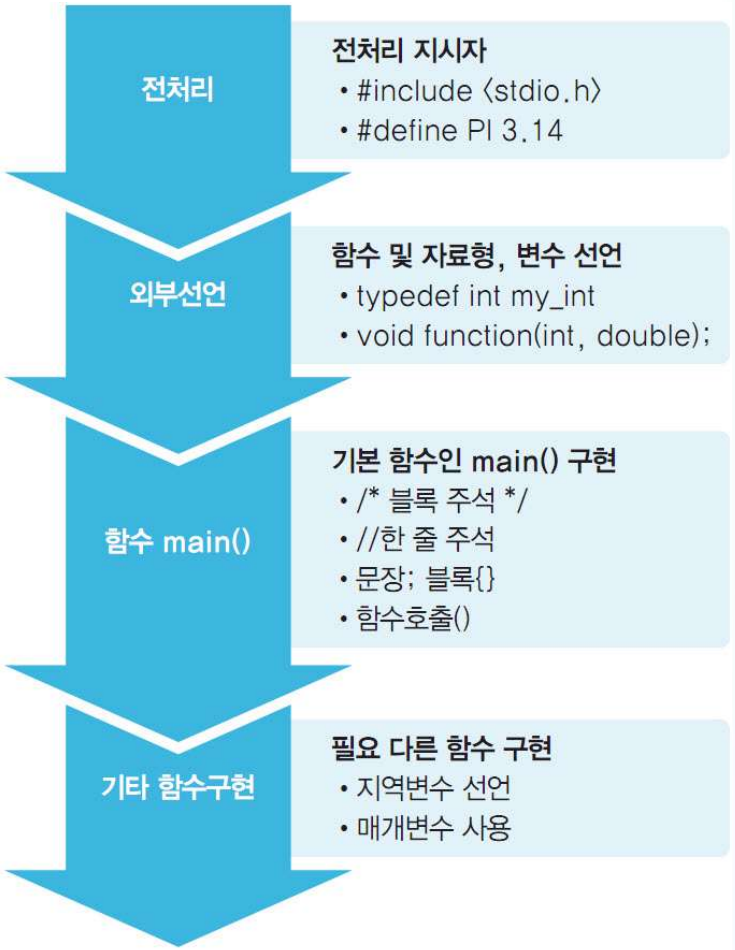


그림 3-1 C 프로그램 소스의 구조

C 프로그램 구조와 프로그램 실행(2)

프로그램 구조

함수 main()이 구현된 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

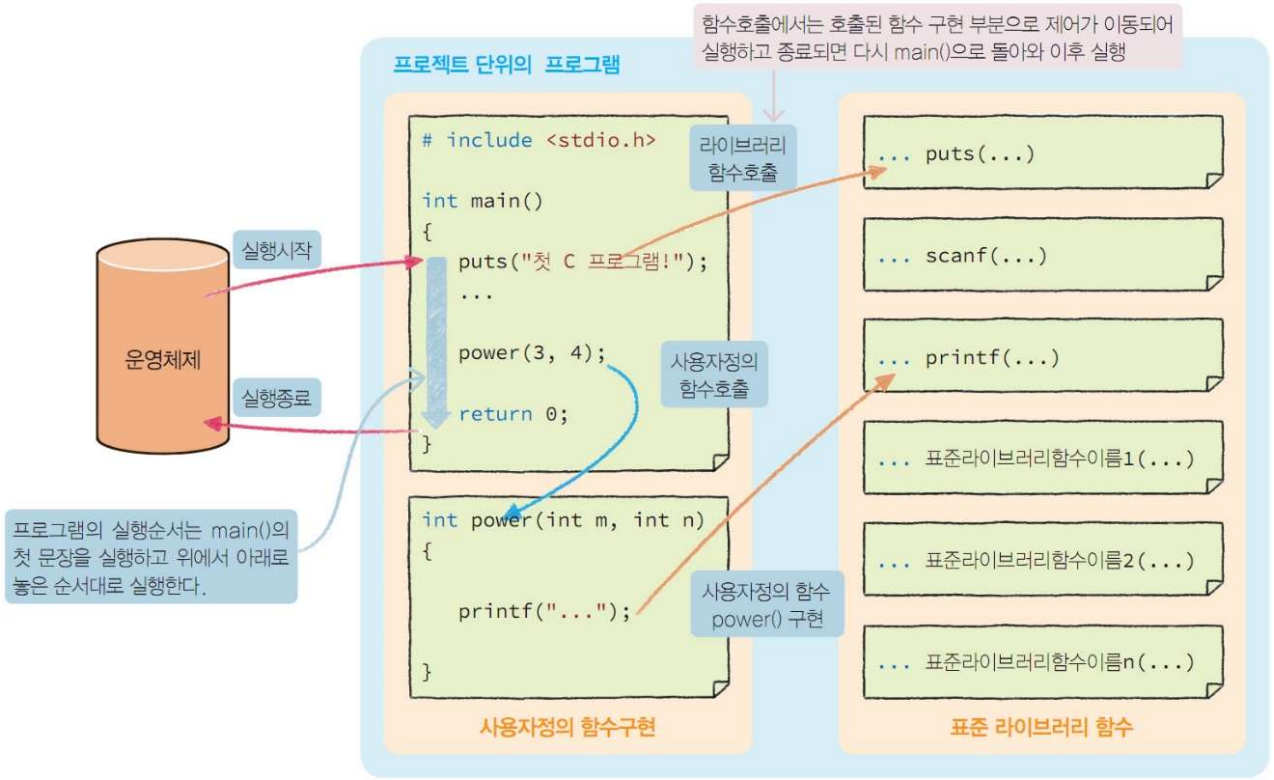


그림 3-2 C 프로그램의 구성과 실행

키워드

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

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(identifiers)

- **프로그래머가 자기 마음대로 정의해서 사용하는 단어**
  - 변수이름
    - age, year 등
  - 함수이름
    - puts, main, printf 등

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

그림 3-5 식별자 구성 규칙과 사용 예

- **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

문장과 블록, 들여쓰기

프로그램 구조

문장과 블록

- 컴퓨터에게 명령을 내리는 최소 단위를 문장(statement)
- 문장은 마지막에 세미콜론 ;으로 종료
  - 문장 마지막에 ;을 빠뜨리면
  - 컴파일 시간에 문법 오류가 발생
- 여러 개의 문장을 묶으면 블록(block)
- { 문장1, 문장2, ... } 처럼 중괄호로 열고 닫음

"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

그림 3-6 문장과 문법 오류

```
int main(void)
{
    puts("puts()는 한 줄에 문자열 출력함수"); //한 줄 출력을 자동으로
    ...
    printf("print()는 다양한 종류의");
    ...
    puts("자바");        puts("C#");
    return 0;
}
```

블록을 시작하는 중괄호 { 다음에는 탭 정도만큼 오른쪽으로 들여쓰고, 블록 종료 표시인 중괄호 } 는 다시 원 위치에 작성한다.

프로그램 이해에 도움이 된다면 한 줄에 여러 문장의 입력도 가능하다.

그림 3-7 블록과 들여쓰기의 이해

들여쓰기와 적절한 소스 구성

- 블록 내부에서 문장들을 탭(tab) 정도만큼 오른쪽으로 들여 쓰는 소스 작성 방식
- 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;
}
```

그림 3-8 적절한 줄 구분과 빈 줄의 삽입, 들여쓰기에 의한 소스 작성 방법

주석의 정의와 중요성

- **주석(comments)**
  - 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

NOTE: **주석의 습관화와 다양한 주석 스타일**

자바의 경우, 주석을 통하여 자동으로 인터넷 문서를 작성할 수 있는 방식도 제공한다. 이와 같이 소프트웨어 개발에서 주석은 바로 문서화(documentation) 작업에 활용될 수 있으니 주석을 습관화해야 한다. 주석의 스타일은 다음과 같이 다양하다.

```
/*
솔루션 / 프로젝트 / 소스파일: Ch02 / Prj01 / comments.c
C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
V 1.0 2015. 06. 29(화) 강 환수 작성
*/
```

```
/*************************************************************************
소스: comments.c
내용: C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
버전: V 1.0 2015. 06. 29(화) 강 환수 작성
*************************************************************************/
```

```
/*************************************************************************/
//  소스: comments.c
//  내용: C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
//  버전: V 1.0 2015. 06. 29(화) 강 환수 작성
/*************************************************************************/
```

```
/**
 *  소스: comments.c
 *  내용: C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
 *  버전: V 1.0 2015. 06. 29(화) 강 환수 작성
 **/
```

그림 3-9 주석의 예

주석 2가지 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

그림 3-11 다중 블록주석 오류

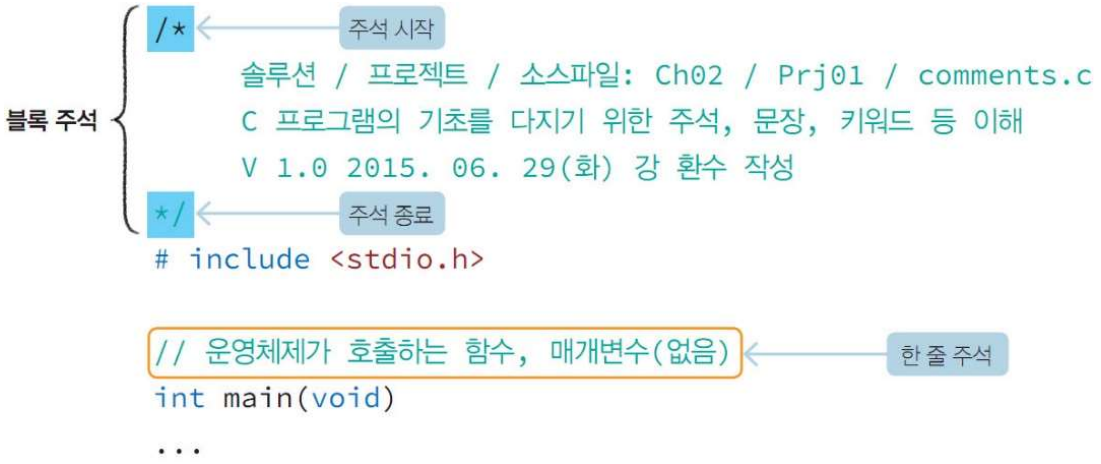


그림 3-10 블록 주석과 한 줄 주석 예

주석, 문장, 키워드 등 이해

예제 comments.c

- 키워드와 식별자 그리고 주석 등을 이해하기 위한 간단한 소스
  - 솔루션과 프로젝트: Ch03 / Prj01
  - 소스파일: comments.c

설명

- 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

실습예제 3-1 comments.c

C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj01 / comments.c
03     C 프로그램의 기초를 다지기 위한 주석, 문장, 키워드 등 이해
04     V 1.0 2016.
05  */
06  #include <stdio.h>
07
08  // 운영체제가 호출하는 함수, 매개변수(없음)
09  int main(void)
10  {
11      puts("2장 첫 C 프로그램!\n");
12
13      printf("키워드: int void return 등\n");
14      printf("식별자: main puts printf 등\n");
15      printf("블록: { ... }\n");
16
17      //인자인 문자열 내부는 //주석도 일반 문자열로 인식
18      printf("한 줄 주석: // 이 줄 끝까지 한 줄 주석입니다.\n");
19      // /*블록 주석*/도 일반 문자열로 인식
20      printf("블록 주석: /* 여러 줄에 걸친\n블록 주석입니다./* \n");
21
22      return 0;
23  }
```

주석은 모두 초록색으로 표시

한 줄 주석 //에서 시작 이후부터는 어떤 입력도 주석으로 인식

문자열 내부이므로 주석으로 인식하지 못한다.

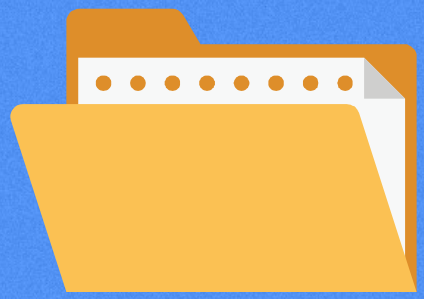
설명

```
01~05 /* ... */: 여러 줄에 걸친 블록 주석으로 컴파일에서는 제외되는 부분
06    컴파일 되기 전에 일련의 작업을 지시하는 전처리 문장
08    // 운영체제가 …: 한 줄 주석으로 // 이후의 열부터 그 행의 끝까지 주석에 해당되므로 컴파일에서 제외
10    {: main() 함수의 몸체 구현 시작을 의미하며, 23행의 몸체 구현 종료 문장 }와 대응
11    puts(): 매개변수로 입력된 문자열을 출력하는 문장으로 첫 실행 문장
22    return: 함수의 결과값을 반환하는 문장으로 반환값 0을 반환하고 프로그램 종료
23    }: 10줄의 함수 시작과 대응되는 함수 종료 의미
```

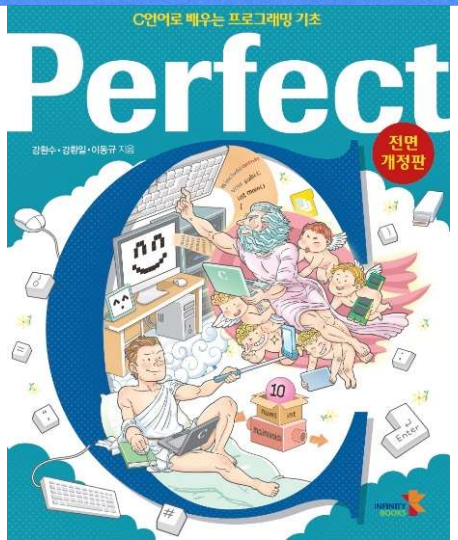
실행결과

```
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.*/
```

"\n"의 출력으로 그 다음 실행은 다음 줄에 " 블록 주석입니다.*/"가 출력된다.



02. 자료형과 변수선언



자료형과 변수 개요

자료형 분류와 변수의 개념

자료형

- 프로그래밍 언어에서 자료를 식별하는 종류
  - 기본형(basic types)
  - 유도형(derived types)
  - 사용자정의형(user defined types)

저장공간

- 저장 공간을 변수(variables)라 부름
- 변수에는 고유한 이름이 붙여지며
  - 기억장치인 메모리에 위치
- 용기에 다양한 식재료를 담듯이
- 변수에 여러 값을 저장할 수 있고
  - 저장되는 값에 따라 변수 값은 바뀔 수 있으며
  - 마지막에 저장된 하나의 값만 저장 유지

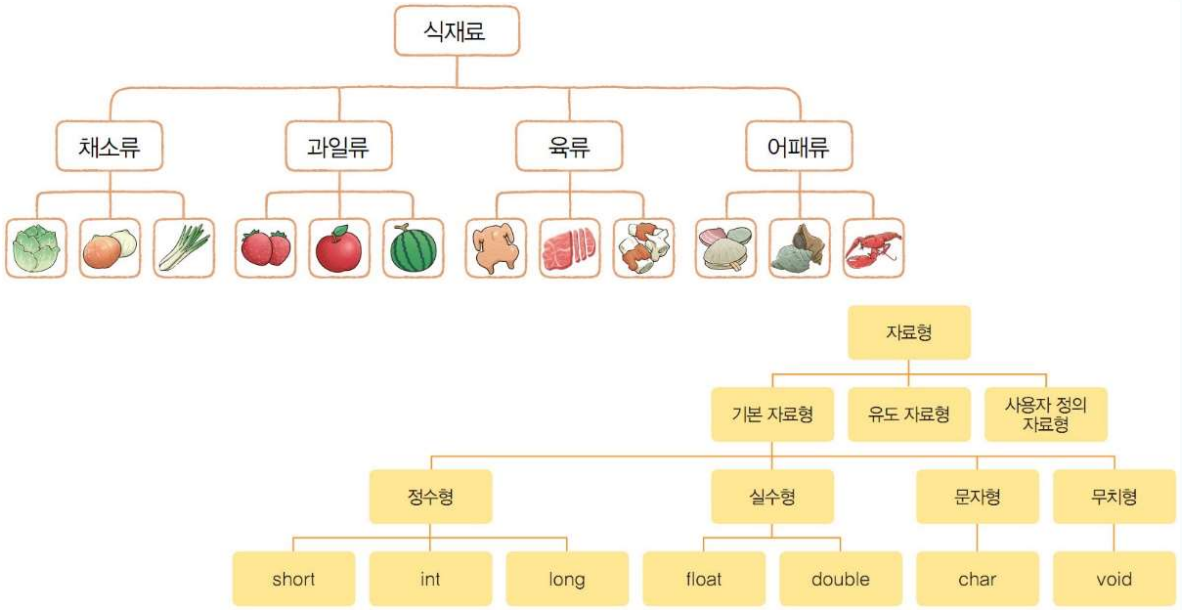


그림 3-13 자료형과 식재료의 분류

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.

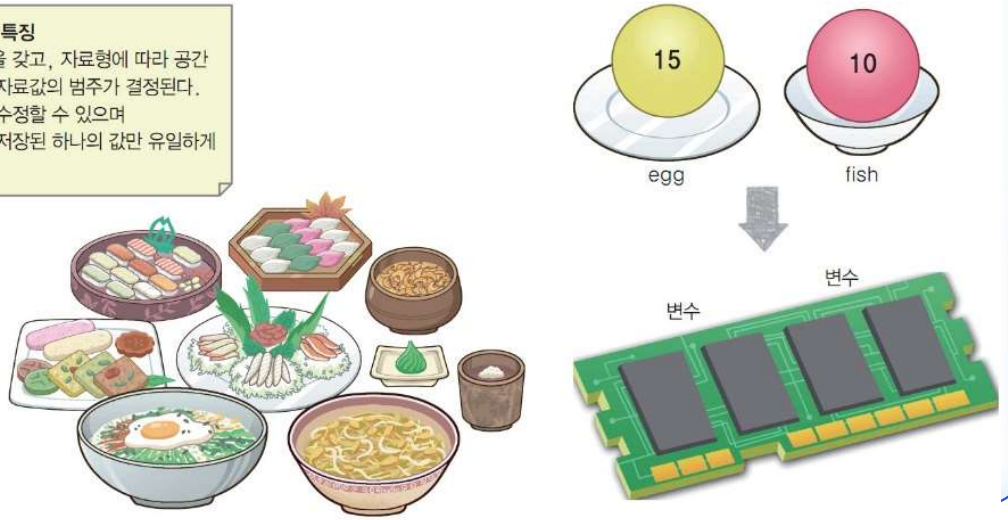


그림 3-14 그릇과 변수

변수선언

- **변수선언**
  - 그릇을 변수라고 한다면
    - 그릇에 이름을 붙여 준비하는 것을 변수선언
  - 컴파일러에게 프로그램에서 사용할 저장 공간인 변수를 알리는 역할
  - 변수는 고유한 이름이 붙여지고, 자료값이 저장되는 영역
- **자료형을 지정한 후 변수이름을 나열**
  - int, double, float와 같이 자료형 키워드를 사용
  - 변수이름은 관습적으로 소문자를 이용
    - 사용 목적에 알맞은 이름으로 영역에서 중복되지 않도록
    - 변수선언도 하나의 문장이므로 세미콜론으로 종료
  - 변수선언 이후에는 정해진 변수이름으로 값을 저장하거나 값을 참조 가능

그림 3-15 변수선언의 의미

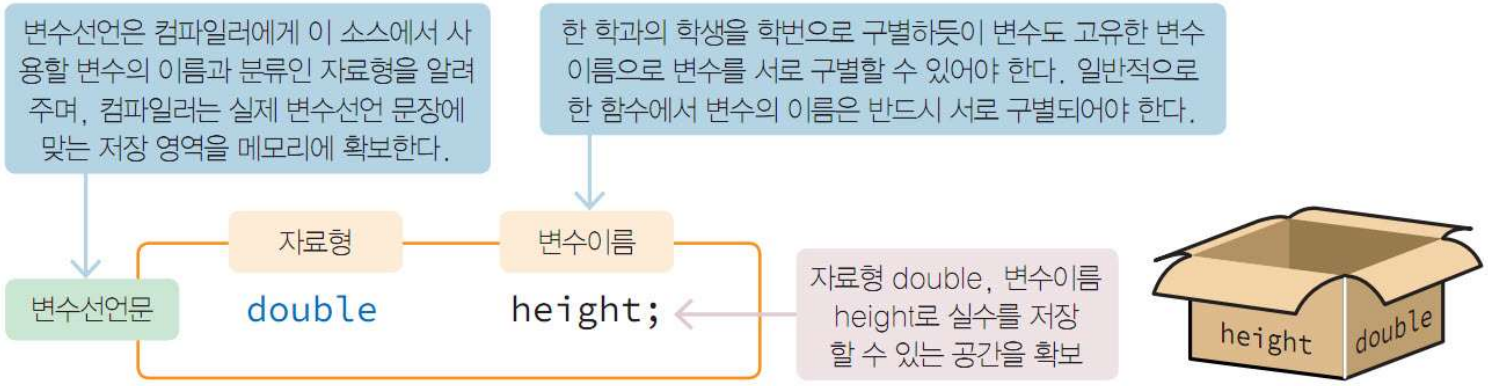


그림 3-16 자료형을 이용한 변수선언

변수에 저장 값 대입

예제 var.c

- 변수 sum. credits 선언과 사용

대입문

- 원하는 자료값을 선언된 변수에 저장
- 대입연산자(assignment operator)표시인 '='를 사용
  - 오른쪽에 위치한 값을 이미 선언된 왼쪽 변수에 저장한다라는 의미
- 대입문(assignment statement)
  - 변수명 age를 int 형으로 선언한 후, 변수 age에 20을 저장하는 문장
  - 가장 마지막에 저장된 값만이 남음

실습예제 3-2 var.c

변수의 선언과 사용

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj02 / var.c
03     C 프로그램의 기초를 다지기 변수선언 이해
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     int snum;    //변수 선언
12     int credits;
13
14     snum = 20163021; //값 지정
15     credits = 18;
16
17     printf("학번: %d\n", snum);
18     printf("신청학점: %d\n", credits);
19
20     return 0;
21  }
```

설명

```
11    정수형 int 변수 snum 선언
12    정수형 int 변수 credits 선언
14    변수 snum에 학번 저장
15    변수 credits에 신청한 학점 저장
17~18 각각 변수 snum과 credits에 저장된 값을 출력
```

실행결과

```
학번: 20163021
신청학점: 18
```

그림 3-19 변수선언과 대입문

변수 초기화

예제 sum.c

- 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

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

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

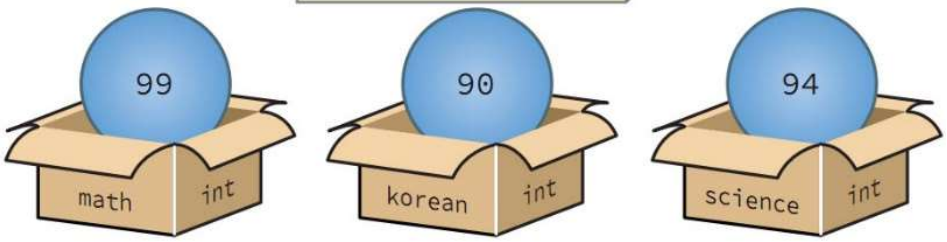


그림 3-23 여러 변수 선언과 초기화

실습예제 3-3 sum.c

변수 초기화 이해

```c
/*
   솔루션 / 프로젝트 / 소스파일: Ch03 / Prj03 / sum.c
   변수 초기화 이해
   V 1.0 2016.
*/

#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;
}
```

설명

```
11    정수형 int 변수 math를 선언하면서 초기값으로 99 저장
12    정수형 int 변수 korean를 선언하면서 초기값으로 90 저장
14    정수형 int 변수 science를 선언하면서 초기값으로 아무 값도 저장하지 않고
15    바로 뒤에 변수 science에 94를 저장
18    정수형 int 변수 total을 선언하면서 새 과목의 합을 저장
20~23 각각 수학, 국어, 과학 점수와 총점을 출력
```

실행결과

```
학번: 20163021
신청학점: 18
```

Tip

- **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
- **초기화 되지 않은 지역변수의 저장값과 오류**
  - 함수 내부에서 선언된 변수를 지역 변수(local 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

```
l-value  =  r-value;

21 = 20 + 1;        //오류발생   오류: 식이 수정할 수 있는 lvalue여야 합니다.
```

그림 3-21 l-value와 r-value

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

그림 3-24 초기화 되지 않은 지역변수의 사용에서 컴파일 오류

변수의 3요소와 이용

예제 subtraction.c

- 변수 num1, num2 사용
- 변수 difference에 차를 저장

변수의 3요소

- 변수이름, 변수의 자료형, 변수 저장 값

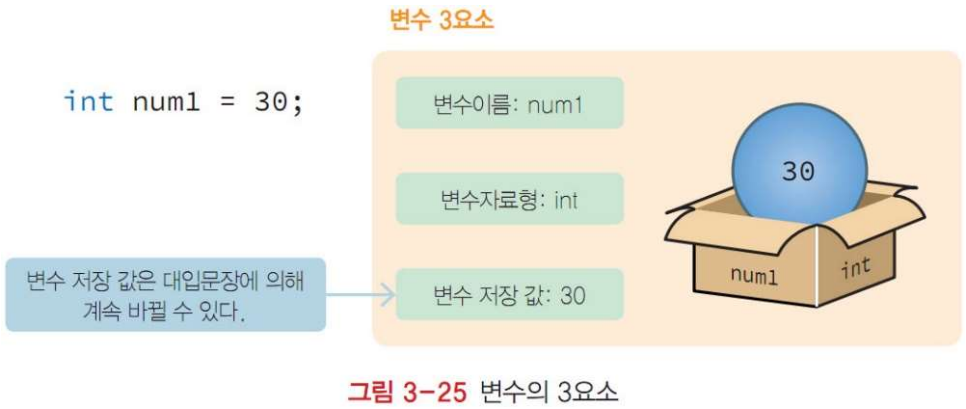


그림 3-25 변수의 3요소

=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

실습예제 3-4 subtraction.c

변수의 l-value와 r-value

```c
/*
   솔루션 / 프로젝트 / 소스파일: Ch03 / Prj04 / subtraction.c
   변수의 저장공간 자체와 변수에 저장된 값을 의미
   V 1.0 2016.
*/

#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;
}
```

설명

15 difference는 l-value로 변수 자체를 의미하고, num1과 num2는 r-value로 저장값이 연산에 참여

17~18 각각 변수 num1과 num2를 출력한 후, 다음 행에 다시 difference를 출력

실행결과

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

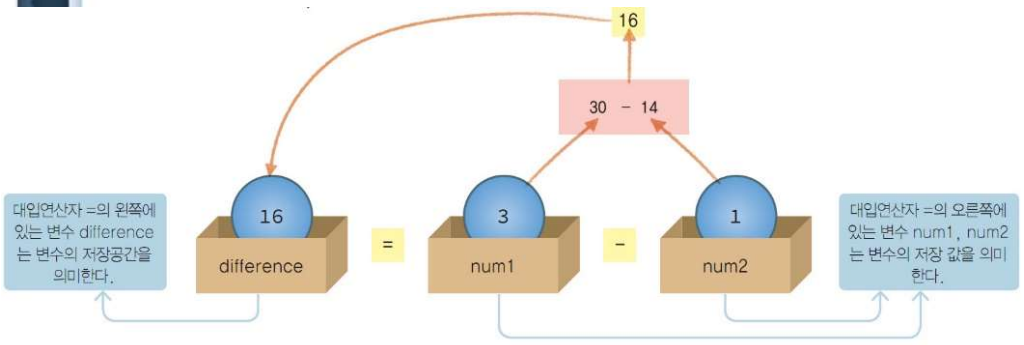


그림 3-26 변수 저장공간과 변수값의 이용

LAB 두 정수의 합, 두 부동소수의 차 출력

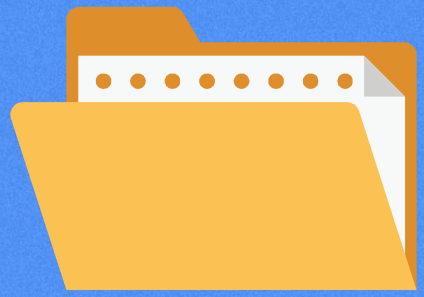
- **두 정수의 합과 두 실수의 차가 출력되는 프로그램**
  - 정수를 위한 자료형은 int로, 실수를 위한 자료형은 double로 이용
  - 합을 위한 연산자 +, 두 실수의 차를 위한 연산자 –와 결과 저장을 위한 변수 difference
- **결과**
  - 합: 73
  - 차: -7.003000

Lab 3-1 basictype.c

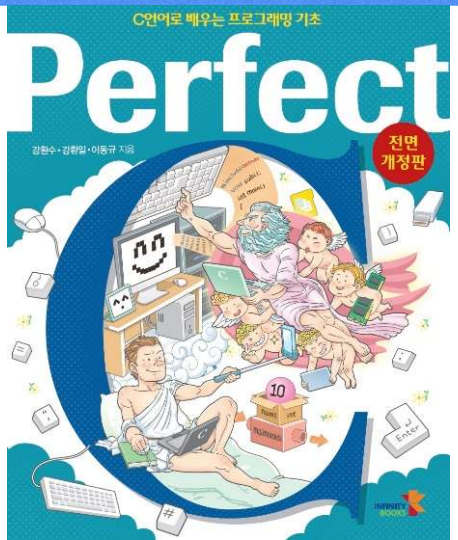
```c
// basictype.c: 두 정수의 합, 두 실수의 차 출력

#include <stdio.h>

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

   double x = 38.342, y = 45.345;      //두 실수 선언과 초기값 대입
   ________________;                   //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;
}
```

정답

```c
sum = a + b;                        //두 정수의 합 구하기
double difference;                  //두 실수의 차을 저장할 변수 선언
difference = x - y;                 //두 실수의 차 구하기
printf("합: %d\n", sum);            //두 정수의 합 출력
printf("차: %f\n", difference)      //두 실수의 차 출력
```



03. 기본 자료형



자료형

- **C의 자료형**
  - 기본형(basic data types), 유도형(derived data types), 사용자정의형(user 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**
  - 배열(array), 포인터(pointer), 함수(function) 등으로 구성
- **사용자정의형**
  - 기본형과 유도형을 이용하여 프로그래머가 다시 만드는 자료형
  - 열거형(enumeration)
  - 구조체(structure)
  - 공용체(union)

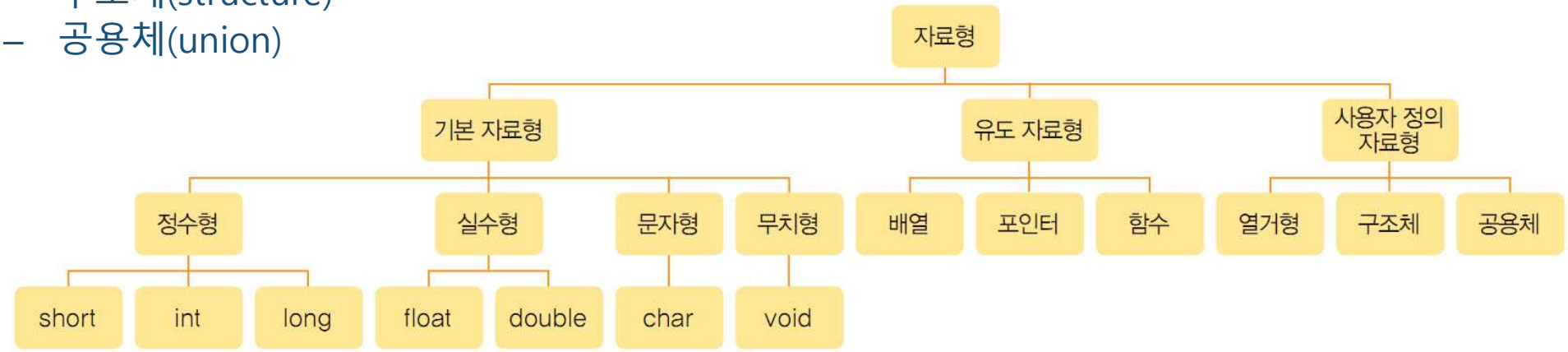


그림 3-27 C 언어의 다양한 자료형

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

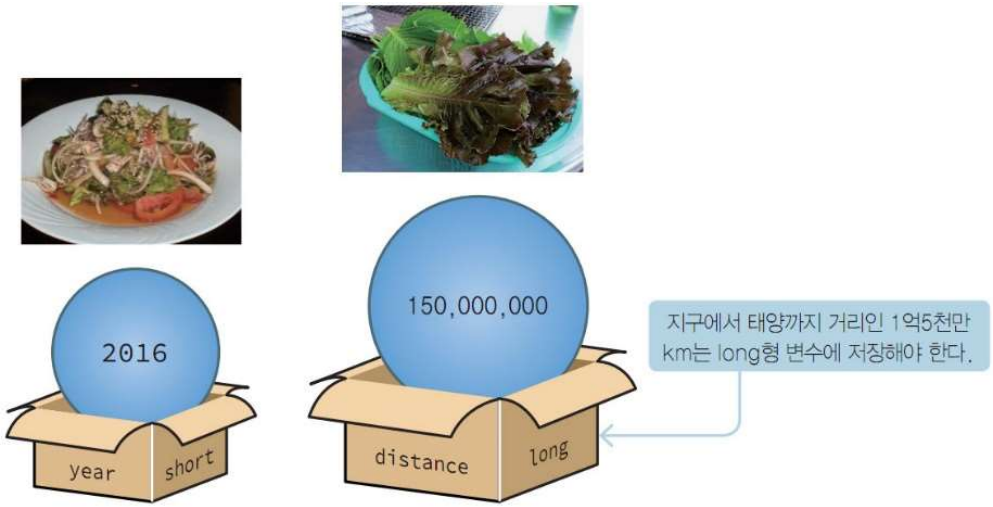


그림 3-28 크기나 사용 범위에 따른 자료형의 선택

그림 3-29 부호가 있는 signed(음수, 0과 양수) 정수를 위한 자료형 세 종류

정수형

- **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
  - 저장공간 크기 n비트인 singed
    - -2^{n-1}에서 $2^{n-1}-1$까지 유효
  - 저장공간 크기 n비트인 unsinged
    - 0에서 2^{n-1}까지 유효

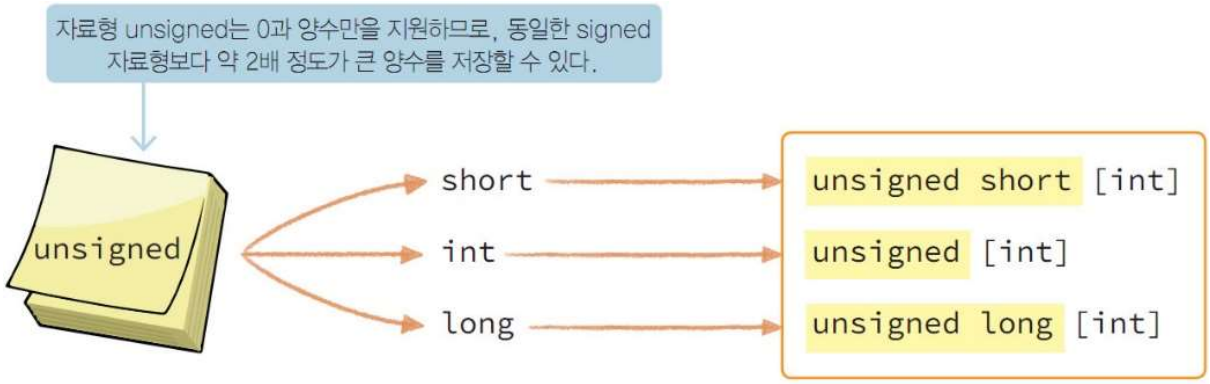


그림 3-30 부호가 없는 [0과 양수] 정수를 위한 자료형 세 종류

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

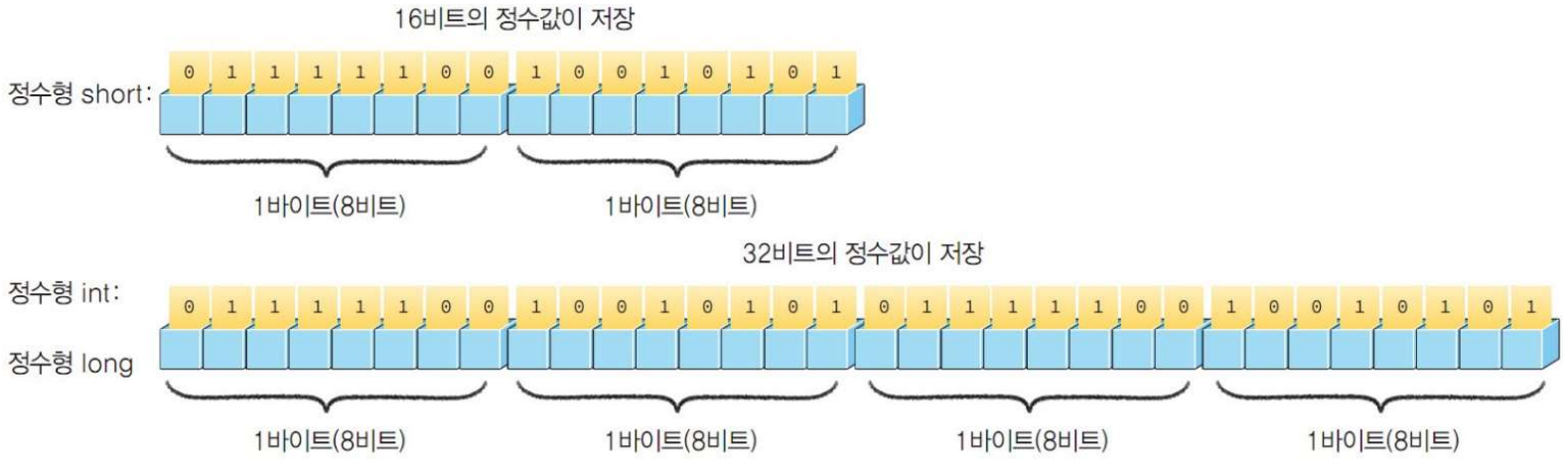


그림 3-31 정수 자료형의 저장공간 크기

다양한 정수형의 사용

예제 integer.c

- 21억보다 큰 정수의 사용

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

그림 3-32 자료형 long long으로 지원 가능한 우주의 행성 간의 거리

C99 표준

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

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)

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

실습예제 3-5 integer.c

정수형 자료형 변수의 선언과 활용

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj05 / integer.c
03     정수형 자료형 변수의 선언과 활용
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     short sVar = 32000;          //-32767에서 32767까지
12     int   iVar = -2140000000;    //약 21억 정도까지 저장 가능
13
14     unsigned short int   usVar = 65000;       //0에서 65535까지 저장 가능
15     unsigned int         uiVar = 4280000000;  //약 0에서 42억 정도까지 저장 가능
16
17     printf("저장값: %d %d\n", sVar, iVar);
18     printf("저장값: %u %u\n", usVar, uiVar);
19
20     long long dist1 = 2720000000000; //지구와 천왕성 간의 거리(km) 27억 2천
21     __int64 dist2 = 4500000000000;   //태양과 해왕성 간의 거리(km) 45억
22
23     printf("지구와 천왕성 간의 거리(km): %lld\n", dist1);
24     printf("태양과 해왕성 간의 거리(km): %lld\n", dist2);
25
26     return 0;
27  }
```

unsinged를 출력할 경우, 형식제어문자를 %u로 기술

long long과 __int64를 출력할 경우, 형식제어문자를 %lld로 기술

설명

```
11~15 적절한 변수의 자료형으로 변수 선언
17~18 signed 정수는 %d로 출력하고 unsigned 정수는 %u로 출력
20~21 자료형 long long, __int64는 64비트로 약 922경까지 저장 가능
23~24 자료형 long long, __int64는 %lld로 출력
```

실행결과

```
저장값: 32000 -2140000000
저장값: 65000 4280000000
지구와 천왕성 간의 거리(km): 2720000000000
태양과 해왕성 간의 거리(km): 4500000000000
```

부동소수 자료형

예제 float.c

- 실수를 위한 저장 공간 사용

실습예제 3-6 float.c

부동소수형 변수의 선언과 활용

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj06 / float.c
03     부동소수형 변수의 선언과 활용
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     float        x = 3.14F;       //float x = 3.14;인 경우, 경고 발생
12     double       y = -3.141592;   //double 저장공간 크기는 float의 2배
13     long double  z = 180000000.0; //double과 long double은 저장공간이 모두 64비트
14
15     printf("저장값: %f %f %f\n", x, y, z);
16
17     return 0;
18  }
```

설명

11 부동소수 상수 3.14F로 반드시 F나 f 삽입

15 부동소수는 %f로 출력

실행결과

```
저장값: 3.140000 -3.141592 180000000.000000
```

부동소수형 3가지

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

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

상수 3.14F

- 소수 3.14와 같은 표현은 모두 자료형 double로 인식
- float형 변수에 저장하면 컴파일 경고나 오류가 발생

```
float x = 3.14;       //float x = 3.14;인 경우, 경고 발생
```

```
warning C4305: '초기화 중' : 'double'에서 'float'(으)로 잘립니다.
```

그림 3-33 float 형 변수에 부동소수 상수로 저장한 경우의 경고

문자형 자료형

예제 char.c

- 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

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

실습예제 3-7 char.c

문자형 변수의 선언과 이용

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj07 / char.c
03     문자형 변수의 선언과 이용
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     char c1 = 'a';       //소문자 a
12     char c2 = 65;        //대문자 A가 코드값 65
13     char c3 = '\132';    //대문자 Z의 8진수 코드값 132
14     char c4 = '\x5A';    //대문자 Z의 16진수 코드값 5A
15
16     printf("저장값(문자): %c %c %c %c\n", c1, c2, c3, c4);
17     printf("저장값(정수): %d %d %d %d\n", c1, c2, c3, c4);
18
19     return 0;
20  }
```

%c는 문자가 출력되며, %d는 문자의 코드값 십진수가 출력된다.

설명

```
11~14 문자, 코드값 십진수, 팔진수, 십육진수로 저장
15    %c는 문자로 출력
16    %d는 코드값을 정수로 출력
```

실행결과

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

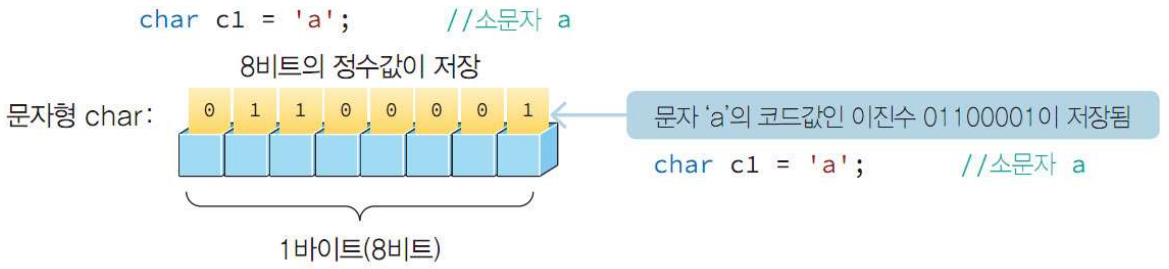


그림 3-34 문자형 자료값의 표현과 저장공간

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

표 3-6 아스키 코드표

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

자료형 14가지 종류와 표현범위

예제 sizeof.c

- 연산자 sizeof 사용

문자형 char

- 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(−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}

실습예제 3-8 size.c

연산자 sizeof를 이용한 저장공간 크기 출력

```c
/*
   솔루션 / 프로젝트 / 소스파일: Ch03 / Prj08 / size.c
   연산자 sizeof를 이용한 저장공간 크기 출력
   V 1.0 2016.
*/

#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;
}
```

실행결과

```
       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

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

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

그림 3-35 연산자 sizeof의 사용법

오버플로와 언더플로

예제 overflow.c

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

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

그림 3-36 오버플로 발생 원리

언더플로

- 실수형 float 변수에 정밀도가 매우 자세한 수를 저장하면 언더플로(underflow)가 발생
  - 0이 저장

실습예제 3-9 overflow.c

오버플로와 언더플로의 발생

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj09 / overflow.c
03     오버플로와 언더플로의 발생
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     unsigned char  uc = 255 + 1;
12     short          s = 32767 + 1;
13     float          min = 1.175E-50;
14     float      max = 3.403E39;
15
16     printf("%u\n", uc);     //오버플로 발생
17     printf("%d\n", s);      //오버플로 발생
18     printf("%e\n", min);    //언더플로 발생
19     printf("%f\n", max);    //오버플로 발생
20
21     return 0;
22  }
```

약 10^{38} 이상 되는 실수는 저장되지 못하고 오버플로 발생

설명

16 연산 255 + 1의 결과는 256이나 자료형 unsigned char에서 256은 오버플로가 발생해서 0이 저장됨

17 연산 32767 + 1의 결과는 32768이나 자료형 short에서 32768은 오버플로가 발생해서 -32767이 저장됨

18 실수 1.175E-50은 매우 작은 수로 float에서는 언더플로가 발생하여 0이 저장됨

19 실수 3.403E39는 매우 큰 수로 float에서는 오버플로가 발생하여 무한대(infinite)라는 의미로 inf가 출력됨

경고

매우 큰 수로 오버플로 발생, 무한대를 의미하는 inf가 저장 출력됨

```
출력
출력 보기 선택(S): 빌드
1>------ 빌드 시작: 프로젝트: Prj09, 구성: Debug Win32 ------
1>  overflow.c
1>g:₩[2016 c]₩ch03₩prj09₩overflow.c(11): warning C4305: '초기화 중': 'int'에서 'unsigned char'(으)로 잘립니다.
1>g:₩[2016 c]₩ch03₩prj09₩overflow.c(13): warning C4305: '초기화 중': 'double'에서 'float'(으)로 잘립니다.
1>g:₩[2016 c]₩ch03₩prj09₩overflow.c(14): warning C4056: 부동 소수점 상수 산술 연산에서 오버플로가 발생했습니다.
1>g:₩[2016 c]₩ch03₩prj09₩overflow.c(14): warning C4756: 상수 산술 연산에서 오버플로가 발생했습니다.
1>  Prj09.vcxproj -> G:₩[2016 C]₩Ch03₩Debug₩Prj09.exe
========== 빌드: 성공 1, 실패 0, 최신 0, 생략 0 ==========
```

매우 작은 수로 언더플로 발생, 0이 저장됨

컴파일 시 발생하는 경고 문구

실행결과

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

LAB 아스키 코드값 126 문자 '~'의 다양한 출력

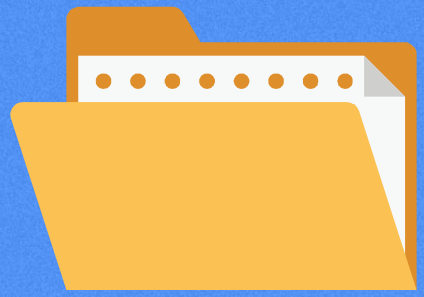
- **다음 결과로 출력되는 프로그램을 작성**
  - 문자 '~'의 코드값: 십진수 126, 팔진수 176, 십육진수 72
  - 출력을 위한 함수 print()에서 %d로 정수를, %c로 문자를 출력
- **결과**
  - 126
  - ~
  - ~
  - ~

Lab 3-2 intchar.c

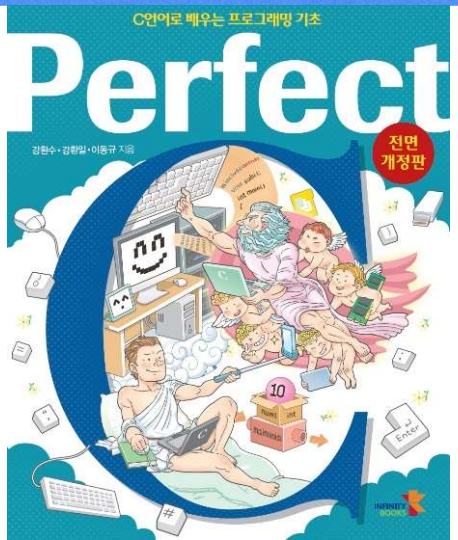
```c
// intchar.c: 아스키 코드값 126 문자 '~'의 다양한 출력

#include <stdio.h>

int main(void)
{
   int ch = 126;

   printf("%d\n", ___);     //십진 코드값 출력
   printf("%___\n", ch);    //문자 출력
   printf("%c\n", '\___'); //문자 출력
   printf("%c\n", '\x__'); //문자 출력

   return 0;
}
```

정답

```c
printf("%d\n", ch);        //십진 코드값 출력
printf("%c\n", ch);        //문자 출력
printf("%c\n", '\176');    //문자 출력
printf("%c\n", '\x7e');    //문자 출력
```



04. 상수 표현방법



상수의 종류와 표현 방법

- **상수(constant)**
  - 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 상수(const constant)
    - 매크로 상수(macro constant)
    - 열거형 상수(enumeration constant)

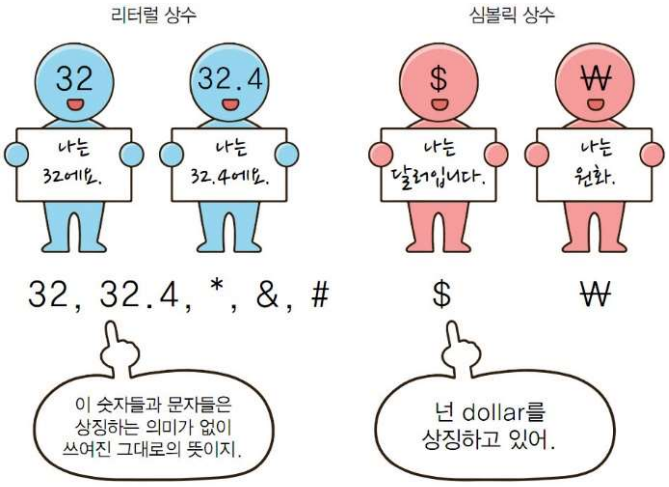


그림 3-38 리터럴 상수와 심볼릭 상수

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};`

정수와 실수, 문자와 문자열

- **리터럴 상수**
  - 정수, 실수
  - 문자, 문자열 상수
- **문자 상수 표현**
  - 문자 하나의 앞 뒤에 작은 따옴표(single quote)를 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

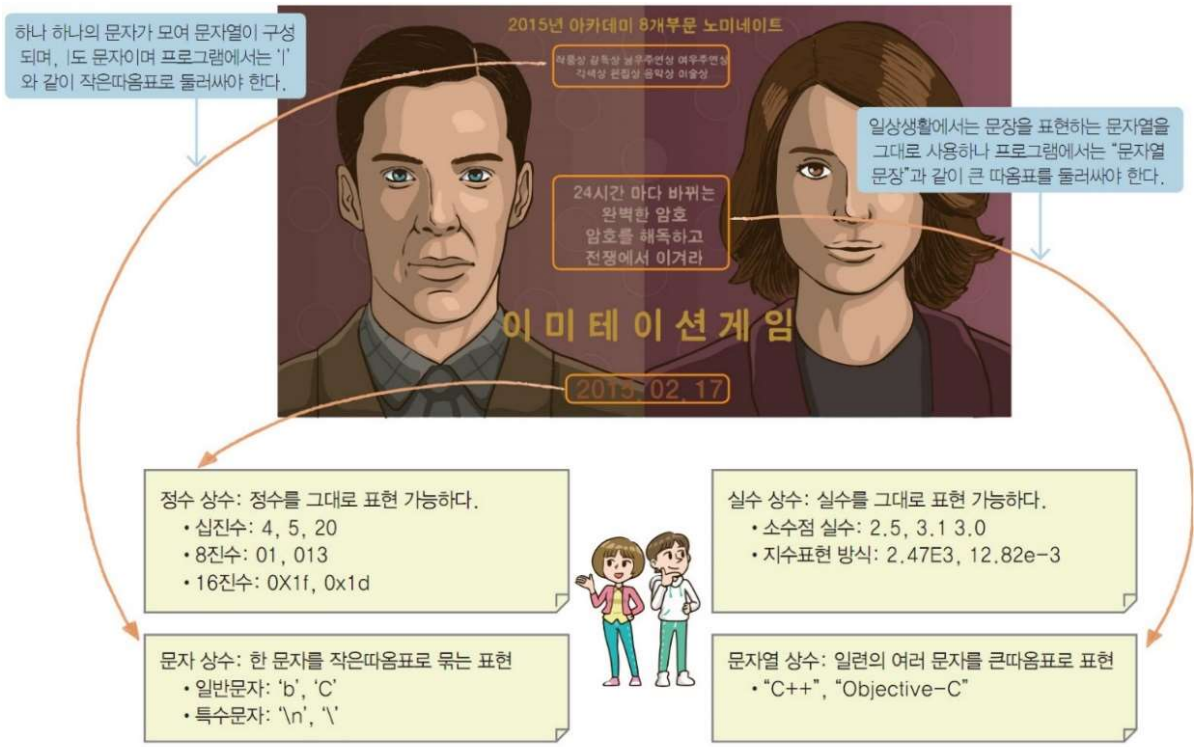


그림 3-39 리터럴 상수의 종류와 표현방법

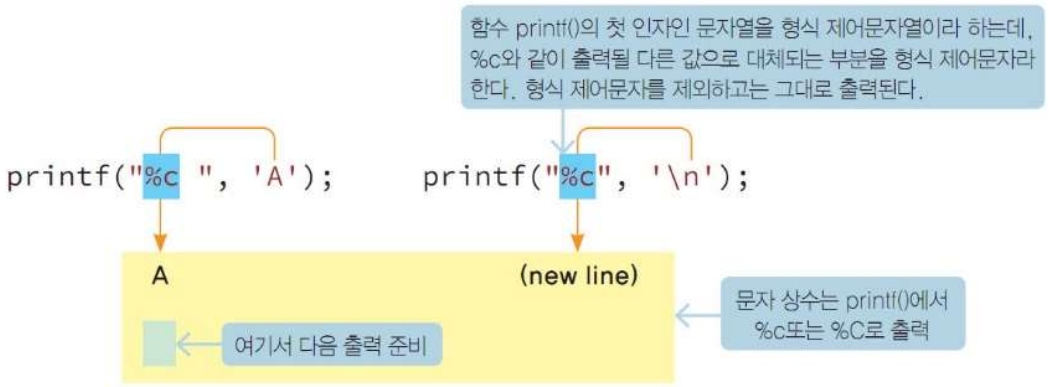


그림 3-40 문자 상수의 표현과 출력

이스케이프 문자를 비롯해서 다양한 문자 리터럴의 표현

예제 charliteral.c

- 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

표 3-9 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

실습예제 3-10 charliteral.c

이스케이프 문자를 비롯해서 다양한 문자 리터럴의 표현

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj10 / charliteral.c
03     이스케이프 문자를 비롯해서 다양한 문자 리터럴의 표현
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     printf("%Casic", 'B');      printf("%c", '\n');
12
13     char sq = '\'';          //작은따옴표
14
15     printf("BCPL\tB\tC\tJava\n");   //문자열 내부에서 \t(탭) 문자 사용
16     printf("%C\7\n", '\a');   //알람 문자를 2번 출력하고 공백 줄
17     printf("%c자바언어'\n", sq);   //문자열 내부에서는 '(작은따옴표) 그대로 사용 가능
18
19     //문자열 내부에서는"(큰따옴표) 반드시 \"로 사용
20     printf("\"C언어\" 정말 재미있다!\n");
21
22     return 0;
23  }
```

Basic 출력

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

설명

```
15  문자열 내부에서 탭은 이스케이프 문자 \t을 사용
16  알람은 \a, \7 모두 사용 가능
17  문자열 내부에서 작은따옴표 표현은 '와 이스케이프 문자 \' 모두 사용 가능
20  문자열 내부에서 큰따옴표 표현은 "는 사용할 수 없으며, 이스케이프 문자 \" 사용 가능
```

실행결과

```
Basic
BCPL    B       C       Java

'자바언어'
"C언어" 정말 재미있다!
```

정수 리터럴 상수

- **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
- **이진수와 십육진수 표현방식**
  - 상수의 정수표현은 십진수로 인식

그림 3-41 정수형 리터럴 상수

  - 숫자 0을 정수 앞에 놓으면 팔진수(octal number)로 인식
  - 숫자 0과 알파벳으로 0x, 0X를 숫자 앞, 십육진수(hexadecimal number)로 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

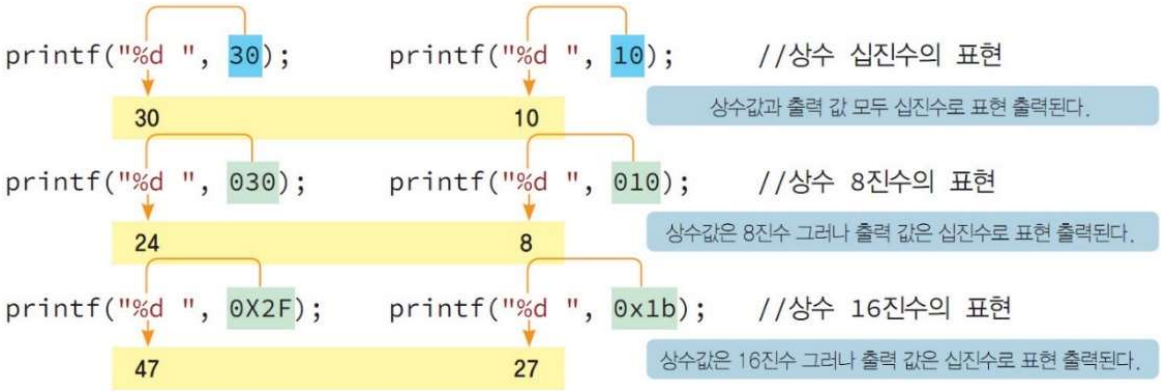


그림 3-42 팔진수와 십육진수의 상수 표현과 출력

실수 리터럴 상수

- **지수표현 방식**
  - 3.14E+2는 $3.14*10^2$
- **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 리터럴 상수**
  - 실수형 상수도
    - float, double long double
  - 소수는 double 유형이며
  - float 상수
    - 숫자 뒤에 f나 F를 붙임
  - long double 상수
    - 숫자 뒤에 L 또는 l을 붙여 표시

그림 3-43 실수 상수의 지수표현 방식과 출력 방법

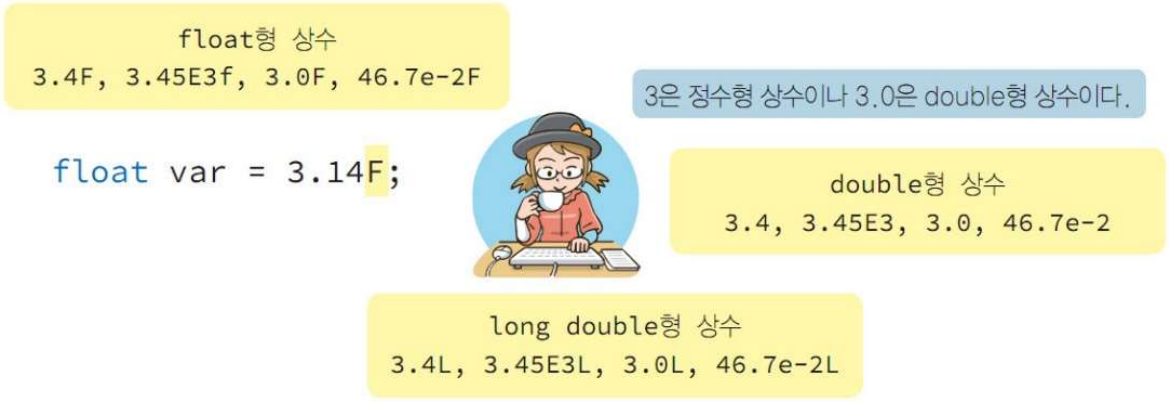


그림 3-44 실수형 리터럴 상수

심볼릭 const 상수

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

```
//키워드 const로 상수 만들기

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

error C2166: l-value가 const 개체를 지정합니다.

RATE = 0.032;     //수정이 불가능

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

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

그림 3-46 심볼릭 상수 선언과 오류

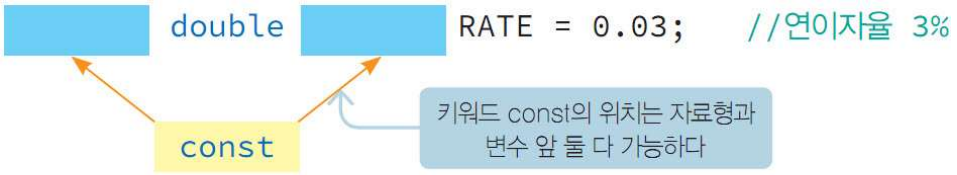


그림 3-45 키워드 const의 위치

여러 심볼릭 상수의 이용

예제 const.c

- 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.

그림 3-47 문자열 리터럴 상수의 변수 저장과 심볼릭 상수 정의

실습예제 3-12 const.c

키워드 const를 사용한 상수 선언

```
01  /*
02     솔루션 / 프로젝트 / 소스파일: Ch03 / Prj12 / const.c
03     키워드 const를 사용한 상수 선언
04     V 1.0 2016.
05  */
06
07  #include <stdio.h>
08
09  int main(void)
10  {
11     //키워드 const로 상수 만들기
12     const double RATE = 0.03;    //연이자율 3%
13     int deposit = 800000;
14
15     //RATE = 0.032;    //수정이 불가능
16     printf("이자율: %f\n", RATE);
17     printf("계좌 잔고: %d\n", deposit);
18     printf("이자액: %f\n", deposit * RATE);
19
20     //문자열을 변수에 저장
21     char* str = "좋은 C 언어 입문서"; //char *str, char * str 모두 가능
22     char* const title = "진보된 C 언어"; //title에 다른 문자열 상수 저장이 불가
23
24     str = "최근 가장 좋은 C 언어 입문서";
25     //title = "C 언어 스케치"; //수정 불가능
26
27     printf("\n%s: %s\n", str, title); //문자열 변수 출력
28
29     return 0;
30  }
```

필요하면 이자율을 여기서 직접 수정할 수 있다.

printf()에서 변수 이름과 %s로 출력한다.

설명

12 일반 변수 선언에서 가장 앞부분에 키워드 const를 삽입
18 별표 *(키보드에는 수학에서의 x 기호는 없음)는 곱하기 연산자를 나타냄
15 상수는 수정할 수 없으므로 주석을 빼면 컴파일 오류가 발생
21 변수선언에서 char* str는 문자 주소값을 저장할 수 있는 변수 str을 선언하는 의미이며, 선언 이후에 str은 주소값을, *str은 주소가 가리키는 첫 문자 자체를 참조
22 변수 title 자체인 첫 문자의 주소값을 수정할 수 없도록 하는 변수 선언
24 변수 str의 주소값은 수정할 수 있으므로 다른 문자열로 대입

실행결과

```
이자율: 0.030000
계좌 잔고: 800000
이자액: 24000.000000

최근 가장 좋은 C 언어 입문서: 진보된 C 언어
```

열거형 상수의 이용

예제 const.c

- Enum을 사용한 정수 상수의 이용

키워드 enum

- 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.

그림 3-48 열거형 상수 목록 선언

- 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

실습예제 3-13 enum.c

키워드 enum으로 만드는 열거형 정수 상수 목록

```c
/*
   솔루션 / 프로젝트 / 소스파일: Ch03 / Prj13 / enum.c
   키워드 enum으로 만드는 열거형 정수 상수 목록
   V 1.0 2016.
*/

#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;
}
```

설명

```
12  상수 SUN은 0에서부터 순차적으로 1씩 증가되어 지정되며, 상수 SAT는 6으로 지정
19  FALSE는 0, TRUE는 1로 지정
21  모두 지정한 값으로 상수값이 지정
22  열거형 상수는 모두 정수이므로 printf()에서 %d로 출력
```

실행결과

```
일요일 상수: 0
수요일 상수: 3
LINE: 1, RECT: 4, CIRCLE: 9
false: 0, cpp: 1983, csharp: 2000
```

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

정수형 상수 목록

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

그림 3-49 직접 정수를 지정하는 열거형 상수 목록

매크로 상수

- **전처리기 지시자 #define**
  - 매크로 상수(macro constant)를 정의하는 지시자
  - 주로 대문자 이름으로 정의
  - 전처리기(preprocessor)
    - 매크로 상수를 모두 #define 지시자에서 정의된 문자열로 대체(replace)

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

그림 3-51 매크로 상수 KPOP과 PI

표 3-12 자료형과 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

```
#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 */
```

그림 3-52 헤더파일 limits.h의 주요 매크로 상수

LAB 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

Lab 3-3 minmaxfloat.c

```
01  // minmaxfloat.c: 부동소수형 세가지의 최대 최소 상수 출력
02
03  #include <stdio.h>
04  #include <float.h>      //부동소수형 상수가 정의된 헤더파일 삽입
05
06  int main(void)
07  {
08     printf("float 범위: %e %e\n", _____________________________);
09     printf("double 범위: %e %e\n", _____________________________);
10     printf(_____________________________________________________);
11
12     return 0;
13  }
```

정답

```
08  printf("float 범위: %e %e\n", FLT_MIN, FLT_MAX);
09  printf("double 범위: %e %e\n", FLT_MIN, FLT_MAX);
10  printf("long double 범위: %e %e\n", LDBL_MIN, LDBL_MAX);
```

Thank you