Chapter 03

변수와 자료형

학습목표

- 식별자와 예약어 및 사용 관례에 대해 알아봅니다.
- 변수의 의미와 변수명을 지정하는 방법을 알아봅니다.
- 상수의 의미와 종류, 사용법을 알아봅니다.
- 자료형의 종류와 사용법을 알아봅니다.
- 정수 및 실수 자료형의 크기와 의미에 대해 알아봅니다.
- C 언어의 표준 입출력 함수와 형식 지정자에 대해 알아봅니다.
- 문자와 문자열 입출력 전용 함수에 대해 알아봅니다.

Chapter 03

제출문제 특강 바로가기

http://www.hanbit.co.kr/4148/ch03

01 식별자와 예약어

C 언어를 본격적으로 학습하기 전에 식별자identifier에 대해 살펴보겠습니다. 식별자란 프로그램을 구성하는 요소를 가리키는 이름을 의미하며 프로그램 작성자가 지정합니다. 프로그램을 작성할 때 식별자로 지정할 수 있는 프로그램의 요소로는 변수, 상수, 배열, 사용자가 정의하는 함수 등이 있습니다.

C 언어에서는 다음과 같은 규칙에 따라 식별자를 사용해야 합니다.

- 식별자는 문자, 숫자, 특수문자(예 : _, $)로 구성될 수 있습니다.
- 식별자의 첫 문자는 문자나 특수문자로 시작할 수 있지만 숫자는 사용할 수 없습니다.
- 식별자는 길이에 제한을 두지 않습니다.
- 같은 문자의 대 · 소문자(예 : Sum과 sum)는 서로 다른 식별자로 취급합니다.
- 식별자 중간에 공백이 들어갈 수 없습니다.
- 예약어reserved word는 식별자로 사용할 수 없습니다.

예약어는 프로그래밍 언어를 만들 때 미리 약속되어 사용하는 용어를 말합니다. C 언어(ANSI C)에는 [표 3-1]과 같이 32개의 예약어가 정의되어 있습니다.

표 3-1 C 언어의 예약어

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

다음 [표 3-2]의 예를 통해 사용 가능한 식별자와 사용 불가능한 식별자를 살펴봅시다.

표 3-2 사용 가능한 식별자와 불가능한 식별자

사용 가능한 식별자			사용 불가능한 식별자
count1	box_number	abc	1kim(숫자로 시작)
$Kim	Park	nbr	while(예약어)
address	addrave_of_score	AveOfScore	@addr(특수문자로 시작)
my_address	MyAddress		my.address(중간에 마침표가 들어감)
			my address(중간에 공백이 있음)
			printf, scanf(표준 입출력 함수의 이름. 식별자로 사용은 가능하나 함수를 사용할 수 없음)

예제 3-1 reserved.c

다음은 예약어와 표준 입출력 함수명을 식별자로 사용하는 프로그램입니다. 예약어를 식별자(변수 이름)로 사용하면 구문 오류가 발생하며, 입출력 함수명(예 : printf)은 식별자로 사용할 수는 있지만 입출력 함수를 사용할 수 없게 됩니다. 함수명을 식별자로 사용할 경우 그 함수가 식별자로 인식되어 사용할 수 없는 것입니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      int for = 30;          --- 예약어를 식별자로 사용(오류)
06      int printf = 20;       --- 출력 함수명을 식별자로 사용(가능)
07      printf = printf + 100;
08      printf("%d", for+printf); --- printf를 식별자로 인식(오류)
09 }
```

실행 결과 5번과 8번에서 컴파일 오류가 발생합니다. 5번에서는 식별자로 사용할 수 없는 예약어를 사용하여 구문 오류가 발생했습니다. 6번에서는 printf를 식별자로 사용했는데, 이처럼 함수명을 식별자로 사용할 수는 있지만 printf가 식별자로 인식되어 출력 함수를 사용할 수 없습니다.

C 언어에서 식별자의 사용은 강제적이지는 않지만 관례가 있습니다. 일반적인 변수나 배열 등의 이름에는 소문자를 사용하고, 값이 변하지 않는 상수를 선언할 경우에는 대문자를 사용합니다.

프로그램에서 식별자를 지정할 때 주의가 필요한데, 식별자를 아무 뜻 없이 a, b, c 등으로 지정하면 프로그램 내에서 의미를 알기 어렵습니다. 이해하기 쉬운 프로그램이 좋은 프로그램이므로 식별자를 지정할 때 의미를 알 수 있는 적합한 이름을 선택하는 것이 좋습니다.

다음 예는 강제적이지는 않지만(오류가 발생하지는 않지만) 좋은 식별자와 좋지 않은 식별자를 나타낸 것입니다.

표 3-3 좋은 식별자와 좋지 않은 식별자

좋은 식별자(이름만으로도 의미를 알 수 있음)				좋지 않은 식별자(이름만으로 의미를 알기 어려움)						
count	number	nbr	address	a	b	c	aaa	bbb	abc	def
addraverage	name	hap	sum	a1	b1	c1	t1	t2	t3	
MyAddress	MAX	MIN	PI							
ave_of_score	AveOfScore	my_address								

02 변수와 상수

프로그램을 작성하기 위해 가장 먼저 이해해야 하는 것이 변수variable입니다. 우리는 컴퓨터에게 일을 시키기 위해 프로그램을 작성합니다. 이때 프로그램은 다양한 종류의 데이터를 표현할 수 있어야 하고, 이러한 데이터를 표현(저장)할 때 변수를 사용합니다.

변수는 프로그램이 실행되면서 계속 변해가는 값을 저장하기 위해 사용하는 저장소입니다. 반면에 한 번 결정되면 프로그램을 수행하는 동안 변하지 않고 사용되는 값을 상수constant라고 합니다. 이 절에서는 변수의 의미와 초기화 및 사용법, 그리고 상수의 개념과 종류에 관해 살펴보겠습니다.

2.1 변수

2.1.1 변수의 필요성

변수는 단어 자체를 통해서도 알 수 있듯이 변해가는 것입니다. 라면을 먹는 과정을 프로그래밍에 비유하여 변수의 의미를 알아봅시다. 라면을 먹는 프로그램을 작성한다면 우선 준비해야 할 것이 라면과 라면을 끓일 수 있는 냄비일 것입니다. 그런데 이 두 가지 요소는 활용 형태가 다릅니다. 실제로 우리가 먹을 수 있는 것은 라면이고, 냄비는 물, 라면, 파, 달걀 등을 담아 끓이는 역할을 합니다. 이때 냄비 속에서 라면은 변해가는 과정을 거치게 됩니다.

냄비 속에서 라면은 계속 변한다(물, 파, 달걀을 넣고 끓임).
= 변수 속에서 데이터는 계속 변한다(더하거나 빼는 등).

그림 3-1 데이터와 변수의 의미

우리가 먹을 수 있는 라면을 프로그램에서는 데이터라 할 수 있고, 이 데이터를 담는 냄비 역할을 하는 것은 변수라고 할 수 있습니다. 즉 변수는 데이터를 저장하는 그릇이고, 그릇에 저장된 데이터는 계속 변해가는 특성이 있습니다.

2.1.2 변수와 메모리

컴퓨터에서 실행되는 프로그램은 사용자에 의해 프로그래밍 언어로 작성되고, 이 프로그램은 컴파일러에 의해 0, 1로 번역되어 실행됩니다. 이때 0, 1로 번역된 프로그램은 주기억장치main memory에 탑재되어 실행됩니다.

프로그램에서는 데이터를 저장하는 변수를 많이 사용합니다. 변수는 데이터를 저장하는 메모리의 특정 위치에 주어진 이름이라고 볼 수 있습니다. 컴퓨터의 주 메모리는 모두 물리적 주소를 가지고 있는데, 프로그램에서 변수를 사용하지 않는다면 데이터를 메모리에 저장하고 저장된 데이터를 읽어올 때 메모리의 물리적 주소를 사용해야 합니다.

하지만 변수를 사용한다면 우리는 실제 메모리의 물리적 주소를 고려할 필요가 없습니다. 변수를 사용하여 프로그램을 작성하면 변수와 변숫값이 저장된 메모리 주소의 연결이 컴파일러와 시스템 프로그램에 의해 해결되기 때문입니다.

예를 들어 2개의 데이터 10과 20을 더하는 프로그램을 작성한다고 합시다. [그림 3-2]는 프로그램에서 메모리의 물리적 주소를 사용하는 경우와 변수를 사용하는 경우를 비교하여 나타낸 것입니다.

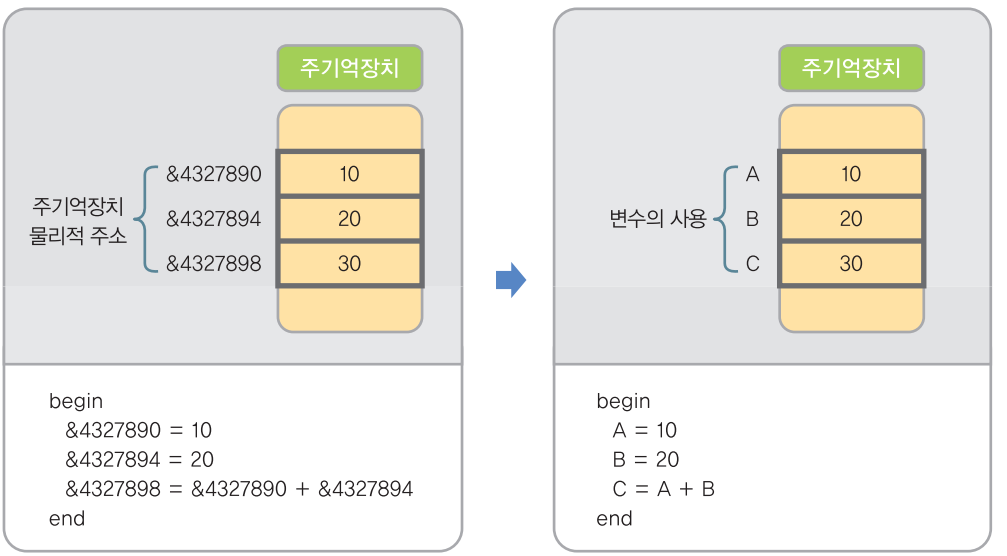


그림 3-2 물리적 주소를 사용하는 경우와 변수를 사용하는 경우

그림에서 보듯이 주기억장치의 물리적 주소를 사용하여 프로그램을 작성하기란 힘든 일입니다. 프로그램이 실행될 때 기억장치의 상세한 배치 구조를 알아내어 프로그래밍하는 것은 거의 불가능하기 때문입니다. 그러므로 모든 프로그램에서는 변수를 사용합니다. 다시 말하자면 변수는 프로그램에서 필요한 데이터를 저장하기 위한 기억 장소에 주어진 이름이라고 할 수 있습니다.

2.1.3 변수의 사용

변수를 사용하려면 먼저 변수의 이름(변수명)을 결정해야 합니다. 변수명을 지정하는 규칙은 1절에서 설명한 '식별자 사용 규칙'과 같습니다. 좋은 변수명이란 변수명만 보고도 그 변수의 의미를

알 수 있는 것입니다. 따라서 가능하면 의미를 나타내는 변수명을 정하는 습관을 들이는 것이 좋습니다.

변수는 프로그래밍을 이루는 핵심으로 프로그램에서 특정 값을 저장하는 역할을 합니다. 프로그램이 진행되면서 변수에 저장된 값이 계속 변화하고, 프로그램이 종료될 때 변수가 가지고 있는 값이 결과로 출력됩니다.

변수에 값을 배정할 때는 =(등호) 기호를 사용합니다. 수학적 의미와는 다르게 =는 변수로 지정된 장소에 해당하는 값을 배정(저장)한다는 의미입니다.

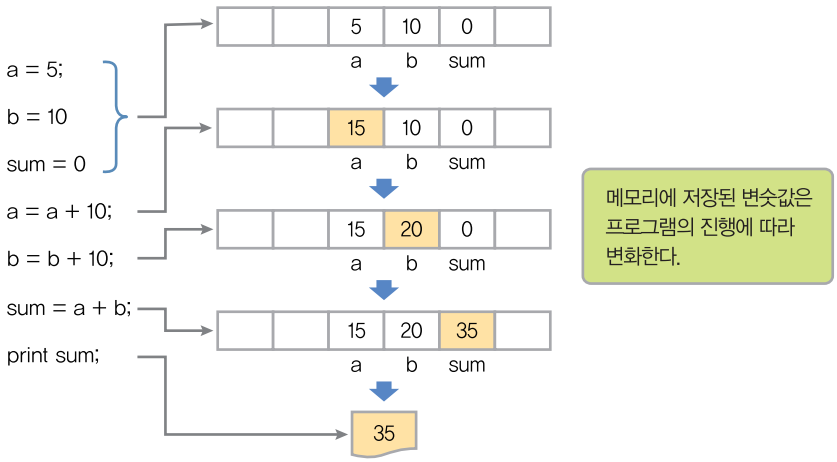


그림 3-3 메모리에 저장된 변숫값의 변화 과정

[그림 3-3]에서 보듯이 변수 a, b, c의 값은 프로그램이 실행되면서 계속 변해가는 과정을 거칩니다.

2.2 상수

한 번 설정되면 프로그램이 수행되는 동안 변하지 않는 상수는 크게 리터럴 상수와 심벌릭 상수로 구분할 수 있습니다.

- **리터럴 상수**literal constant : 프로그램에서 직접 사용하는, 이름이 없는 상수
- **심벌릭 상수**symbolic constant : 이름을 붙여서 변수처럼 사용하는 상수

2.2.1 리터럴 상수

리터럴 상수는 사용자가 입력한 값 그대로를 나타냅니다. 다음은 출력문에 리터럴 상수를 사용한 예입니다. 출력 형식 지정자(%d, %f, %c, %s 등)는 다음 절에서 설명할 것이며 여기에서는 상수의 개념만 이해하면 됩니다.

리터럴 상수로는 10진수, 8진수, 16진수 등의 정수 및 실수, 문자, 문자열 상수를 사용할 수 있습니다. 다음은 출력문에서 리터럴 상수를 직접 사용한 예입니다.

```
01 printf("%d", 300);                  --- 정수형(10진법) 상수 사용
02 printf("%d", 0x300);                --- 정수형(16진법) 상수 사용
03 printf("%d", x300);                 --- 정수형(8진법) 상수 사용
04 printf("%f", 3.14159);              --- 실수형 상수 사용
05 printf("%c", 'A');                  --- 문자형 상수 사용
06 printf("처음 시작하는 C 프로그램입니다");  ┐
07 printf("%s", "OK!! C 프로그래밍");       ┘-- 문자열 상수 사용
```

2.2.2 심벌릭 상수

심벌릭 상수는 변수처럼 상수에 이름을 붙여서 프로그램의 여러 곳에서 사용할 수 있습니다. 단, 프로그램에서 한 번 설정된 상숫값은 바꿀 수 없으며, 바꾸는 경우 컴파일 오류가 발생합니다.

심벌릭 상수를 사용하는 이유는 프로그램의 유지 · 보수(수정)에 편리하고 프로그램 작성 시 오류를 줄이기 위함입니다. 프로그램에서 여러 번 사용되는 상수를 변수화함으로써, 값을 수정해야 할 때 심벌릭 상수만 고치면 간단히 해결할 수 있습니다.

C 언어에서 심벌릭 상수의 이름은 대문자를 사용하는 것이 관례입니다. 소문자로 지정해도 오류가 발생하지는 않지만, 일반 변수와 구분되도록 대문자를 사용하는 것이 좋은 프로그래밍 습관입니다.

C 언어에서 심벌릭 상수를 사용하는 방법은 두 가지가 있습니다. 첫 번째 방법은 const 예약어를 사용하여 상수를 선언하는 것입니다. 다음은 프로그램에서 3.14159라는 상수가 여러 번 사용될 때 심벌릭 상수로 선언하여 사용한 예입니다. const를 이용한 심벌릭 상수는 선언과 동시에 그 값을 초기화해야 합니다.

```
01 ..... // 생략
02 int main()
03 {
04       const double PI = 3.14159;          --- 심벌릭 상수(실수형) 선언과 초기화
05       ......
06       printf("원의 넓이는 %f\n", PI*r*r);  ┐
07       printf("원의 둘레는 %f\n", 2*PI*r);  ┘-- PI(심벌릭 상수) 사용
08       ......
09       PI = 3.14;                          --- 상숫값의 변화를 시도하면 오류 발생
10 ......
```

두 번째 방법은 헤더 파일 부분에 #define문을 사용하여 심벌릭 상수를 정의하는 것입니다.

#define문으로 정의된 심벌릭 상수를 프로그램에서 사용하면 컴파일을 수행할 때 전처리기에 의해 정의된 값으로 대치됩니다.

```
01 ......
02 #define PI 3.14159                  --- 심벌릭 상수 선언과 초기화
03 ......
04 int main()
05 {
06       ......
07       printf("원의 넓이는 %f\n", PI*r*r);   ┐ PI(심벌릭 상수) 사용. 전처리기에 의해 PI가
08       printf("원의 둘레는 %f\n", 2*PI*r);   ┘ 3.14159로 대치됨
09       ......
```

03 자료형

C 언어에서 변수를 선언할 때는 변수의 형type을 지정해야 하는데 이를 자료형data type이라고 합니다. 이 절에서는 자료형의 의미와 C 언어에서 제공되는 자료형을 예제를 통해 자세히 살펴보겠습니다.

3.1 자료형의 의미

라면과 냄비의 비유를 다시 생각해보면, 냄비는 다양한 종류가 있으며 냄비 외에도 솥, 프라이팬 등의 용기가 있습니다. 그런데 냄비에는 라면을 담고 솥에는 국을 담고 프라이팬에는 생선을 담듯이 프로그램에서도 변수의 형(자료형)에 따라 넣을 수 있는 데이터(자료)의 형태가 달라집니다. 이때 적절한 용기(변수)에 적합한 재료(데이터)를 담아야 편리하게 요리할 수 있습니다.

그림 3-4 자료형과 자료

C 언어는 다양한 자료형을 제공하는데, 우리가 프로그램을 작성할 때는 용도에 가장 적합한 자료형으로 변수를 선언하고 그 자료형에 적합한 데이터를 넣어서 사용하면 됩니다. C 언어에는 정수형integer type과 실수형floating point type의 두 가지 자료형이 있으며, 다시 정수형은 네 가지 자료형으로, 실수형은 세 가지 자료형으로 구분됩니다.

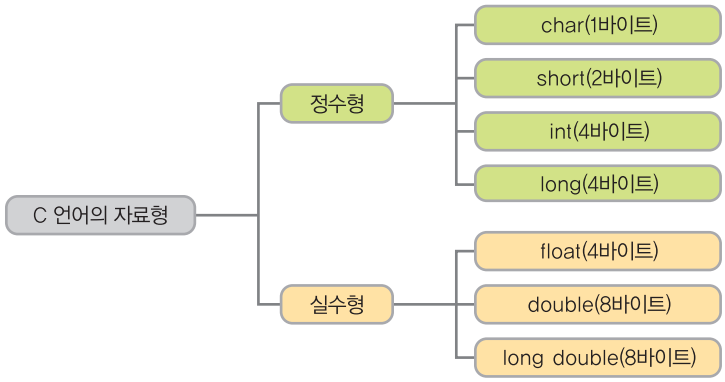


그림 3-5 C 언어의 자료형

3.2 변수의 선언과 자료형의 크기

3.2.1 변수의 선언

C 언어는 프로그램에서 변수를 사용할 경우 반드시 미리 선언을 해야 합니다. 선언하지 않은 변수를 사용하면 오류가 발생합니다. 변수를 선언할 때는 그 변수의 자료형을 지정해야 하며, 변수는 자료형과 변수명만 선언할 수도 있고 초기 값까지 설정하여 선언할 수도 있습니다.

다음은 변수를 선언한 예입니다.

```
01 int num;              --- int형 변수 num 선언
02 short grade;          --- short형 변수 grade 선언
03 char flag = 'y';      --- char형 변수 flag를 선언하고 'y'로 초기화
04 double rate;          --- double형 변수 rate 선언
05 float rmax = 0.3;     --- float형 변수 rmax를 선언하고 '0.3'으로 초기화
```

C 언어에서는 변수를 사용하기 전에 선언하지 않으면 다음과 같이 오류가 발생합니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05       int max = 100;
06       printf("%d\n",max);
07       printf("%d\n",min); --- 선언 전에 변수를 사용하여 오류 발생
08       int min = 1;
09 }
```

또한 변수를 초기화하지 않고 사용하는 경우에도 다음과 같이 오류가 발생합니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05       int max;
06       printf("%d\n",max);  --- 변수가 초기화되지 않아 실행 시간 오류 발생
07 }
```

3.2.2 자료형의 크기

C 언어에서는 sizeof() 연산자를 사용하여 자료형이나 변수에 배정된 기억 장소의 크기를 구할 수 있습니다. sizeof() 연산자는 해당되는 변수나 형이 기억 장소에서 차지하는 바이트 수를 반환합니다.

예제 3-2 sizeof1.c

다음 프로그램은 sizeof() 연산자를 사용하여 메모리에서 변수나 형이 차지하는 바이트 수를 출력합니다. sizeof() 연산자를 사용하여 직접 지정하는 리터럴의 바이트 수도 출력할 수 있습니다.

```
01 #include <stdio.h>
02 #define NAT "대한민국"
03
04 int main()
05 {
06      short s;
07      int num=100;
08      long lnumber;
09      char sex;          --- 일곱 가지 자료형 변수 선언
10      float f= 3.14;
11      double d;
12      long double ld;
13
14      printf("short형의 크기 sizeof(short) : %d   sizeof(변수명) :
   %d\n",sizeof(short), sizeof(s));
15      printf("int형의 크기 sizeof(int) : %d   sizeof(변수명) :
   %d\n",sizeof(int), sizeof(num));
16      printf("long형의 크기 sizeof(long) : %d   sizeof(변수명) :
   %d\n",sizeof(lnumber), sizeof(lnumber));
17      printf("char형의 크기 sizeof(char) : %d   sizeof(변수명) :
   %d\n",sizeof(char), sizeof(sex));
18      printf("float형의 크기 sizeof(float) : %d   sizeof(변수명) :
   %d\n",sizeof(float), sizeof(f));
```

--- 일곱 가지 자료형과 변수의 크기 출력

```
19      printf("double형의 크기 sizeof(double) : %d  sizeof(변수명) :
    %d\n",sizeof(double), sizeof(d));
20      printf("long double형의 크기 sizeof(long double) : %d  sizeof(변수명) :
    %d\n",sizeof(long double), sizeof(ld));
21      printf("문자 리터럴 \"대한민국\"의 크기 : %d\n",sizeof(NAT));
        └── 문자 리터럴 "대한민국"의 크기 출력
22      printf("문자 리터럴 \"university\"의 크기 : %d\n",sizeof("university"));
        └── 문자 리터럴 "university"의 크기 출력
23  }
```

프로그램 설명

06~12 : 일곱 가지 자료형의 변수를 선언하고 일부는 초기화를 수행했습니다.

14~20 : sizeof() 연산자를 사용하여 자료형의 크기를 출력했습니다. sizeof() 연산자는 자료형을 직접 지정하거나 변수를 지정하여 크기를 출력할 수 있습니다.

21 : 전처리기에 의해 처리되는 문자 리터럴의 크기를 출력했습니다. 크기가 9인 이유는 C 언어에서 하나의 한글 문자는 2바이트로 표시되며 모든 문자열의 끝에 종료를 나타내는 종료 문자(\0)가 자동으로 추가되기 때문입니다.

22 : 직접 문자 리터럴을 지정하여 크기를 출력했습니다. 하나의 영문자는 1바이트에 저장됩니다.

실행 결과

```
short형의 크기 sizeof(short) : 2   sizeof(변수명) : 2
int형의 크기 sizeof(int) : 4   sizeof(변수명) : 4
long형의 크기 sizeof(long) : 4   sizeof(변수명) : 4
char형의 크기 sizeof(char) : 1   sizeof(변수명) : 1
float형의 크기 sizeof(float) : 4   sizeof(변수명) : 4
double형의 크기 sizeof(double) : 8   sizeof(변수명) : 8
long double형의 크기 sizeof(long double) : 8   sizeof(변수명) : 8
문자 리터럴 "대한민국"의 크기 : 9
문자 리터럴 "university"의 크기 : 11
```

3.3 정수형

C 언어의 정수형에는 네 가지 형태가 있습니다. 그중에서 char형은 문자를 나타내지만 정수형으로 사용됩니다. [그림 3-6]은 C 언어의 정수 자료형이 나타낼 수 있는 수의 범위를 나타낸 것입니다. 자료형을 구성하는 비트 수가 많을수록 큰 수를 나타낼 수 있습니다.

C 언어의 수치 정수형은 첫 번째 비트를 부호 비트로 사용합니다. 그러므로 특정 비트 수가 나타낼 수 있는 정수형의 표현 범위는 -2^{n-1}에서 $+2^{n-1}-1$입니다.

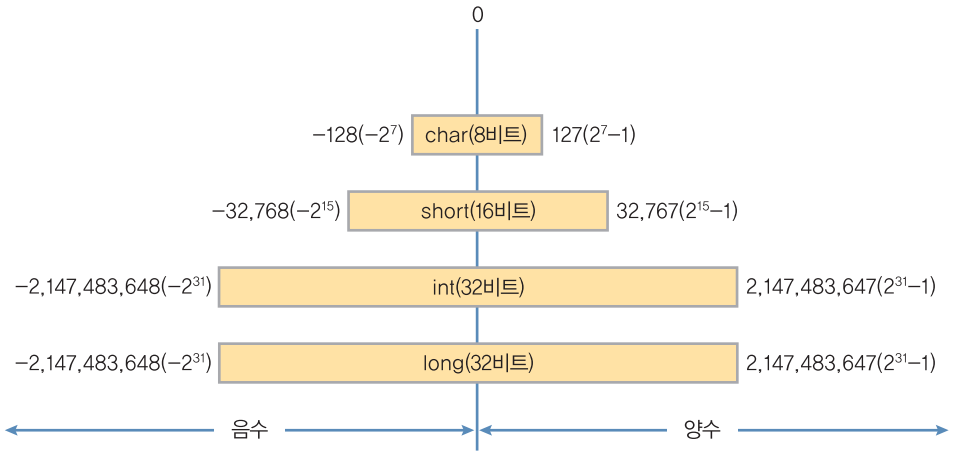


그림 3-6 정수형의 표현 범위

자료형 중에서 int형은 4바이트로 정의되었지만 int의 바이트 수는 CPU가 처리하는 데이터의 크기와 관련이 있습니다. 현재 대부분의 CPU는 32비트로 데이터를 처리하기 때문에 int가 long과 같이 4바이트로 정의되었습니다.

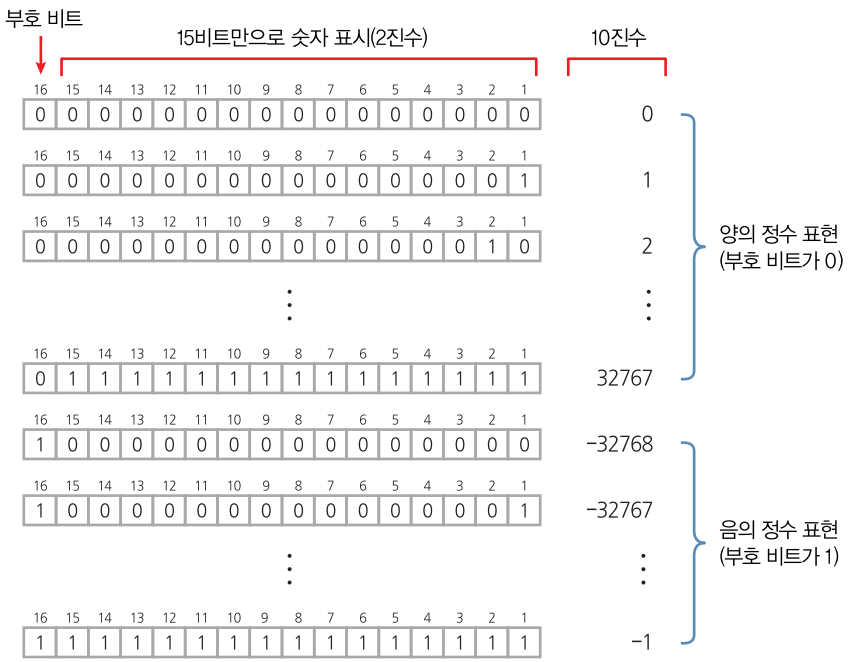


그림 3-7 16비트 short형의 표현 범위

C 언어에서는 음수를 나타내기 위해 각 자료형의 최상위 비트를 부호 비트로 하는 2의 보수법2's complement을 사용하고 있습니다. [그림 3-7]은 short형이 나타낼 수 있는 수의 범위를 나타낸 것으로, short형은 모두 16비트로 구성되며 최상위 1비트는 부호를 나타내는 데 사용됩니다.

C 언어는 양의 정수 표현을 넓히기 위한 방법으로 정수형에 대해 unsigned형을 제공하고 있습니다. unsigned형은 정수형 표현에서 음수를 표현하지 않고 모두 양수로 표현하는 방법으로, 음수를 표현하지 않음으로써 2배의 양수 표현 범위를 가지게 됩니다. unsigned 정수 자료형의 표현 범위를 [그림 3-8]에 나타냈습니다.

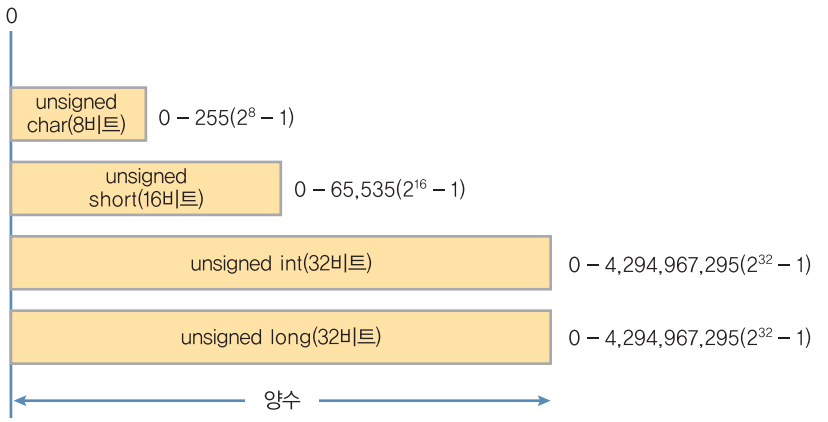


그림 3-8 unsigned 정수 자료형의 표현 범위

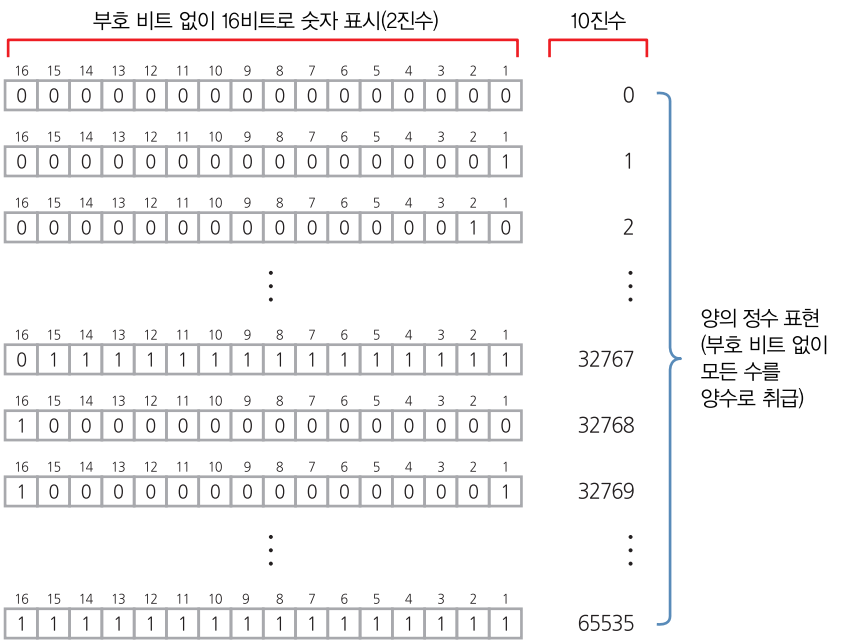


그림 3-9 16비트 unsigned short의 표현 범위

3.3.1 수치 정수형

C 언어의 수치 정수형은 short, int, long을 제공합니다. C 언어에는 각 자료형의 최댓값과 최솟값을 심벌릭 상수로 제공하는 라이브러리 파일(limits.h)이 있는데, 프로그램에서 이 라이브러리 파일을 포함하면 [표 3-4]와 같은 심벌릭 상수를 직접 사용할 수 있습니다.

표 3-4 〈limits.h〉에 정의된 정수형의 심벌릭 상수

short	SHRT_MAX SHRT_MIN USHRT_MAX
int	INT_MAX INT_MIN UINT_MAX
long	LONG_MAX LONG_MIN ULONG_MAX

예제 3-3 numeric1.c

다음은 〈limits.h〉에서 제공되는 상수를 이용하여 정수형 값의 최솟값과 최댓값을 구하는 프로그램입니다.

```
01 #include <stdioh>
02 #include <limits.h>  --- 자료형의 심벌릭 상수를 가진 헤더 파일 포함
03
04 int main()
05 {
06      printf("short형의 최댓값 : %d, 최솟값 : %d\n",SHRT_MAX, SHRT_MIN);
        └── short형의 최댓값, 최솟값 출력
07      printf("unsigned short형의 최댓값 : %d\n",USHRT_MAX);
        └── unsigned short형의 최댓값 출력
08      printf("int형의 최댓값 : %d, 최솟값 : %d\n",INT_MAX, INT_MIN);
        └── int형의 최댓값, 최솟값 출력
09      printf("unsigned int형의 최댓값(%%d 형식) : %d\n",UINT_MAX);
        └── %d 형식으로 unsigned int형의 최댓값 출력
10      printf("unsigned int형의 최댓값(%%u 형식) : %u\n",UINT_MAX);
        └── %u 형식으로 unsigned int형의 최댓값 출력
11      printf("unsigned int형의 최댓값(%%0x 형식) : %0x\n",UINT_MAX);
        └── %0x 형식으로 unsigned int형의 최댓값 출력
12      printf("long형의 최댓값 : %d, 최솟값 : %d\n", LONG_MAX, LONG_MIN);
        └── long형의 최댓값, 최솟값 출력
13      printf("unsigned long형의 최댓값 : %u\n",ULONG_MAX);
        └── unsigned long형의 최댓값 출력
14 }
```

프로그램 설명

02 : 〈limits.h〉 파일은 각 자료형의 최솟값과 최댓값을 심벌릭 상수로 제공합니다. 심벌릭 상수를 사용하기 위해 헤더 파일을 포함합니다.

06 : short형의 최댓값과 최솟값을 출력합니다.

07 : unsigned short형의 최댓값을 출력합니다.

08 : int형의 최댓값과 최솟값을 출력합니다.

09 : unsigned int형의 최댓값을 %d 형식 지정자를 사용하여 출력합니다. -1이 출력된 이유는 %d 형식 지정자가 숫자를 부호가 있는 숫자로 취급하기 때문입니다. 즉 unsigned int형의 최댓값은 모든 비트가 1이므로 음수로 취급하여 -1이 출력됩니다.

10 : unsigned int형의 최댓값을 %u 형식 지정자를 사용하여 출력합니다. %u 형식 지정자는 숫자를 부호가 없는 숫자로 취급하므로 정상적인 최댓값이 출력됩니다.

11 : unsigned int형의 최댓값을 %0x 형식 지정자를 사용하여 출력합니다. 16진수로 최댓값이 출력됩니다.

12 : long형의 최댓값과 최솟값을 출력합니다.

13 : unsigned long형의 최댓값을 출력합니다.

실행 결과

```
short형의 최댓값 : 32767, 최솟값 : -32768
unsigned short형의 최댓값 : 65535
int형의 최댓값 : 2147483647, 최솟값 : -2147483648
unsigned int형의 최댓값(%d 형식) : -1
unsigned int형의 최댓값(%u 형식) : 4294967295
unsigned int형의 최댓값(%0x 형식) : ffffffff
long형의 최댓값 : 2147483647, 최솟값 : -2147483648
unsigned long형의 최댓값 : 4294967295
```

C 언어의 정수형은 부호 비트를 사용하여 수를 표현하므로 각 형에 따라 표현할 수 있는 최댓값과 최솟값이 결정되어 있습니다. [예제 3-4]는 정수가 표현 범위를 벗어났을 때 발생하는 현상에 대한 예입니다. 정수의 표현 범위를 벗어나면 오버플로overflow 또는 언더플로underflow 현상이 나타납니다.

여기서 잠깐 **정수에서의 음수 표현법**

C 언어의 정수형에서 음수의 표현은 부호 비트를 사용한 2의 보수법을 사용합니다. 2의 보수는 음수에 해당하는 양수에 대해 모든 비트를 반전한(1의 보수) 결과에 1을 더하여 구할 수 있습니다. 반대로, 어떤 수의 부호 비트가 1인 경우 그 수를 구하기 위해 다시 2의 보수법을 적용하여 음숫값을 결정하게 됩니다.

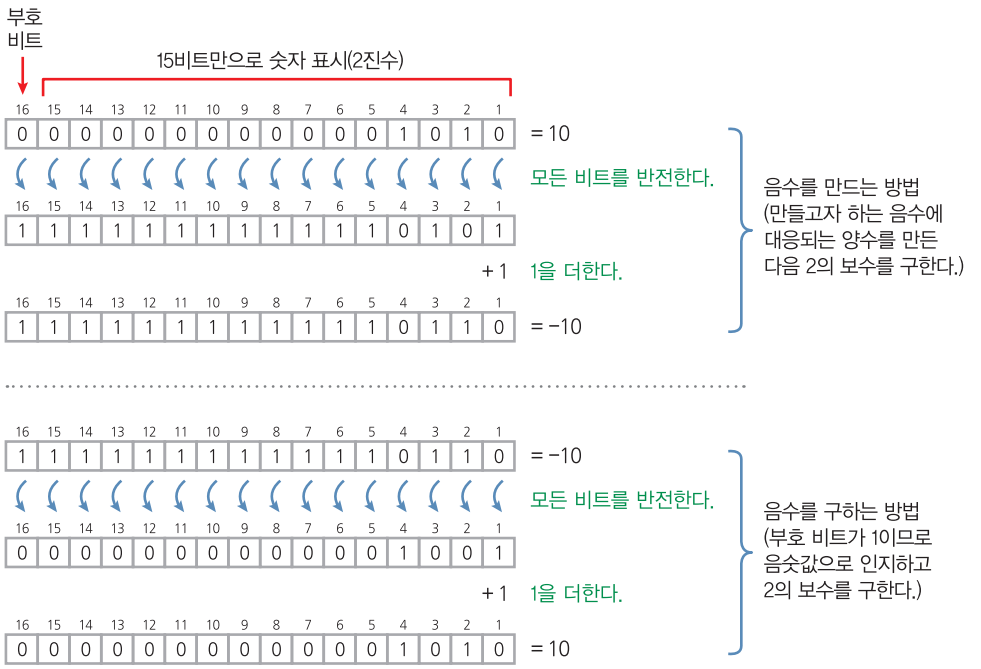


그림 3-10 음수를 만드는 방법과 음숫값을 구하는 방법

예제 3-4 numeric2.c

다음은 정수의 표현 범위를 벗어난 접근을 나타내는 프로그램으로 오버플로와 언더플로의 예를 보여줍니다.

```
01 #include <stdio.h>
02 #include <limits.h>
03
04 int main()
05 {
06      short s1 = SHRT_MIN;
07      short s2 = SHRT_MAX;
08      unsigned short us1 = 0;
09      unsigned short us2 = USHRT_MAX;
10      printf("short형 최솟값 : %d\n",s1);
```

-- 각 형에 따라 최댓값과 최솟값 설정

```
11      printf("short형 최댓값 : %d\n",s2);
12      printf("unsigned short형 최솟값 : %u\n",us1);
13      printf("unsigned short형 최댓값 : %u\n",us2);
14      printf("=======================================\n");
15      s1 = SHRT_MIN - 1;
16      s2 = SHRT_MAX + 1;
17      us1 = 0 - 1;
18      us2 = USHRT_MAX + 1;
19      printf("short형 최솟값 -1 : %d\n",s1);
20      printf("short형 최댓값 +1 : %d\n",s2);
21      printf("unsigned short형 최솟값 -1 : %u\n",us1);
22      printf("unsigned short형 최댓값 +1 : %u\n",us2);
23  }
```

최솟값은 1을 빼고 최댓값은 1을 더하여 저장

언더플로, 오버플로가 발생한 값 출력

프로그램 설명

06~09 : short형과 unsigned short형의 최솟값과 최댓값을 설정했습니다.

10~13 : 최솟값과 최댓값을 출력합니다.

15~18 : short형 최솟값과 unsigned short형 최솟값은 1을 빼서 다시 저장하고, 최댓값은 1을 더하여 변수에 다시 저장합니다. 이 경우 언더플로와 오버플로가 발생합니다.

19~22 : 언더플로와 오버플로가 발생한 값을 출력합니다.

실행 결과

```
short형 최솟값 : -32768
short형 최댓값 : 32767
unsigned short형 최솟값 : 0
unsigned short형 최댓값 : 65535

short형 최솟값 -1 : 32767
short형 최댓값 +1 : -32768
unsigned short형 최솟값 -1 : 65535
unsigned short형 최댓값 +1 : 0
```

정수 자료형의 변수에 배정된 값이 표현 범위를 벗어나는 경우에는 원치 않는 결과가 나타납니다. 이런 경우 C 언어에서는 구문 오류를 발생시키지 않기 때문에 프로그램을 작성할 때 주의해야 합니다. [그림 3-11]은 short형에서 범위를 벗어났을 때 나타나는 결과를 표현한 것입니다.

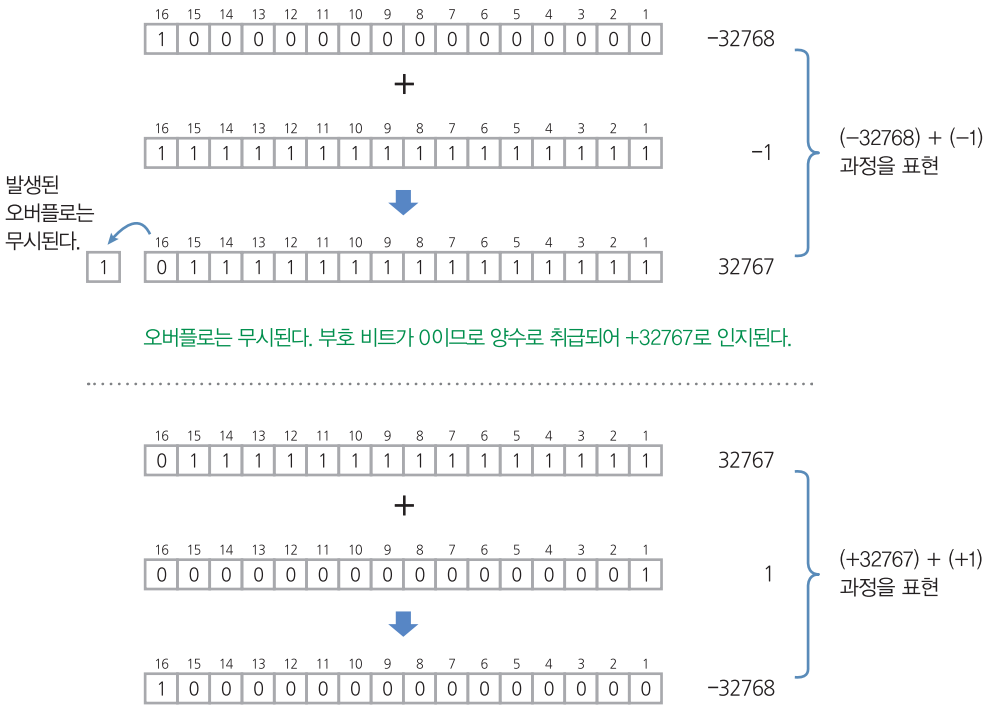


그림 3-11 short형에서 저장 범위를 벗어난 연산의 결과

3.3.2 문자 정수형

컴퓨터는 2진법을 사용하여 숫자를 표현하는데, 예를 들어 숫자 4를 2진수로 나타내면 '0100'(4비트 사용)이 됩니다. 컴퓨터는 문자도 2진법을 사용하여 표현하는데, 문자를 인식하려면 미리 약속된 코드가 필요합니다. 예를 들어 문자 'A'는 '01000001'(8비트 사용)을 사용하자는 약속(규약)이 있어야 합니다. 이러한 약속에 따라 컴퓨터는 'A'를 저장할 때 항상 '01000001'을 사용하고, 저장된 2진 데이터를 표현할 때도 '01000001'을 'A'로 나타냅니다.

이러한 약속을 미국표준협회ANSI에서 아스키ASCII, American Standards Committee for Information Interchange 코드라는 이름으로 제정했으며, 이 코드는 현재까지 표준으로 사용되고 있습니다. 아스키코드는 8비트 중에서 7비트만을 사용해서 128개의 문자를 표현하며, 맨 앞의 1비트는 코드를 전송할 때 발생하는 오류를 검사하기 위해 사용합니다. 128개의 문자로 구성된 아스키코드 중에서 0~31번과 127번은 인쇄할 수 없는 문자이고 32~126번은 인쇄할 수 있는 문자로 영문자(대 · 소문자), 숫자(0~9), 특수문자를 나타냅니다.

8개의 비트로 구성된 아스키코드에는 한글이 포함되지 않았으며, 안타깝게도 C 언어의 char형에서는 한글을 사용할 수 없습니다. C 언어에서 한글을 사용하려면 문자열을 사용해야 합니다.

표 3-5 아스키코드

2진법	10진법	문자
000 0000	000	NUL(Null 공백 문자)
000 0001	001	SOH(Start of Header)
000 0010	002	STX(Start of Text)
000 0011	003	ETX(End of Text)
000 0100	004	EOT(End of Transmission)
000 0101	005	ENQ(Enquiry 응답 요구)
000 0110	006	ACK(Acknowledgment)
000 0111	007	BEL(Bell 경고음)
000 1000	008	BS(Backspace)
000 1001	009	HT(Horizontal Tab)
000 1010	010	LF(Line Feed)
000 1011	011	VT(Vertical Tab)
000 1100	012	FF(Form Feed)
000 1101	013	CR(Carriage Return)
000 1110	014	SO(Shift Out)
000 1111	015	SI(Shift In)
001 0000	016	DLE(Data Link Escape)
001 0001	017	DC1(Device Control 1)
001 0010	018	DC2(Device Control 2)
001 0011	019	DC3(Device Control 3)
001 0100	020	DC4(Device Control 4)
001 0101	021	NAK(Negative Acknowledgement)
001 0110	022	SYN(Synchronous idle)
001 0111	023	ETB(End of Transmission Block)
001 1000	024	CAN(Cancel)
001 1001	025	EM(End of Medium)
001 1010	026	SUB(Substitute)
001 1011	027	ESC(Escape)
001 1100	028	FS(File Separator)
001 1101	029	GS(Group Separator)
001 1110	030	RS(Record Separator)
001 1111	031	US(Unit Separator)
010 0000	032	$_{sp}$
010 0001	033	!
010 0010	034	"
010 0011	035	#
010 0100	036	$
010 0101	037	%
010 0110	038	&
010 0111	039	'
010 1000	040	(
010 1001	041	)

표 3-5 (계속)

2진법	10진법	문자	2진법	10진법	문자
010 1010	042	*	101 0101	085	U
010 1011	043	+	101 0110	086	V
010 1100	044	,	101 0111	087	W
010 1101	045	–	101 1000	088	X
010 1110	046	.	101 1001	089	Y
010 1111	047	/	101 1010	090	Z
011 0000	048	0	101 1011	091	[
011 0001	049	1	101 1100	092	₩
011 0010	050	2	101 1101	093	]
011 0011	051	3	101 1110	094	^
011 0100	052	4	101 1111	095	_
011 0101	053	5	110 0000	096	`
011 0110	054	6	110 0001	097	a
011 0111	055	7	110 0010	098	b
011 1000	056	8	110 0011	099	c
011 1001	057	9	110 0100	100	d
011 1010	058	:	110 0101	101	e
011 1011	059	;	110 0110	102	f
011 1100	060	<	110 0111	103	g
011 1101	061	=	110 1000	104	h
011 1110	062	>	110 1001	105	i
011 1111	063	?	110 1010	106	j
100 0000	064	@	110 1011	107	k
100 0001	065	A	110 1100	108	l
100 0010	066	B	110 1101	109	m
100 0011	067	C	110 1110	110	n
100 0100	068	D	110 1111	111	o
100 0101	069	E	111 0000	112	p
100 0110	070	F	111 0001	113	q
100 0111	071	G	111 0010	114	r
100 1000	072	H	111 0011	115	s
100 1001	073	I	111 0100	116	t
100 1010	074	J	111 0101	117	u
100 1011	075	K	111 0110	118	v
100 1100	076	L	111 0111	119	w
100 1101	077	M	111 1000	120	x
100 1110	078	N	111 1001	121	y
100 1111	079	O	111 1010	122	z
101 0000	080	P	111 1011	123	{
101 0001	081	Q	111 1100	124	|
101 0010	082	R	111 1101	125	}
101 0011	083	S	111 1110	126	~
101 0100	084	T	111 1111	127	DEL(Delete)

문자형(char)을 문자 정수형으로 표현하는 이유는 문자형 변수가 가진 값이 정수라는 의미이기 때문입니다. 즉 문자형을 정수형처럼 사용할 수도 있으며, 또 unsigned char형도 제공하고 있습니다.

그리고 정수 자료형과 같이 문자 자료형인 char형도 최댓값과 최솟값을 〈limits.h〉 파일에서 심벌릭 상수(CHAR_MIN, CHAR_MAX, UCHAR_MAX)로 제공합니다. 문자형 변수에 문자 값을 저장하려면 반드시 작은따옴표를 사용해야 합니다. [예제 3-5]를 통해 심벌릭 상수의 사용법과 문자형 변수에 문자 값을 저장하는 방법을 살펴봅시다.

예제 3-5 chartest1.c

다음은 char형의 값에 대한 최솟값과 최댓값을 나타내는 프로그램으로 char형이 정수형으로 표현되는 예도 보여주고 있습니다.

```
01 #include <stdio.h>
02 #include <limits.h>
03
04 int main()
05 {
06      char c1 = CHAR_MIN;                          ┐ char형과 unsigned char형의
07      char c2 = CHAR_MAX;                          ├ 최솟값, 최댓값 설정
08      unsigned char c3 = UCHAR_MAX;                ┘
09      printf("char형 최솟값 : %d\n",c1);           ┐
10      printf("char형 최댓값 : %d\n",c2);           ├ char형과 unsigned char형의 최솟값, 최댓값 출력
11      printf("unsigned char형 최댓값 : %u\n",c3);  ┘
12      printf("=======================================\n");
13      c1 = 65;                                     ┐
14      c2 = 66;                                     ├ char형의 변수에 정숫값 설정
15      c3 = c2+1;                                   ┘
16      printf("c1값 : %d, c2값 : %d, c3값 : %d\n", c1, c2, c3);
        └── char형의 변숫값을 %d 형식 지정자를 사용하여 출력
17      printf("c1값 : %c, c2값 : %c, c3값 : %c\n", c1, c2, c3);
        └── char형의 변숫값을 %c 형식 지정자를 사용하여 출력
18      printf("=======================================\n");
19      c1 = 'x';                                    ┐
20      c2 = 'y';                                    ├ char형의 변수에 문자 값 설정
21      c3 = 'z';                                    ┘
22      printf("c1값 : %d, c2값 : %d, c3값 : %d\n", c1, c2, c3);
        └── char형의 변숫값을 %d 형식 지정자를 사용하여 출력
23      printf("c1값 : %c, c2값 : %c, c3값 : %c\n", c1, c2, c3);
        └── char형의 변숫값을 %c 형식 지정자를 사용하여 출력
24 }
```

프로그램 설명

06~08 : char형과 unsigned char형의 최솟값과 최댓값을 설정했습니다.

09~11 : char형의 unsigned char형의 최솟값과 최댓값을 출력합니다.

13~15 : char형의 변수에 정숫값을 설정했습니다. char형의 변수는 정수형으로서 어떠한 정수라도 배정될 수 있습니다.

16~17 : char형의 변숫값을 %d와 %c 형식 지정자를 사용하여 출력합니다. 결과로 정숫값과 문자 값을 출력합니다.

19~21 : char형의 변수에 문자 값을 설정했습니다. 하나의 문자를 나타내기 위해 작은따옴표를 사용했습니다.

22~23 : char형의 변숫값을 %d와 %c 형식 지정자를 사용하여 출력합니다. 결과로 정숫값과 문자 값을 출력합니다.

실행 결과

```
char형 최솟값 : -128
char형 최댓값 : 127
unsigned char형 최댓값 : 255
================================
c1값 : 65, c2값 : 66, c3값 : 67
c1값 : A, c2값 : B, c3값 : C
================================
c1값 : 120, c2값 : 121, c3값 : 122
c1값 : x, c2값 : y, c3값 : z
```

예제 3-6 chartest2.c

다음은 문자형 변수와 정수형 변수의 값을 키보드로 입력받아 정수와 문자로 출력하는 프로그램입니다. 수치 정수형과 문자 정수형은 같은 정수 범위 내에서 호환되어 사용될 수 있습니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      char c1;
06      int ic1;
07
```

-- char형과 int형의 변수 선언

```
08      printf("문자입력(char형 변수 저장) : ");
09      scanf("%c", &c1);   --- %c 형식 지정자를 사용하여 값을 입력
10      printf("char 변수에 저장된 값 : %%d_형식 %d, %%c_형식 %c\n", c1, c1);
        └-- char형의 변숫값을 %d와 %c 형식 지정자를 사용하여 출력
11      printf("숫자입력(int형 변수 저장) : ");
12      scanf("%d", &ic1);  --- %d 형식 지정자를 사용하여 값을 입력
13      printf("int 변수에 저장된 값 : %%d_형식 %d, %%c_형식 %c\n", ic1, ic1);
        └-- int형의 변숫값을 %d와 %c 형식 지정자를 사용하여 출력
14  }
```

프로그램 설명

05~06 : char형과 int형의 변수를 선언했습니다.

09 : %c 형식 지정자를 사용하여 값을 입력받아 char형 변수에 저장합니다.

10 : char형의 변숫값을 %d 형식 지정자를 사용하여 정숫값으로 출력하고, %c 형식 지정자를 사용하여 문자 값으로 출력합니다.

12 : %d 형식 지정자를 사용하여 값을 입력받아 int형 변수에 저장합니다.

13 : int형의 변숫값을 %d 형식 지정자를 사용하여 정숫값으로 출력하고, %c 형식 지정자를 사용하여 문자 값으로 출력합니다.

실행 결과

```
문자입력(char형 변수 저장) : m
char 변수에 저장된 값 : %d_형식 109, %c_형식 m
숫자입력(int형 변수 저장) : 110
int 변수에 저장된 값 : %d_형식 110, %c_형식 n
```

3.4 실수형

C 언어에서 실수형은 소수점 이하의 부분을 가진 실수를 저장할 수 있는 변수를 의미합니다. C 언어에서는 실수형으로 float형(4바이트), double형(8바이트), long double형(8바이트)을 제공하고 있습니다. long double형은 더 정밀한 실수를 표현하기 위해 도입된 것이지만 배정되는 메모리는 컴파일러에 따라 다릅니다. 비주얼 C 컴파일러의 경우 long double형에도 double형과 같은 8바이트의 메모리를 배정하고 있습니다.

실수형의 표현은 고정 소수점fixed point 표현과 부동 소수점floating point 표현으로 나눌 수 있습니다. 부동 소수점 표현은 소수점 이하를 나타내는 가수mantissa 부분과 승수를 나타내는 지수exponential

부분으로 구성됩니다.

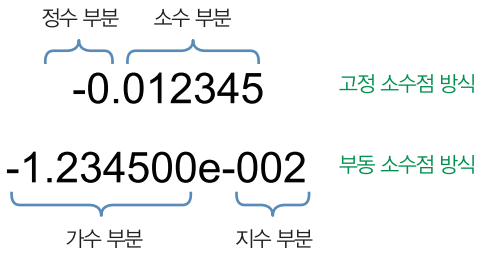


그림 3-12 실수형 표현 방식

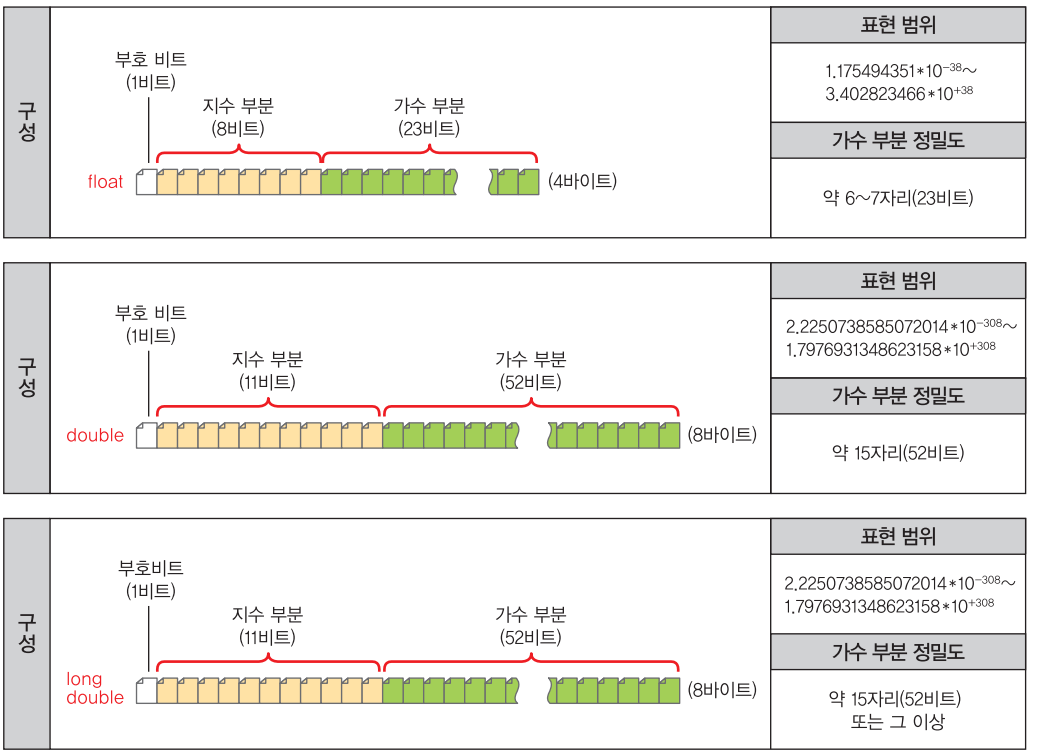


그림 3-13 float형, double형, long double형의 표현 범위와 가수 부분 정밀도

표 3-6 〈float.h〉에 정의된 실수형의 심벌릭 상수

float	FLT_MAX FLT_MIN
double	DBL_MAX DBL_MIN
long double	LDBL_MAX LDBL_MIN

C 언어의 실수형은 float형, double형, long double형을 제공합니다. C 언어의 실수형에는 정수형과 마찬가지로 각 자료형의 최댓값과 최솟값을 심벌릭 상수로 제공하는 라이브러리 파일(float.h)이 있습니다. 프로그램에서 이 라이브러리 파일을 포함하면 [표 3-6]과 같은 심벌릭 상수를 직접 사용할 수 있습니다.

예제 3-7 doubletest1.c

다음은 〈float.h〉에서 제공되는 상수를 이용하여 실수형 값의 최솟값과 최댓값을 구하는 프로그램입니다.

```
01 #include <stdio.h>
02 #include <float.h>  --- 실수 자료형의 심벌릭 상수를 가진 헤더 파일 포함
03
04 int main()
05 {
06     printf("float 형의 최솟값 : %e, 최댓값 : %e\n",FLT_MIN,
   FLT_MAX);
07     printf("double 형의 최솟값 : %e, 최댓값 : %e\n",DBL_MIN,
   DBL_MAX);
08     printf("long double 형의 최솟값 : %e, 최댓값 : %e\n",
   LDBL_MIN, LDBL_MAX);
09 }
```

float형, double형, long double형의 최솟값과 최댓값을 %e 형식 지정자를 사용하여 출력

프로그램 설명

02 : 〈float.h〉 파일은 실수 자료형의 최솟값과 최댓값을 심벌릭 상수로 제공합니다. 심벌릭 상수를 사용하기 위해 헤더 파일을 포함했습니다.

06~08 : float형, double형, long double형의 최솟값과 최댓값을 출력합니다. 출력 형식 지정자는 %e를 사용해야 합니다.

실행 결과

```
float형의 최솟값 : 1.175494e-038, 최댓값 : 3.402823e+038
double형의 최솟값 : 2.225074e-308, 최댓값 : 1.797693e+308
long double형의 최솟값 : 2.225074e-308, 최댓값 : 1.797693e+308
```

C 언어에서 실수형은 저장할 수 있는 크기에 따라 float형, double형, long double형으로 구분됩니다. C 언어에서는 실수형의 리터럴을 묵시적으로 double형으로 처리합니다. float형의 실수를 사용하려면 반드시 실수 리터럴 값의 끝에 f(또는 F)를 붙여야 하며, 그렇지 않을 경우 경고warining 오류가 발생합니다.

일반적으로 C 프로그램에서 실수를 사용할 때는 double형이 좋습니다. 유효 자릿수가 float형은 약 7자리이지만 double형은 15자리이기 때문입니다. 다음 예에서 그 차이점을 살펴볼 수 있습니다.

```
01 float f = 3.14 ;  --- 컴파일은 되지만 경고 오류 발생
02 float f = 3.14f ;  --- 정상적으로 처리
03 printf("%f\n",0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+0.3f+
   0.3f+0.3f+0.3f+0.3f+0.3f+0.3f);
   └── 실수 0.3을 float로 지정하여 20번 더한 결과로 6.000001 출력(오차 발생)
04 printf("%f\n",0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+0.3+
   0.3+0.3);  --- 실수 0.3(묵시적으로 double)을 20번 더한 결과로 6.000000 출력
```

예제 3-8 doubletest2.c

다음은 float형과 double형의 유효 자릿수를 나타내는 프로그램입니다. 정확한 실수 연산을 위해서는 double형을 사용하는 것이 좋습니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05     float f = 0.12345678901234567890f;  --- float형 변수에 유효 자릿수가 19자리인 실수 저장
06     double d = 0.12345678901234567890;  --- double형 변수에 유효 자릿수가 19자리인 실수 저장
07     float f1 = 0.1e+40;  --- float형 변수에 지수가 40승인 실수 저장. 경고 오류 발생
08     float f2 = 0.1e-40;  --- float형 변수에 지수가 -40승인 실수 저장. 경고 오류 발생
09
10     printf("%.12f\n",f);  --- 소수점 이하 12자리까지 출력
11     printf("%.20f\n",d);  --- double형 변숫값을 소수점 이하 20자리까지 출력
12     printf("%.12e\n",f);  ┐
13     printf("%.12e\n",d);  ├── 소수점 이하 12자리까지 출력
14     printf("%.12f\n",f1);  ┤
15     printf("%.12f\n",f2);  ┘
16 }
```

프로그램 설명

05 : float형 변수에 유효 자릿수가 19자리인 실수를 저장합니다. 일반적으로 float형 변수의 유효 자릿수는 6~7자리입니다.

06 : double형 변수에 유효 자릿수가 19자리인 실수를 저장합니다. 일반적으로 double형 변수의 유효 자릿수는 15자리입니다.

07~08 : float형 변수가 가질 수 있는 실숫값은 최대 10^{38}에서 최소 10^{-38}입니다. 이 범위를 벗어난 실수를 배정하면 컴파일 오류가 발생하지는 않지만 경고 오류를 나타내며 부정확한 값이 나옵니다.

10 : 값을 출력합니다. 7자리까지는 정확한 값이지만 그 이후로는 알 수 없는 값이 출력됩니다.

float형 변수는 소수점 이하 7자리까지 유효하기 때문에 그 범위에서 사용하는 것이 바람직합니다.

11 : 값을 출력합니다. double형도 마찬가지로 15자리까지는 정확한 값이지만 그 이후로는 알 수 없는 값이 출력됩니다.

12~13 : 지수형 형태로 값을 출력합니다. 유효 자릿수는 앞의 형태와 같습니다.

14~15 : float형의 범위를 벗어난 값을 배정하여 오버플로와 언더플로가 발생했습니다. 오버플로의 경우 무한대를 나타내는 1.#INF00000000이 출력되고, 언더플로의 경우 0.000000000000이 출력됩니다.

실행 결과

```
0.123456791043
0.12345678901234568000
1.234567910433e-001
1.234567890123e-001
1.#INF00000000
0.000000000000
```

04 표준 입출력과 형식 지정자

C 프로그램을 본격적으로 학습하기 전에 입출력문에 대해서도 알아야 합니다. 이 절에서는 입출력에 필요한 제어 문자와 C 언어의 표준 입출력 함수인 printf(), scanf()의 사용법 및 자료형에 따른 형식 지정자를 살펴보겠습니다. 또한 문자와 문자열 입출력 전용 함수에 대해서도 알아봅니다.

4.1 제어 문자

제어 문자escape sequence는 주로 출력문에서 출력의 위치를 조정하거나 특수한 문자를 표현하는 데 사용합니다. 역빗금 기호로 나타내는데 컴퓨터에 따라 \이나 ₩으로 표시됩니다. 앞에서 행을 바꾸는 제어 문자인 \n을 사용했는데, C 언어에서는 이 외에도 다양한 종류의 제어 문자를 제공하고 있습니다.

표 3-7 C 언어의 제어 문자

제어 문자	기능
\a	'삐' 경고음 발생
\b	한 칸 뒤로 옮겨 출력(backspace)
\f	새 페이지의 처음으로 옮겨 출력(form feed)
\n	다음 행의 처음으로 옮겨 출력(new line)
\r	현재 행의 처음으로 옮겨 출력(carriage return)
\t	수평으로 탭만큼 옮겨 출력
\\	역빗금(\) 출력
\'	작은따옴표(') 출력
\"	큰따옴표(") 출력

예제 3-9 escape.c

다음은 다양한 제어 문자의 사용을 보여주는 프로그램입니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05     printf("\t\t\'C언어\'\r재미있는\n");          --- 탭과 위치를 처음으로 옮김
06     printf("\"진짜 재미없습니다\b\b\b\b있습니다\"\n");
       └── 큰따옴표와 문자를 출력하고 위치를 뒤로 옮겨 다시 문자 출력
07     printf("\a\a\a\a\a");                        --- 경고음 발생
08 }
```

프로그램 설명

05 : 2개의 탭과 작은따옴표를 사용하여 'C언어'를 출력하고, \r을 사용하여 출력 위치를 현재 행의 처음으로 옮겨서 '재미있는'을 출력합니다.

06 : 큰따옴표를 사용하여 "진짜 재미없습니다"를 출력하고, \b를 사용하여 뒤로 네 칸 이동한 다음 '있습니다'를 출력하여 앞의 '없습니다'를 삭제한 효과를 냅니다.

07 : 다섯 번의 경고음이 발생합니다.

실행 결과

재미있는 'C언어'
"진짜 재미있습니다"

4.2 표준 출력 함수 : printf()

C 언어에서 표준 출력은 화면(모니터)의 출력을 의미하며 이를 위해 printf() 함수를 제공하고 있습니다. printf() 함수는 단순하게 하나의 리터럴 상수나 변수가 가진 값을 출력할 수도 있고, 형식 제어format specification 문자를 이용하여 다양한 형태로 출력할 수도 있습니다.

```
printf([형식 제어 문자], [변수 또는 리터럴 상수]);
```

[그림 3-14]와 같은 요소로 구성되는 형식 제어 문자는 큰따옴표로 묶어 표시하고 앞에 % 기호를 사용하여 지정합니다. 선택 사항으로 정렬 방식과 부호 표시 여부, 나타낼 숫자의 자릿수와 크기 등을 지정할 수 있으며, 마지막에는 형식 지정자를 표시해야 합니다.

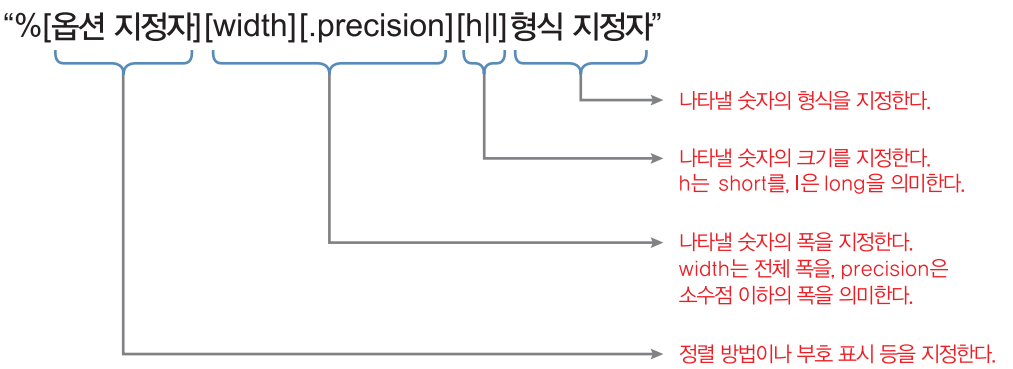


그림 3-14 형식 제어 문자의 구성

표 3-8 printf()의 형식 지정자

형식 지정자	자료형	출력 형식
%d, %i	short, int	부호가 있는 10진수 정수 출력
%X, %x	int	16진수 출력(양수만 가능), 대문자 X의 경우 16진수 알파벳을 대문자로 표시
%o	int	8진수 출력(양수만 가능)
%p	int	포인터 값(메모리 주소 값)을 16진수 여덟 자리로 출력
%u	unsigned int	10진수 정수 출력(양수만 가능)
%c	char	하나의 문자 출력
%f	double	고정 소수점 실수 출력
%E, %e	double	부동 소수점 실수 출력, 대문자 E의 경우 지수 문자로 'E'를 사용
%G, %g	double	소수점 이하 자릿수 고정 소수점 또는 부동 소수점으로 출력, e와 f 중에서 자릿수가 짧은 것을 기준으로 선택하여 출력
%s		문자열 출력
%%		% 출력

표 3-9 printf()의 옵션 지정자

옵션 지정자	의미
–	폭이 지정된 경우 지정된 폭에서 왼쪽 정렬하고, 지정되지 않은 경우 묵시적으로 오른쪽 정렬
+	출력 결과에 부호 기호를 붙이고, 지정되지 않은 경우 음수에만 부호를 붙임
0	오른쪽 정렬인 경우 앞쪽을 모두 0으로 채우고, 지정되지 않은 경우 0으로 채우지 않음
#	형식 지정자가 o(8진수)인 경우 앞에 0을 붙여 출력하고, x 또는 X(16진수)인 경우 앞에 0x 또는 0X를 붙여 출력

printf() 함수는 형식 제어 문자 부분에 일반 문자열을 혼용하여 사용할 수 있으며 일반 문자열은 그대로 출력됩니다. 다음 예를 통해 사용법을 알아봅시다.

```
01 printf("감사합니다\n");
02 printf("%s\n","감사합니다");
03 printf("오늘 %d번째 손님입니다\n", count);
04 printf("%d번째 손님께서 %s을(는) %d개 주문하셨습니다\n", count, title, num);
```

01, 02 ← 문자열 자체 출력. 동일한 결과 출력

03 ← 문자열과 숫자(count 변숫값) 함께 출력

04 ← 숫자와 문자열 함께 출력

4.2.1 정수형 값의 출력

정수 리터럴 값이나 정수형 변숫값을 출력하기 위해 주로 사용하는 형식 지정자는 %d 또는 %i입니다.

예제 3-10 integer1.c

다음은 정수형 변수와 정수 리터럴을 출력하는 프로그램으로 형식 지정자에 따라 정수형 데이터가 다양한 형태로 출력됩니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05     int u_price=-300;
06     int count = 9;
07
08     printf("%d원짜리 %d개를 사면 %d입니다\n",300,9,300*9);
09     printf("%i원짜리 %i개를 사면 %i입니다\n",u_price, count, u_price*count);
       └── %i를 사용하여 변숫값 출력
10     printf("10진수 15를 16진수로 출력하면 : %x, 8진수로 출력하면 : %o\n", 15, 15);
       └── 리터럴(10진수) 15를 %x와 %o를 사용하여 16진수와 8진수로 출력
11     printf("음숫값 -10을 %%d로 출력하면 %d, %%u로 출력하면 %u\n",-10,-10);
       └── %d와 %u를 사용하여 -10 출력. %를 출력하기 위해 %% 사용
12     return 0;
13 }
```

05 ~ 06 ← 정수 변수 선언과 초기화

08 ← %d를 사용하여 정수 리터럴 출력

프로그램 설명

05~06 : 2개의 정수 변수를 선언하고 초기화했습니다.

08 : %d 형식 지정자를 사용하여 10진수를 직접 출력합니다. 위치 순서에 따라 형식 지정자와 10진수가 연결됩니다. 수식으로 지정되는 경우 수식이 계산되어 결과가 출력됩니다.

09 : %i 형식 지정자를 사용하여 변숫값을 출력합니다. 위치 순서에 따라 형식 지정자와 변수가 연결됩니다. 변수를 이용한 수식이 지정되는 경우에도 수식의 결과가 출력됩니다.

10 : %x와 %o 형식 지정자를 사용하여 10진수 값을 16진수와 8진수로 출력합니다.

11 : %d와 %u 형식 지정자를 사용하여 음숫값을 출력합니다. %d를 사용하는 경우 정상적인 음숫값이 출력되지만, 모든 수를 양수로 출력하는 형식 지정자인 %u는 음수를 양수로 취급하여 출력합니다. 또한 %를 출력하기 위해 %% 형식 지정자를 사용했습니다.

실행 결과

```
300원짜리 9개를 사면 2700입니다
-300원짜리 9개를 사면 -2700입니다
10진수 15를 16진수로 출력하면 : f, 8진수로 출력하면 : 17
음숫값 -10을 %d로 출력하면 -10, %u로 출력하면 4294967286
```

[그림 3-15]는 형식 지정자와 옵션 지정자를 사용하여 정수 리터럴을 출력하는 예입니다. 지정자를 사용하여 다양한 출력 형태를 나타냈습니다.

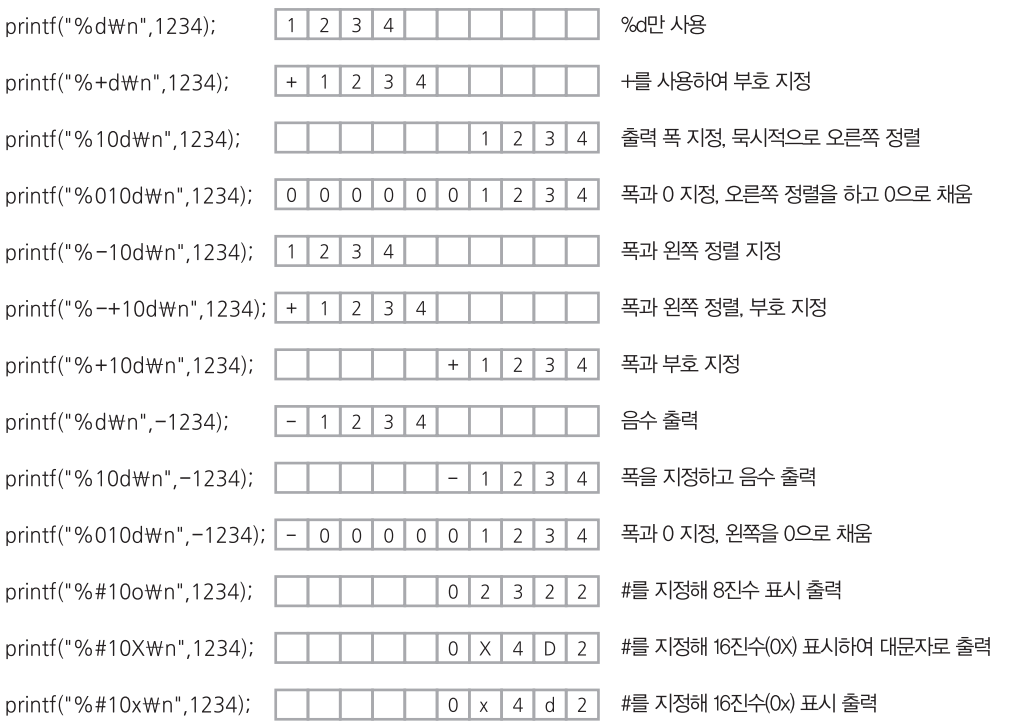


그림 3-15 옵션 지정자를 사용한 정수 리터럴 출력

4.2.2 실수형 값의 출력

실숫값이나 실수형 변숫값을 출력하는 데 주로 사용하는 형식 지정자는 %f 또는 %e입니다. %f는 정수부와 소수부로 구성된 일반적인 실수의 출력에 사용되고, %e는 실수를 지수 형태로 나타

낼 때 사용합니다.

예제 3-11 double1.c

다음은 실수형 값을 출력하는 프로그램입니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      double r = 5.0;
06      double pi = 3.14159;
07
08      printf("반지름이 %f인 원의 넓이는 %f입니다\n", 5.0, 3.14159*5*5);
        └── %f를 사용하여 실수 출력
09      printf("반지름이 %f인 원의 넓이는 %f입니다\n", r, pi*r*r);
        └── %f를 사용하여 실수 변숫값 출력
10      printf("반지름이 %e인 원의 넓이는 %e입니다\n", r, pi*r*r);
        └── %e를 사용하여 실수 변숫값 출력
11      printf("123.456을 %%f로 : %f\n", 123.456);
12      printf("123.456을 %%e로 : %e\n", 123.456);
13      printf("-123.456을 %%E로 : %E\n", -123.456);
14      return 0;
15 }
```

변수 r, pi 선언과 초기화

11 %f를 사용하여 123.456 출력

12~13 %e와 %E를 사용하여 -123.456 출력

프로그램 설명

05~06 : 2개의 실수 변수를 선언하고 초기화했습니다.

08 : %f 형식 지정자를 사용하여 실수를 직접 출력합니다. 위치 순서에 따라 형식 지정자와 실수가 연결됩니다. 수식으로 지정되는 경우 수식이 계산되어 결과가 출력됩니다.

09 : %f 형식 지정자를 사용하여 실수 변숫값을 출력합니다. 위치 순서에 따라 형식 지정자와 변수가 연결됩니다. 변수를 이용한 수식이 지정되는 경우에도 수식이 계산되어 결과가 출력됩니다.

10 : %e 형식 지정자를 사용하여 실수를 지수 형태로 표현합니다.

11 : %f 형식 지정자를 사용하여 실수 123.456을 출력합니다. 일반적인 실수 형태로 출력됩니다.

12~13 : %e와 %E 형식 지정자를 사용하여 123.456과 −123.456을 출력합니다. 지수 형태로 출력됩니다.

실행 결과

```
반지름이 5.000000인 원의 넓이는 78.539750입니다
반지름이 5.000000인 원의 넓이는 78.539750입니다
```

```
반지름이 5.000000e+000인 원의 넓이는 7.853975e+001입니다
123.456을 %f로 : 123.456000
123.456을 %e로 : 1.234560e+002
-123.456을 %E로 : -1.234560E+002
```

[그림 3-16]은 형식 지정자와 옵션 지정자를 사용하여 실수 리터럴을 출력하는 예입니다. 지정자를 사용한 다양한 출력 형태를 나타냈습니다.

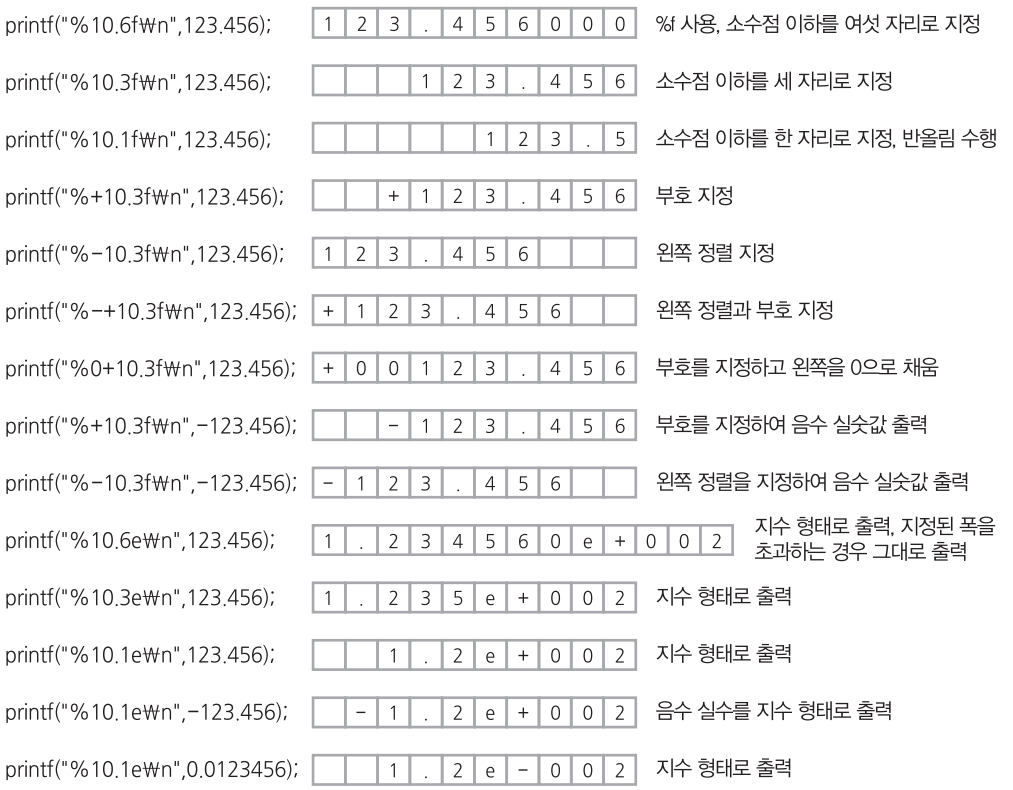


그림 3-16 옵션 지정자를 사용한 실수 리터럴 출력

4.2.3 문자와 문자열 값의 출력

하나의 문자를 출력하기 위한 형식 지정자는 %c이고, 문자열을 출력하기 위한 형식 지정자는 %s입니다. 일반적으로 문자와 문자열은 리터럴 형태로 직접 출력문에 지정할 수 있지만, 변수에 저장한 다음 사용하는 것이 일반적입니다.

C 언어에서 문자를 선언하는 데는 char, 문자열을 선언하는 데는 문자의 배열을 이용합니다. 여기서는 문자열을 생성하기 위한 문자 배열만 간단하게 언급하고 7장에서 자세히 살펴보겠습니다.

예제 3-12 string1.c

다음 프로그램은 문자와 문자열을 사용하는 예입니다. 하나의 문자는 작은따옴표로 리터럴을 표시하고, 문자열은 큰따옴표로 리터럴을 표시합니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
05     char ch1 = 'A';                  --- 문자형 변수를 선언하고 A로 초기화
06     char name[10] = "홍길동";        --- 문자열을 생성하기 위해 문자 배열을 선언하고 초기화
07
08     printf("%c\n",'A');              --- %c를 사용하여 문자 리터럴 출력
09     printf("%s\n","AB");             ---|
10     printf("%s\n","AB CCCCC");       ---|-- %s를 사용하여 문자열 리터럴 출력
11     printf("%s씨 %s\n%s\n", name, "재미있는 C언어","열심히 하세요");
       └--- %s를 사용하여 문자열 변수와 문자열 리터럴 출력
12     return 0;
13 }
```

프로그램 설명

05 : 문자형 변수를 선언하고 값을 A로 초기화했습니다. 문자형 변수는 char로 선언하고, 문자형 리터럴을 나타내기 위해 작은따옴표를 사용합니다.

06 : 문자열을 사용하기 위해 문자형 변수의 배열을 선언하고, 문자열 '홍길동'으로 초기화했습니다. 문자열 리터럴을 나타내는 데는 큰따옴표를 사용합니다.

08 : %c 형식 지정자를 사용하여 문자 'A'를 직접 출력합니다.

09~10 : %s 형식 지정자를 사용하여 문자열을 출력합니다.

11 : %s 형식 지정자를 사용하여 문자열을 저장한 변수와 2개의 문자열을 서로 다른 행에 출력합니다. 위치 순서에 따라 형식 지정자와 변수 또는 리터럴이 연결됩니다.

실행 결과

```
A
AB
AB CCCCC
홍길동씨 재미있는 C언어
열심히 하세요
```

[그림 3-17]은 형식 지정자와 옵션 지정자를 사용하여 문자와 문자열을 출력하는 예입니다.

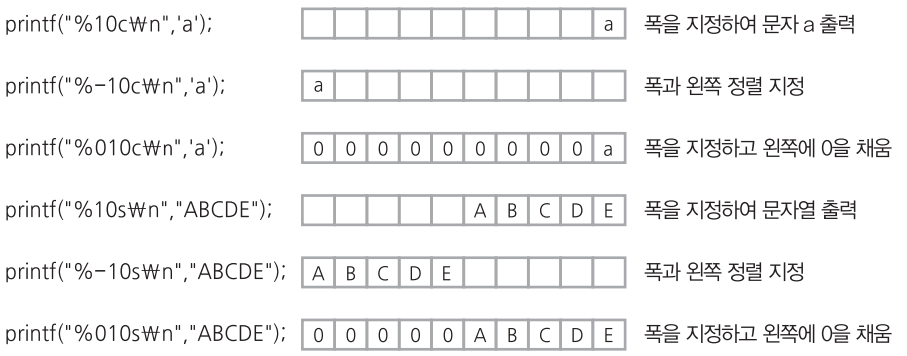


그림 3-17 옵션 지정자를 사용한 문자와 문자열 리터럴 출력

4.3 표준 입력 함수 : scanf()

C 언어에서의 표준 입력은 프로그램 실행 중 키보드를 통한 입력을 의미하며, 이를 위해 표준 입력 함수 scanf()를 제공하고 있습니다. 표준 출력 함수 printf()는 인수로 리터럴이나 변수명을 사용했지만, scanf()는 함수의 인수로 변수의 주소(&)를 사용하는 것이 차이점입니다.

& 연산자가 변수에 적용되면 그 변수의 값이 저장된 메모리의 주소를 의미합니다. 즉 scanf() 함수는 키보드로부터 입력된 값을 메모리의 주소(변수의 값이 저장된 주소)에 저장합니다.

```
scanf([형식 제어 문자], [& 변수 또는 배열, 문자열 변수]);
```

TIP 일반 변수는 변수명 앞에 & 기호를 붙이고 배열과 문자열 변수에는 붙이지 않습니다.

표 3-10 scanf()의 형식 지정자

형식 지정자	자료형	출력 형식
%d, %u	short, int	부호가 있는 10진수 정수 입력
%i	int	부호가 있는 10진수, 16진수, 8진수 정수 입력
%X, %x	int	부호가 있는 16진수 입력, 대문자 X의 경우 A~F까지 사용 가능
%o	int	부호가 있는 8진수 입력
%p	int	포인터로 입력
%c	char	하나의 문자 출력
%f, %E, %e, %G, %g	float	고정 소수점과 부동 소수점 실수 입력
%lf	double	부동 소수점 실수 입력
%s	char 배열	문자열

scanf() 함수의 형식 제어 문자 구성은 printf() 함수와 같습니다([그림 3-14] 참조). scanf()의 형식 지정자는 printf()의 형식 지정자와 유사하지만 차이점이 있는데, scanf() 함수는 형식 제어 문자 내에 문자 리터럴을 직접 사용할 수 없다는 것입니다.

다음 예와 같이 scanf() 함수 내에서는 형식 지정자와 옵션 지정자만 사용할 수 있으며, 리터럴을 사용하면 예상치 않은 결과가 나타납니다.

```
01 scanf("나이 입력 : %d", &age);   --- 예상치 않은 결과가 나타남
02 ..... // 생략
03 printf("나이 입력 :");          --- 문자열 출력
04 scanf("%d", &age);              --- scanf()는 형식 제어 문자만 사용하여 값을 입력
```

여기서 잠깐 scanf() 함수와 scanf_s() 함수

- C 언어는 scanf() 함수와 scanf_s() 함수를 제공하는데 비주얼 스튜디오 2010 이상의 버전에서는 scanf() 함수보다 scanf_s() 함수의 사용을 권장하고 있습니다. scanf() 함수의 경우 문자열을 입력할 때 정의된 크기보다 큰 문자열이 입력되면 실행 시간 오류(오버플로)가 발생합니다. 반면 scanf_s() 함수는 입력된 문자열이 정의된 크기를 벗어나면 입력을 받지 않아 오류를 발생시키지 않습니다.
- 비주얼 스튜디오 2013 이전 버전에서는 scanf() 함수를 사용하는 경우 경고 메시지가 출력되지만 프로그램은 실행됩니다.
- 비주얼 스튜디오 2013 버전부터는 scanf() 함수를 사용하면 오류 메시지가 출력됩니다. 이 경우 scanf_s() 함수를 사용하거나, 또는 〈stdio.h〉(C:₩Program Files (x86)₩Microsoft Visual Studio 12.0₩VC₩include₩stdio.h) 파일의 끝에 #pragma warning(disable:4996) 문장을 추가하면 사용할 수 있습니다.

4.3.1 정수형 값의 입력과 주소 연산자(&)

scanf() 함수로 정수형 값을 입력받기 위해서는 주로 %d, %u, %i 형식 지정자를 사용합니다. & 연산자는 주소를 나타내는 연산자로서 변수 이름 앞에 붙여서 사용합니다.

예제 3-13 intinput1.c

다음 프로그램은 scanf() 함수를 사용하여 정수형 값을 입력하는 예를 보여줍니다. 형식 지정자를 사용하여 정숫값을 다양한 진법으로 출력할 수 있습니다. 또한 변수의 주소 값을 나타내는 형식 지정자 %p를 사용하여 16진수 여덟 자리 주소 값의 출력이 가능합니다.

```
01 #include <stdio.h>
02
03 int main()
04 {
```

```
05      int price;
06      int count;
07      printf("가격을 입력하십시오 : ");
08      scanf("%d", &price);
09      printf("수량을 입력하십시오 : ");
10      scanf("%d", &count);
11      printf("감사합니다. %d원짜리 %d개를 선택하였습니다. 총액은 %d입니다\n", price, count,
    price*count);
12      printf("price 변수의 값은 : %d, 주소는 : %0x\n",price, &price);
13      printf("count 변수의 값은 : %d, 주소는 : %0x\n",count, &count);
14      return 0;
15  }
```

2개의 정수 변수 선언

--- %d를 사용하여 정숫값을 입력받아 변수에 저장

--- %d를 사용하여 정숫값을 입력받아 변수에 저장

각 변수의 값과 주소 출력

프로그램 설명

05~06 : 2개의 정수 변수를 선언했습니다. 초기화는 scanf()를 사용하여 수행합니다.

08, 10 : scanf() 함수를 사용하여 키보드로부터 정숫값을 입력받았습니다. 입력에는 변수 이름만을 사용할 수 없고 변수 이름 앞에 주소를 나타내는 &를 붙여야 합니다. 즉 '&변수 이름'은 변수 이름이 가리키는 곳에 값을 저장하라는 의미입니다.

12~13 : 각각의 변수에 대해 변수가 가지고 있는 값과 변수의 주소(16진수 여덟 자리)를 출력합니다.

실행 결과

```
가격을 입력하십시오 : 3000
수량을 입력하십시오 : 3

감사합니다. 3000원짜리 3개를 선택하였습니다. 총액은 9000입니다
price 변수의 값은 : 3000, 주소는 : 0031FAD4
count 변수의 값은 : 3, 주소는 : 0031FAC8
```

TIP 실행할 때마다 주소의 값이 바뀝니다.

4.3.2 실수형 값의 입력

scanf() 함수를 사용하여 float형 실숫값을 입력받기 위해서는 %f 또는 %e 형식 지정자를 주로 사용하고, double형의 실숫값을 입력받기 위해서는 %lf 형식 지정자를 사용합니다. scanf() 함수에서 실수형 값을 입력받을 때도 & 연산자를 변수 이름 앞에 붙여야 합니다.

예제 3-14 doubleinput1.c

다음 프로그램은 scanf() 함수로 실수형 값을 입력하는 예를 보여줍니다. 형식 지정자를 사용하여 실숫값을 다양한 형태로 출력할 수 있습니다.

```
01 #include <stdio.h>
02 #define PI 3.14159    --- 전처리기에 의해 처리되는 상수 PI 선언
03
04 int main()
05 {
06     double r;          ┐
07     double pi;         ┘-- double형 변수 선언
08
09     printf("반지름 r을 입력 : ");
10     scanf("%lf", &r);  --- %lf를 사용하여 반지름 입력
11     printf("반지름이 %f인 원의 넓이는 %f입니다\n", r, PI*r*r);
       └── 반지름과 PI를 이용하여 넓이 출력(%f 사용)
12     printf("반지름이 %e인 원의 둘레는 %e입니다\n", r, 2*PI*r);
       └── 반지름과 PI를 이용하여 둘레 출력(%e 사용)
13     return 0;
14 }
```

프로그램 설명

02 : 전처리기에 의해 처리되는 상수 PI를 선언했습니다. 전처리기는 프로그램에서 PI를 사용하는 모든 곳을 3.14159로 대치합니다.

06~07 : double형 실수 변수 2개를 선언했습니다.

10 : %lf 형식 지정자를 사용하여 double형 실수를 입력받았습니다. double형 변수에 %f(float형 값을 입력하는 형식 지정자)를 사용하면 엉뚱한 값이 입력되어 결과가 틀려집니다.

11 : 입력받은 반지름과 PI를 이용하여 원의 넓이를 출력합니다. 출력 형식 지정자로 %f를 사용했습니다.

12 : 입력받은 반지름과 PI를 이용하여 원의 둘레를 출력합니다. 출력 형식 지정자로 %e를 사용했습니다.

실행 결과

```
반지름 r을 입력 : 50
반지름이 50.000000인 원의 넓이는 7853.975000입니다
반지름이 5.000000e+001인 원의 둘레는 3.141590e+002입니다
```

4.3.3 문자와 문자열 값의 입력

scanf() 함수로 하나의 문자를 입력받기 위해서는 %c 형식 지정자를 사용하고, 문자열을 입력받기 위해서는 %s를 사용합니다. C 언어에서는 문자열을 저장하는 데 문자 배열을 사용합니다(7장 참조).

C 언어에서 문자 배열은 여러 개의 문자로 구성된 하나의 자료 구조로 취급되며, 문자 배열로 선언된 변수(식별자)는 문자들의 첫 번째 문자가 저장된 주소를 가지게 됩니다. 그러므로 scanf() 함수로 문자열을 입력받는 경우에는 변수 이름 앞에 &를 붙이지 말아야 합니다(7장 참조).

예제 3-15 charinput1.c

다음은 scanf() 함수로 문자와 문자열 값을 입력받아 처리하는 프로그램입니다. 문자는 주소 연산자 &를 사용해서 입력받지만 문자열에는 주소 연산자를 사용하지 않습니다.

```
01 #include <stdio.h>
02 #define NAT "대한민국"          --- 심벌릭 리터럴 상수 정의
03
04 int main()
05 {
06     char sex;
07     char city[20];              -- 문자와 문자 배열 선언
08     char name[10];
09
10     printf("성별 입력(F/M) : ");
11     scanf("%c",&sex);           --- %c를 사용하여 문자형 변숫값 입력
12     printf("살고 있는 도시 이름과 성함을 입력(도시명 이름) : ");
13     scanf("%s %s", city, name); -- %s를 사용하여 2개의 문자열 입력
14     printf("%s씨(%c)는 %s %s에 살고 있습니다\n", name, sex, NAT, city); --- 문자열 출력
15     return 0;
16 }
```

프로그램 설명

02 : 전처리기에 의해 처리되는 문자열 리터럴을 선언하고 초기화했습니다.

06~08 : 문자형 변수와 문자 배열을 선언했습니다.

11 : %c 형식 지정자를 사용하여 키보드로부터 입력받은 하나의 문자를 문자 변수에 저장합니다. 이때 주소 연산자를 사용했습니다.

13 : %s 형식 지정자를 사용하여 문자열을 2개의 문자 배열에 저장합니다. 입력되는 2개의 문자열은 스페이스에 의해 구분됩니다. 문자열은 이름 자체가 첫 번째 문자를 가리키는 주소이므로 주소 연산자를 사용하지 않았습니다.

14 : 문자와 문자열을 출력합니다.

실행 결과

```
성별 입력(F/M) : M
살고 있는 도시 이름과 성함을 입력(도시명 이름) : 서울특별시 홍길동
홍길동씨(M)는 대한민국 서울특별시에 살고 있습니다
```

4.4 문자와 문자열 입출력 전용 함수

문자와 문자열을 입출력하는 데는 printf(), scanf() 함수를 사용하는 방법 외에 전용 함수를 사용하는 방법이 있습니다. C 언어는 문자, 문자열과 연관된(입출력, 문자열 복사, 문자열 조작) 다양한 함수를 제공합니다. 이 절에서는 문자를 입출력하는 전용 함수 getchar(), putchar()와 문자열을 입출력하는 전용 함수 gets(), puts()에 대해 살펴보겠습니다.

TIP 7장에서 문자열과 연관된 함수에 대해 자세히 설명했습니다.

문자 입출력 전용 함수인 getchar(), putchar()는 하나의 문자를 입출력하는 함수입니다. getchar() 함수는 키보드를 통해 하나의 문자를 입력받아 변수에 저장하고, putchar() 함수는 하나의 문자를 출력합니다. 다음은 하나의 문자를 입력받고 출력하는 예입니다.

```
01 char sex;          --- 문자형 변수 선언
02 sex=getchar();     --- 키보드를 통해 하나의 문자를 입력받아 변수에 저장
03 putchar(sex);      --- 변수의 값 출력
```

문자열 입출력 전용 함수인 get(), puts() 함수가 scanf(), printf() 함수와 다른 점은 형식 지정자를 사용하지 않는다는 것입니다. 즉 gets()와 puts() 함수는 하나의 문자열 변수 또는 하나의 문자열 리터럴을 지정하여 입출력합니다.

또한 입력 함수인 scanf()와 gets()의 차이는, scanf() 함수는 스페이스를 기준으로 입력 문자열을 구분하지만 gets() 함수는 행의 끝까지를 하나의 입력 문자열로 인식한다는 것입니다.

```
01 char addr[100];          --- 주소 문자열을 저장할 문자 배열 선언
02 scanf("%s", addr);       --- 문자열 '서울시 마포구 공덕동' 입력
03 printf("%s\n", addr);    --- '서울시'만 출력
```

다음 예는 gets() 함수를 사용하여 같은 작업을 한 경우입니다. puts() 함수는 문자열을 출력하고 행을 자동으로 바꿔서 출력합니다.

```
01 char addr[100];      --- 주소 문자열을 저장할 문자 배열 선언
02 gets(addr);          --- 문자열 '서울시 마포구 공덕동' 입력
03 puts(addr);          --- '서울시 마포구 공덕동'을 출력하고 행을 바꿈
04 printf("%s", addr);  --- '서울시 마포구 공덕동' 출력
```

다양한 입출력 전용 함수를 사용하다 보면 엔터 키의 사용에 문제가 발생하는 경우가 있습니다. getchar()는 하나의 문자만을 입력받고, scanf("%d",age)는 공백이 나타날 때까지의 정수 데이터를 입력받습니다. 즉 getchar() 함수와 scanf() 함수는 엔터 키를 포함하지 않은 값을 입력으로 받아들입니다. 그러나 프로그램에서는 데이터를 입력하고 엔터 키를 쳐서 다음으로 진행하는데, 이때의 엔터 키가 다음 입력문의 입력이 되기 때문에 문제가 발생하는 것입니다.

다음과 같이 여러 종류의 데이터를 입력받는 경우에 문제가 발생합니다.

```
01 printf("나이 입력 : ");
02 scanf("%d", &age);   --- 나이를 입력하고 엔터 키를 치면 나이만 변수에 저장됨
03 puts("성별 입력(male(남) : m / female(여) : f) : ");
04 sex=getchar();       --- 2행에서 입력된 엔터 키가 값으로 입력되어 다음 행이 수행됨
05 puts("주소 입력 : ");
06 gets(addr);          --- gets() 함수는 엔터 키를 포함한 값을 입력으로 인지
07 puts("이름 입력 : ");
08 gets(name);
```

위의 예는 3개의 값을 입력받아야 하는데 나이를 입력하고 엔터 키를 쳐서 엔터 키의 값이 다음 입력문의 입력이 된 경우입니다.

C 언어는 이러한 문제를 해결하기 위해 표준 입출력 라이브러리에 현재의 키보드 입력을 모두 버리는 fflush(stdin) 함수를 제공합니다. 위 예의 경우에는 다음과 같이 fflush(stdin) 함수를 사용하여 정상적으로 값을 입력받을 수 있습니다.

```
01 printf("나이 입력 : ");
02 scanf("%d", &age);
03 fflush(stdin);       --- 입력받은 엔터 키 값을 버림
04 puts("성별 입력(male(남) : m / female(여) : f) : ");
05 sex=getchar();
06 fflush(stdin);       --- 입력받은 엔터 키 값을 버림
07 puts("주소 입력 : ");
08 gets(addr);          --- gets() 함수는 엔터 키를 포함한 값을 입력받으므로 fflush() 함수가 불필요함
09 puts("이름 입력 : ");
10 gets(name);
```

예제 3-16 getput1.c

다음은 문자와 문자열 입출력 전용 함수를 사용하는 프로그램으로 나이, 성별, 주소, 이름을 입력받아 출력하는 것을 보여줍니다. 다양한 입력 함수를 사용할 때 발생하는 엔터 키의 문제를 fflush(stdin) 함수로 해결했습니다.

```
01 #include <stdio.h>
02 #define NAT "대한민국"  --- 심벌릭 문자열 상수(리터럴) 정의
03
04 int main()
05 {
06     int age;
07     char sex;
08     char addr[100];
09     char name[20];
10
11     printf("나이를 입력하세요 : ");
12     scanf("%d", &age);  --- scanf() 함수를 사용하여 나이 입력
13     fflush(stdin);  --- 나이 입력 후의 엔터 키를 버림
14     puts("성별 입력(male(남) : m / female(여) : f) : ");
15     sex=getchar();  --- getchar() 함수를 사용하여 한 문자 입력
16     fflush(stdin);  --- 문자 입력 후의 엔터 키를 버림
17     puts("주소 입력 : ");
18     gets(addr);  --- gets() 함수는 엔터 키를 포함한 행 전체 입력
19     puts("이름 입력 : ");
20     gets(name);  --- gets() 함수는 엔터 키를 포함한 행 전체 입력
21     puts("\n");
22     printf("%s씨는 성별(m:남자 f:여자)은 %c이며, %d살이고 %s %s에 살고 있습니다\n", name,
   sex, age, NAT, addr);
23     printf("%d\n",age);
24     puts(name);
25     putchar(sex);  --- putchar() 함수를 사용하여 문자 출력
26     putchar('\n');  --- putchar() 함수를 사용하여 문자 출력
27     puts(NAT);
28     puts(addr);
29     return 0;
30 }
```

프로그램 설명

02 : 전처리기에 의해 처리되는 심벌릭 문자열 상수를 선언했습니다.

06~08 : 2개의 변수와 2개의 문자 배열을 선언했습니다.

12 : scanf() 함수를 사용하여 나이를 입력하고 엔터 키를 쳤습니다. 이때 숫자인 나이는 변수에

입력되고 엔터 키는 표준 입력 버퍼에 남아 다음 입력문의 입력이 됩니다.

13, 16 : fflush(stdin) 함수는 표준 입력 장치의 버퍼에 있는 내용을 비우는 역할을 합니다. 이 함수를 수행함으로써 엔터 키를 버립니다.

15 : getchar() 함수를 사용하여 한 문자를 입력하고 엔터 키를 쳤습니다. 이 경우도 문자만 변수에 입력되고 엔터 키는 표준 입력 버퍼에 남아 다음 입력문의 입력이 됩니다.

18, 20 : gets() 함수는 scanf(), getchar() 함수와 달리 엔터 키까지 포함한 내용을 하나의 문자열로 입력받습니다. 그러므로 gets() 함수를 사용하여 입력한 다음에는 엔터 키가 표준 입력 버퍼에 남지 않습니다.

25, 26 : putchar() 함수를 사용하여 하나의 문자를 출력합니다. 이 함수는 puts() 함수와 달리 출력 후에 행을 바꾸지 않습니다.

실행 결과

```
나이를 입력하세요 : 20
성별 입력(male(남) : m / female(여) : f) : m
주소 입력 :
서울시 마포구 공덕동 256-2
이름 입력 :
홍길동

홍길동씨는 성별(m:남자 f:여자)은 m이며, 20살이고 대한민국 서울시 마포구 공덕동 256-2에 살고 있습니다
20
홍길동
m
대한민국
서울시 마포구 공덕동 256-2
```

요약

1 식별자와 예약어

❶ C 언어에는 32개의 예약어가 있으며, 예약어나 C 언어에서 제공되는 라이브러리 함수 이름을 식별자로 사용할 수 없습니다.

❷ C 언어에서는 대 · 소문자를 구분합니다.

❸ C 언어에서 식별자를 사용하는 데는 다음과 같은 규칙을 따라야 합니다.

- 식별자는 문자, 숫자, 특수문자(예 : _, $)로 구성될 수 있습니다.
- 식별자의 첫 문자는 문자나 특수문자로 시작할 수 있지만 숫자는 사용할 수 없습니다.
- 식별자는 길이에 제한을 두지 않습니다.
- 같은 문자의 대 · 소문자(예 : Sum과 sum)는 서로 다른 식별자로 취급합니다.
- 식별자 중간에 공백이 들어갈 수 없습니다.
- 예약어는 식별자로 사용할 수 없습니다.

2 변수와 상수

❶ 변수는 값이 저장된 메모리에 주어진 이름입니다.

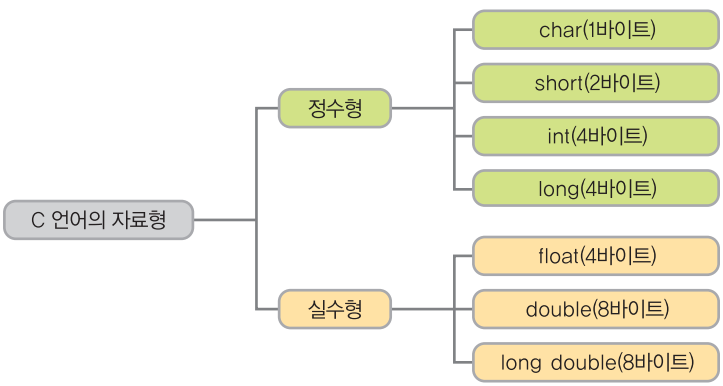
❷ 변수 이름을 정하는 규칙은 식별자의 경우와 같습니다.

❸ C 언어에서 상수는 한 번 설정되면 프로그램이 수행되는 동안 변하지 않는 값을 의미합니다. 상수는 크게 두 가지 종류로 구분할 수 있습니다.

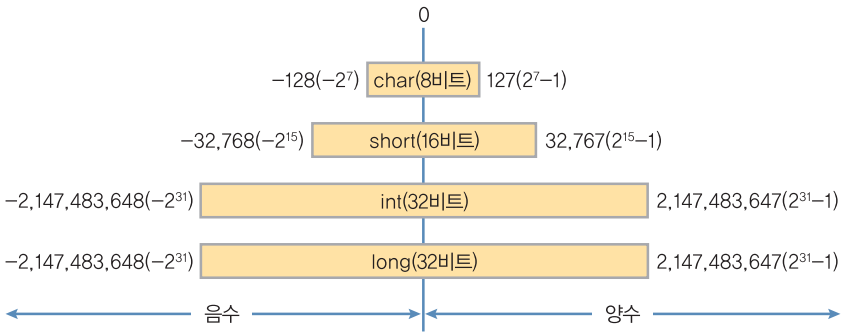
- 리터럴 상수 : 프로그램에서 직접 사용하는, 이름이 없는 상수
- 심벌릭 상수 : 이름을 붙여서 변수처럼 사용하는 상수

3 자료형

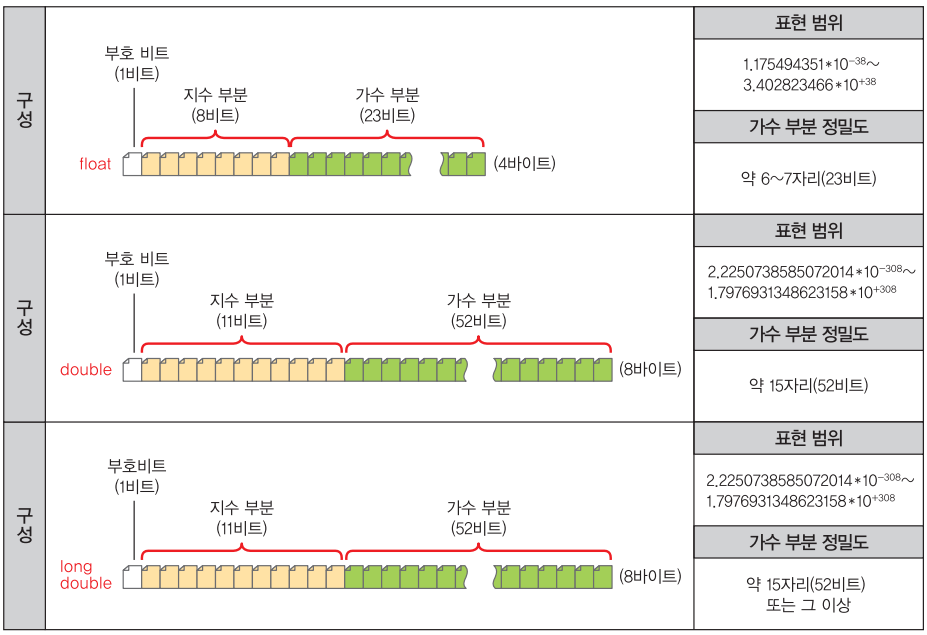
❶ C 언어는 기본 자료형으로 정수형 4개, 실수형 3개를 제공합니다.



❷ 정수형 자료형 4개는 서로 다른 비트 수로 구성되며, 표현할 수 있는 수의 범위도 비트 수에 따라 다릅니다.



❸ 실수 자료형의 소수점은 고정 소수점 방식과 부동 수소점 방식으로 표현이 가능하며, C 언어는 세 가지 실수형을 제공합니다.



4 표준 입출력과 형식 지정자

❶ C 언어는 키보드로 값을 입력받는 scanf() 함수와 모니터로 값을 출력하는 printf() 함수를 표준 입출력 함수로 제공합니다.

❷ 입출력 함수는 다양한 형식 지정자와 옵션 지정자를 사용하여 출력을 정밀하게 조절할 수 있습니다.

❸ C 언어는 문자와 문자열의 편리한 입출력을 위해 scanf(), printf() 함수 외에 getchar(), putchar(), gets(), puts() 함수를 제공합니다.

CHAPTER 03 제출문제

학과(전공)	학년	학번	이름	제출일자

01 식별자와 예약어

1 다음 중 식별자로 사용할 수 없는 것을 고르고 그 이유를 설명하세요(프로그램으로 테스트해보세요).

count1	box_number	abcde	@name	test_score	CSKim
C Program	5park	my.addr	else	Class_cLASS	return

2 다음 중에서 C 예약어가 아닌 것을 고르세요(프로그램으로 테스트해보세요).

final	auto	int	case	repeat	goto	default	extern
define	do	typedef	until	enum	include	import	long

3 C 언어의 식별자 사용 규칙에 대해 설명하세요.

4 다음 프로그램을 실행해보고 오류가 발생하는 경우 오류의 원인을 설명하세요. 오류가 발생하지 않는다면 결과를 나타내세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      int scanf = 30;
06      int a = 20;
07      printf("%d\n", scanf + a);
08 }
```

02 변수와 상수

1 변수를 정의하고, 변수를 사용하지 못할 경우 발생할 수 있는 문제를 쓰세요.

2 상수는 리터럴 상수와 심벌릭 상수로 구분할 수 있는데 각각에 대해 설명하세요.

3 실행 결과가 다음과 같은 프로그램을 작성하세요. 리터럴 상수를 출력문에 직접 사용하여 정수를 출력하고, 심벌릭 상수는 #define문과 const문을 사용하여 문자열과 정수를 출력하세요.

실행 결과

```
리터럴 상수(정수) : 18
리터럴 상수(정수) : -45
심벌릭 상수(문자열) : Seoul
심벌릭 상수(정수) : 100
```

03 자료형

1 다음 프로그램을 실행하면 오류가 발생하는데, 오류를 발생시키는 부분을 찾아내고 그 이유를 설명하세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      age = 20;
06      printf("저의 나이는 %d세입니다.", age);
07      return 0;
08 }
```

2 C 언어의 자료형 중 정수형이 아닌 것은?

❶ char ❷ long ❸ float ❹ short ❺ int

3 다음 프로그램을 실행하면 오류가 발생하는데, 오류를 발생시키는 부분을 찾아내고 그 이유를 설명하세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      int a = 10;
06      printf("변수 a의 값은 : %d\n", a);
07      printf("변수 b의 값은 : %d\n", b);
08      int b = 20;
09      return 0;
10 }
```

4 상수를 적절한 변수에 저장하고, 실행 결과와 같이 지수 표기법으로 출력하는 프로그램을 작성하세요(입력 상수 : 3.1415926535, 123456, 0.00987).

실행 결과

```
3.1415926535e+000
1.234560e+005
9.870000e-003
```

5 다음과 같은 실수를 scanf() 함수로 입력받고 소수점 이하 세 자리까지 출력하는 프로그램을 작성하고 실행 결과를 나타내세요.

123.4567	1.234567e+002	123456.7e-003

실행 결과

6 C 언어에서 사용하는 데이터형 중 알맞은 것을 빈칸에 쓰세요.

❶ [ ] name[10] = "홍길동";

❷ [ ] sum = 100;

❸ [ ] avg = 0.123456789;

❹ [ ] div = 5.9;

❺ [ ] grade = 'A';

7 다음의 정수형이 표현할 수 있는 수의 범위를 쓰세요.

변수	범위
char(8비트)	
short(16비트)	
int(32비트)	

8 다음 프로그램은 아래와 같은 결과를 출력합니다. 예상치 않은 결과가 나타난 이유를 설명하세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      int num = 2147483647;
06      printf("num의 값 : %d\n", num);
07      num = num + 1;
08      printf("num의 값+1 : %d\n", num);
09 }
```

실행 결과

num의 값 : 2147483647
num의 값+1 : –2147483648

04 표준 입출력과 형식 지정자

1 printf() 함수를 사용하여 실수 2014.727에 대해 오른쪽 정렬을 수행하고, 부호를 나타내며, 앞의 남는 부분을 0으로 채워 다음과 같이 출력하는 프로그램을 작성하세요.

실행 결과

+0002014.727

2 다음 출력문의 결과를 오른쪽 칸에 나타내세요.

출력문

```
printf("%-5c%5c\n", 'x','y');
printf("%-5c%-5c\n", 'x', 'y');
printf("%s\n", "XYZ");
printf("%10s\n", "XYZ");
printf("%5s%05s\n", "XYZ","ABC");
printf("%010s\n", "XYZ");
```

결과

3 다음 출력문의 결과를 오른쪽 칸에 나타내세요.

출력문

```
printf("%10.5f\n", 3.14159);
printf("%-10.2f\n", 3.14159);
printf("%10.4f\n", 3.14159);
printf("%10.3e\n", 3.14159);
printf("%010.1e\n", 3.14159);
printf("%10.1e\n", 0.00314159);
```

결과

4 다음은 printf() 함수와 scanf() 함수를 사용하여 작성한 프로그램입니다. 이를 puts() 함수와 gets() 함수를 사용하여 다시 작성하세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      char name[10];
06      char unv[20];
07
08      printf("이름을 입력하세요 : ");
09      scanf("%s", &name);
10      printf("학교를 입력하세요 : ");
11      scanf("%s", &unv);
12      printf("%s \n%s", name, unv);
13      return 0;
14 }
```

5 다음 중 '다음 행의 처음으로 옮겨 출력'하는 제어 문자는?

❶ \a ❷ \t ❸ \n ❹ \f

6 2개의 실수 23.7과 18.5의 덧셈, 뺄셈, 곱셈, 나눗셈 결과를 다음과 같이 출력하는 프로그램을 작성하세요.

```
두 수의 합 : 42.2
두 수의 차 : 5.2
두 수의 곱 : 438.5
두 수의 나누기 : 1.3
```

7 다음 두 문장의 출력 결과가 다르다면 그 이유를 설명하세요.

```
printf("%d\n", 5+1);
printf("5+1\n");
```

8 다음은 하나의 숫자를 형식 지정자를 다르게 하여 출력한 프로그램입니다. 실행 결과를 나타내세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      printf("%d\n", 93);
06      printf("%o\n", 93);
07      printf("%x\n", 93);
08      printf("%X\n", 93);
09      return 0;
10 }
```

실행 결과

9 숫자 하나를 입력받아 다음과 같이 원의 반지름, 둘레, 넓이와 구의 부피를 나타내는 프로그램을 작성하세요(단, PI 값은 3.14159). TIP 원의 둘레 : $2\pi r$, 원의 넓이 : πr^2, 구의 부피 : $4/3\pi r^3$

실행 결과

숫자를 입력하세요 : 5
원의 반지름 : 5
원의 둘레 : 31.42
원의 넓이 : 78.54
구의 부피 : 392.70

10 다음 프로그램을 실행하면 오류가 발생하는데, 오류를 발생시키는 부분을 찾아내고 그 이유를 설명하세요.

```
01 #include <stdio.h>
02
03 int main()
04 {
05      int num1, num2, sum;
06      printf("합을 구할 두 수를 입력하세요(space로 구분) : ");
07      scanf("%d %d", num1, num2);
08      sum = num1 + num2;
09      printf("입력받은 두 수의 합은 : %d\n", sum);
10      return 0;
11 }
```