C프로그래밍

4 연산자, 상수, 조건문 (1/2)

01
관계 연산자

관계 연산자 (1/3)

- **관계 연산자(또는 비교 연산자)는 값의 크기를 비교**
  - ✓ 그 결과 는 참(true)이나 거짓(false) 중 하나
  - ✓ 조건문(if)이나 반복문(for, while)에서 사용하며 단독으로 사용하는 경우는 별로 없음
  - ✓ 일반적으로 참은 1로, 거짓은 0으로 표시

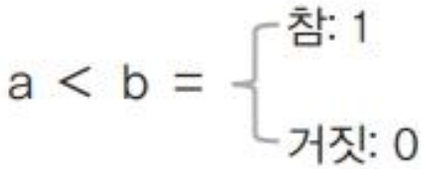


그림 4-4 관계 연산자의 기본 개념

표 4-3 관계 연산자

연산자	의미	설명
==	같다.	두 값이 동일하면 참이다.
!=	같지 않다.	두 값이 다르면 참이다.
〉	크다.	왼쪽이 크면 참이다.
〈	작다.	왼쪽이 작으면 참이다.
〉=	크거나 같다.	왼쪽이 크거나 같으면 참이다.
〈=	작거나 같다.	왼쪽이 작거나 같으면 참이다.

관계 연산자 (2/3)

기본 4-5 관계 연산자 사용 예 — 4-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 100 , b = 200;
06
07    printf(" %d  == %d 는   %d  이다.\n", a, b, a == b);
08    printf(" %d  != %d 는   %d  이다.\n", a, b, a != b);
09    printf(" %d  > %d 는   %d  이다.\n",a, b, a > b);
10    printf(" %d  < %d 는   %d  이다.\n",a, b, a < b);
11    printf(" %d  >= %d 는   %d  이다.\n", a, b, a >= b);
12    printf(" %d  <= %d 는   %d  이다.\n", a, b, a <= b);
13
14    printf(" %d  =  %d 는 %d  이다.\n", a, b, a=b);
15 }
```

관계 연산자 '같다'

관계 연산자 '같지 않다'

관계 연산자 '크다'

관계 연산자 '작다'

관계 연산자 '크거나 같다'

관계 연산자 '작거나 같다'

관계 연산자가 아닌 대입 연산자를 수행한다.

실행 결과

```
100  == 200 는   0  이다.
100  != 200 는   1  이다.
100  > 200 는   0  이다.
100  < 200 는   1  이다.
100  >= 200 는   0  이다.
100  <= 200 는   1  이다.
200  =  200 는 200  이다.
```

관계 연산자 (3/3)

- **기본 4-5 복기**

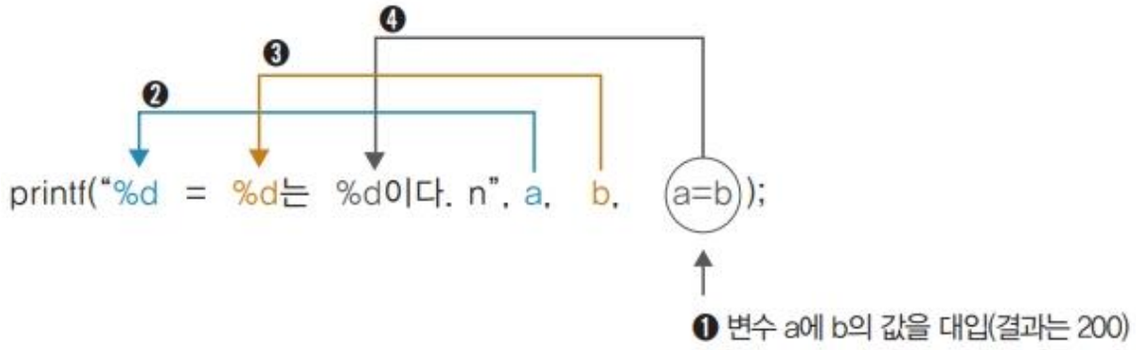


그림 4-5 대입 연산자의 작동

- ✓ ❶의 a=b는 b의 값을 a에 대입하라는 의미
  - 현재 b에는 200이 들어 있으므로 a에도 200이 대입
  - 그 결과 a=b의 위치에는 200이라는 값이 들어감
- ✓ ❷의 a에는 현재 200이 들어 있으므로 첫 번째 %d에는 200이 대입
- ✓ ❸의 b에도 현재 200이 들어 있으므로 두 번째 %d 에도 200이 대입
- ✓ ❹의 a=b는 200이므로 세 번째 %d에도 200이 대입되어 결국 '200 = 200는 200이다.'가 출력

02
논리 연산자

논리 연산자 (1/3)

- **논리 연산자는 주로 여러 가지 조건을 복합적으로 사용**
  - ✓ 예를 들어 “a라는 값이 100과 200 사이에 들어 있어야 한다.”
  - ➔ ‘a는 100보다 크다. 그리고 a는 200보다 작다.’라고 표현할 수 있음
  - ➔ 참이 되려면 (a 〉 100)도 참이 되어야 하고 (a 〈 200)도 참이 되어야
  - ➔ 즉 a가 100과 200 사이에 있어야만 두 조건 모두 참이 됨

```
(a 〉 100 ) && (a 〈 200)
```

표 4-4 논리 연산자

연산자	의미		사용 예	설명
&&	~이고	그리고(AND)	(a 〉 100) && (a 〈 200)	둘 다 참이어야 참이다.
||	~이거나	또는(OR)	(a 〉 100) || (a 〈 200)	둘 중 하나만 참이어도 참이다.
!	~아니다	부정(NOT)	!(a == 100)	참이면 거짓, 거짓이면 참이다.

논리 연산자 (2/3)

기본 4-6 논리 연산자 사용 예 1 4-6.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int  a = 99;
06
07    printf(" AND 연산 : %d \n", (a >= 100) && (a <= 200));   ----- AND 연산을 사용한다.
08    printf(" OR 연산  : %d \n", (a >= 100) || (a <= 200));   ----- OR 연산을 사용한다.
09    printf(" NOT 연산 : %d \n", !(a==100));                  ----- NOT 연산을 사용한다.
10 }
```

실행 결과

```
AND 연산 : 0
OR 연산  : 1
NOT 연산 : 1
```

논리 연산자 (3/3)

응용 4-7 논리 연산자 사용 예 2 — 4-7.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 100, b = -200;
06
07    printf(" 상수의 AND 연산 : %d \n", a && b);      ----- AND 연산을 사용한다.
08    printf(" 상수의 OR  연산 : %d \n", __1__);      ----- OR 연산을 사용한다.
09    printf(" 상수의 NOT 연산 : %d \n", __2__);      ----- NOT 연산을 사용한다.
10 }
```

정답_ 1 a || b 2 !a

실행 결과

```
상수의 AND 연산 : 1
상수의 OR  연산 : 1
상수의 NOT 연산 : 0
```

03
비트 연산자

비트 연산자 개요

- **비트 연산자는 정수나 문자 등을 2진수로 변환한 후 각 자리의 비트끼리 연산을 수행**

표 4-5 비트 연산자

연산자	명칭	설명
&	비트 논리곱(AND)	둘 다 1이면 1이다.
|	비트 논리합(OR)	둘 중 하나만 1이면 1이다.
^	비트 배타적 논리합(XOR)	둘이 같으면 0, 둘이 다르면 1이다.
~	비트 부정	1은 0으로, 0은 1로 변경한다.
<<	비트 왼쪽 시프트(이동)	비트를 왼쪽으로 시프트(이동)한다.
>>	비트 오른쪽 시프트(이동)	비트를 오른쪽으로 시프트(이동)한다.

비트 논리곱(&) 연산자 (1/2)

- **10진수를 2진수로 변환한 후 각 비트마다 AND 연산을 수행**

A	B	A&B
0	0	0
0	1	0
1	0	0
1	1	1

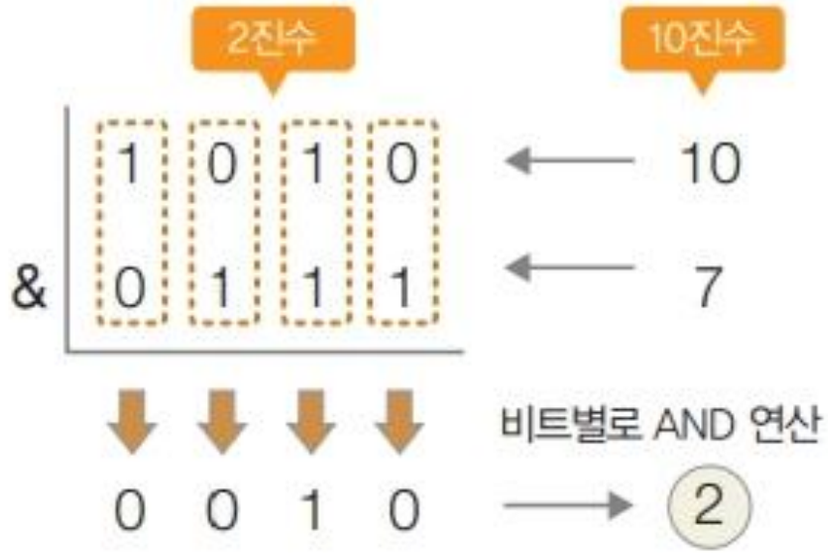


그림 4-6 비트 논리곱의 예

비트 논리곱(&) 연산자 (2/2)

기본 4-8 비트 논리곱 연산자 사용 예 4-8.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    printf(" 10 & 7 = %d \n", 10 & 7);          ----- 10과 7의 비트 논리곱을 수행한다.
06    printf(" 123 & 456 = %d \n", 123 & 456);    ----- 123과 456의 비트 논리곱을 수행한다.
07    printf(" 0xFFFF & 0000 = %d \n ", 0xFFFF & 0000);
                                                  └---- 16진수 FFFF와 0의 비트 논리곱을 수행한다.
08 }
```

실행 결과

```
10 & 7 = 2
123 & 456 = 72
0xFFFF & 0000 = 0
```

- **6행에서는 123의 2진수인 1111 0112과 456의 2진수인 11 1001 0002의 비트 논리곱 결과가 1001 0002이므로 10진수로 72**

- **7행에서는 16진수 FFFF(2진수로는 1111 1111 1111 1111)와 0000(2진수로는 0000 0000 0000 0000)의 비트 논리곱 결과인 0이 출력**

비트 논리합(|) 연산자 (1/2)

- **10진수를 2진수로 변환한 후 각 비트마다 OR 연산을 수행**

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

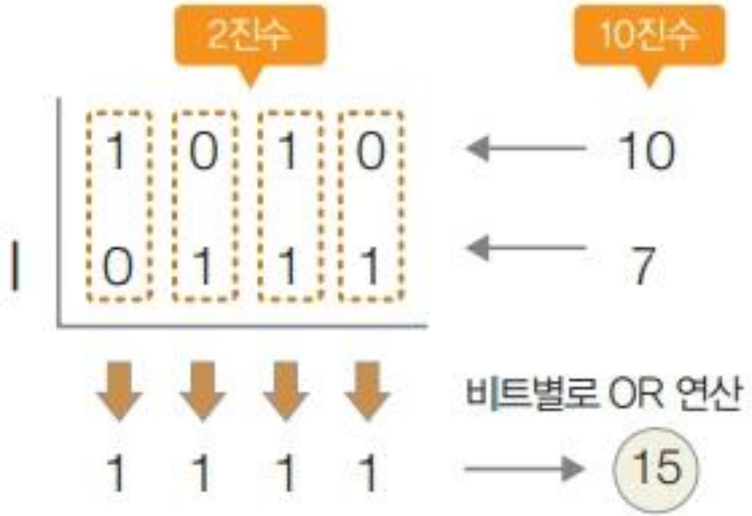


그림 4-7 비트 논리합의 예

비트 논리합(|) 연산자 (2/2)

- **10진수를 2진수로 변환한 후 각 비트마다 OR 연산을 수행**

기본 4-9 비트 논리합 연산자 사용 예 4-9.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    printf(" 10 | 7 = %d \n", 10 | 7);          ----- 10과 7의 비트 논리합을 수행한다.
06    printf(" 123 | 456 = %d \n", 123 | 456);    ----- 123과 456의 비트 논리합을 수행한다.
07    printf(" 0xFFFF | 0000 = %d \n ", 0xFFFF | 0000);
                                                  └---- 16진수 FFFF와 0의 비트 논리합을 수행한다.
08 }
```

실행 결과

```
10 | 7 = 15
123 | 456 = 507
0xFFFF | 0000 = 65535
```

비트 배타적 논리합(^) 연산자 (1/2)

- **10진수를 2진수로 변환한 후 각 비트마다 XOR 연산을 수행**
  - ✓ 두 값이 다르면 1, 같으면 0이 됨

A	B	A^B
0	0	0
0	1	1
1	0	1
1	1	0

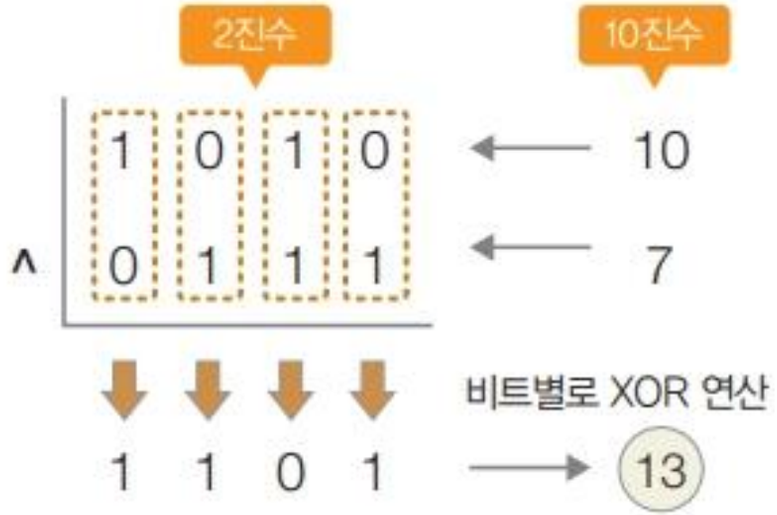


그림 4-8 비트 배타적 논리합의 예

비트 배타적 논리합(^) 연산자 (2/2)

기본 4-10 비트 배타적 논리합 연산자 사용 예 4-10.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    printf(" 10 ^ 7 = %d \n", 10 ^ 7);          ----- 10과 7의 비트 배타적 논리합을 수행한다.
06    printf(" 123 ^ 456 = %d \n", 123 ^ 456);    ----- 123과 456의 비트 배타적 논리합을 수행한다.
07    printf(" 0xFFFF ^ 0000 = %d \n ", 0xFFFF ^ 0000);
                                                  └---- 16진수 FFFF와 0의 비트 배타적 논리합을 수행한다.
08 }
```

실행 결과

```
10 ^ 7 = 13
123 ^ 456 = 435
0xFFFF ^ 0000 = 65535
```

비트 연산 응용 (1/3)

응용 4-11 비트 연산에 마스크를 사용한 예 4-11.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    char  a = 'A', b, c;
06    char  mask = 0x0F;                    ----- 마스크 값(0000 1111₂)을 설정한다.
07
08    printf(" %X & %X = %X \n", a, mask, a & mask);   ----- 'A'와 0x0F의 비트 논리곱을 수행한다.
09    printf(" %X | %X = %X \n", a, mask, a | mask)    ----- 'A'와 0x0F의 비트 논리합을 수행한다.
10
11    mask = 'a' - 'A';                     ----- 'a'와 'A'의 차이는 32이다.
12
13    b = __1__                             ----- 'A'와 마스크(32)의 비트 배타적 논리합을 수행한다.
14    printf(" %c ^ %d = %c \n", a, mask, b);
15    a = __2__                             ----- 'a'와 마스크(32)의 비트 배타적 논리합을 수행한다.
16    printf(" %c ^ %d = %c \n", b, mask, a);
17 }
```

실행 결과

```
41 & F = 1
41 & F = 4F
A ^ 32 = a
a ^ 32 = A
```

정답_ 1 a ^ mask; 2 b ^ mask;

비트 연산 응용 (2/3)

- **응용 4-11 복기**
  - ✓ 6행에서 마스크 값을 16진수 0x0F16로 선언했는데 이는 2진수로 0000 1111
  - ✓ 비트 논리곱 연산을 진행하면 앞의 4비트는 모두 0000이 되고 뒤의 4비트는 A의 원래 값이 그대로 남음

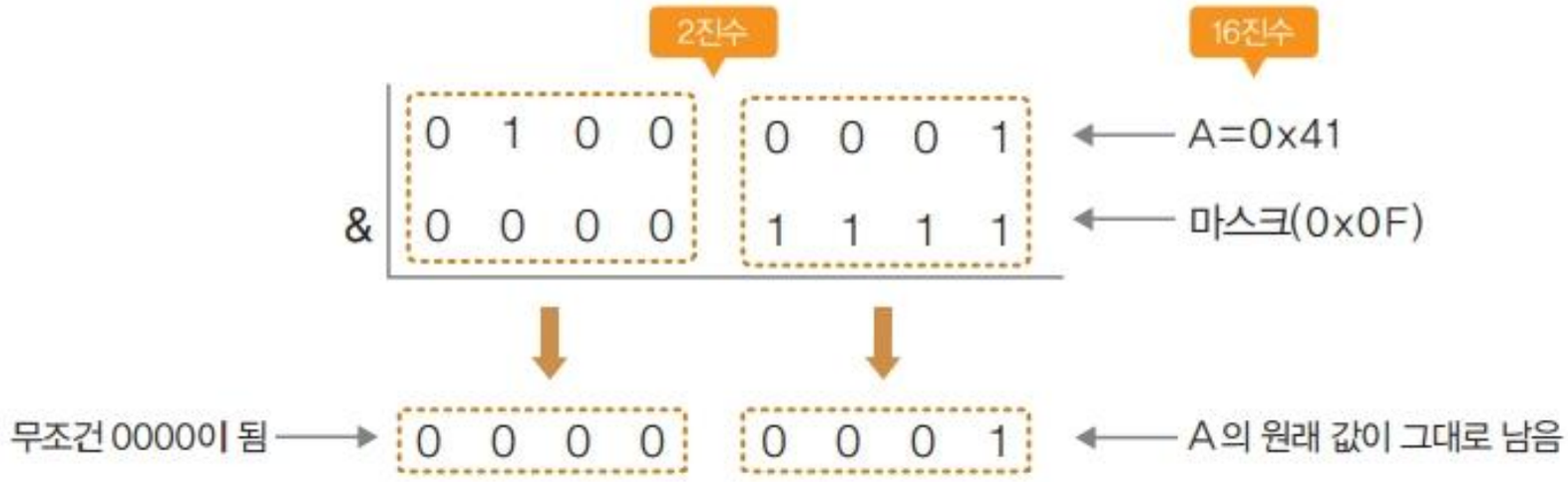


그림 4-9 마스크 0x0F를 사용한 비트 논리곱의 예

비트 연산 응용 (3/3)

- **응용 4-11 복기 (cont'd)**
  - ✓ 9행의 0x0F로 비트 논리합 연산을 하면 앞의 4비트는 A의 원래 값이 그대로 남고 뒤의 4비트는 무조건 1111

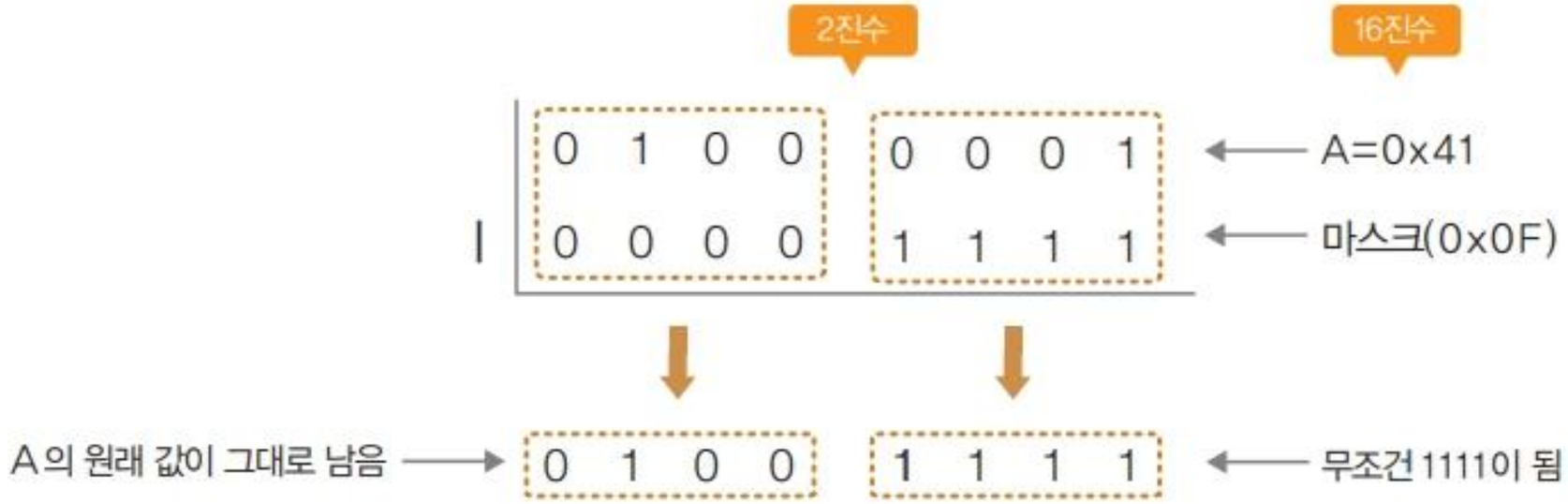


그림 4-10 마스크 0x0F를 사용한 비트 논리합의 예

비트 부정(~) 연산자

- **10진수를 2진수로 변환한 후 각 비트마다 NOT 연산을 수행**
  - ✓ 모든 0은 1로, 모든 1은 0으로 바꿈
  - ✓ 이렇게 반전된 값을 '1의 보수'라고 하며, 그 값에 1을 더한 값을 '2의 보수'라고 함

기본 4-12 비트 부정 연산자 사용 예 4-12.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 12345;
06
07    printf(" %d \n", ~a + 1);          ----- 2의 보수(a 값)를 구한다.
08 }
```

실행 결과

```
-12345
```

비트 왼쪽 시프트(<<) 연산자 (1/2)

- **나열된 비트를 왼쪽으로 시프트(shift)하는 연산자**

그림 4-11 26을 왼쪽으로 두 칸 시프트한 결과

✓ 왼쪽으로 1회 시프트할 때는 2^1을, n회 시프트 할 때는 2^n을 곱하는 셈

비트 왼쪽 시프트(<<) 연산자 (2/2)

기본 4-13 비트 왼쪽 시프트 연산자 사용 예 4-13.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 10;
06    printf("%d 를 왼쪽 1회 시프트하면 %d 이다.\n", a, a << 1);
07    printf("%d 를 왼쪽 2회 시프트하면 %d 이다.\n", a, a << 2);
08    printf("%d 를 왼쪽 3회 시프트하면 %d 이다.\n", a, a << 3);
09 }
```

왼쪽으로 시프트한 결과를 출력한다.

실행 결과

```
10 를 왼쪽 1회 시프트하면 20 이다.
10 를 왼쪽 2회 시프트하면 40 이다.
10 를 왼쪽 3회 시프트하면 80 이다.
```

비트 오른쪽 시프트(〉〉) 연산자 (1/3)

- **나열된 비트를 오른쪽으로 시프트(shift)하는 연산자**

그림 4-12 26을 오른쪽으로 두 칸 시프트한 결과

✓ 오른쪽으로 1회 시프트 할 때는 2^1으로, n회 시프트 할 때는 2^n으로 나누는 셈

비트 오른쪽 시프트(>>) 연산자 (2/3)

기본 4-14 비트 오른쪽 시프트 연산자 사용 예 4-14.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 10;
06    printf("%d 를 오른쪽 1회 시프트하면 %d 이다.\n", a, a >> 1);
07    printf("%d 를 오른쪽 2회 시프트하면 %d 이다.\n", a, a >> 2);
08    printf("%d 를 오른쪽 3회 시프트하면 %d 이다.\n", a, a >> 3);
09    printf("%d 를 오른쪽 4회 시프트하면 %d 이다.\n", a, a >> 4);
10 }
```

----- 오른쪽으로 시프트한 결과를 출력한다.

실행 결과

```
10 를 오른쪽 1회 시프트하면 5 이다.
10 를 오른쪽 2회 시프트하면 2 이다.
10 를 오른쪽 3회 시프트하면 1 이다.
10 를 오른쪽 4회 시프트하면 0 이다.
```

비트 오른쪽 시프트(>>) 연산자 (3/3)

응용 4-15 비트 왼쪽 시프트, 비트 오른쪽 시프트 연산자 사용 예 4-15.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 100, result;
06    int i;
07
08    for(i = 1; i <= 5; i ++)
09    {
10       result = __1__  a << i;
11       printf("%d << %d = %d\n", a, i, result);
12    }
13
14    for(i = 1; i <= 5; i ++)
15    {
16       result =  __2__  a >> i;
17       printf("%d >> %d = %d\n", a, i, result);
18    }
19 }
```

----- 왼쪽 시프트 연산을 5회 반복해서 출력한다.

----- 오른쪽 시프트 연산을 5회 반복해서 출력한다.

실행 결과

```
100 << 1 = 200
100 << 2 = 400
100 << 3 = 800
100 << 4 = 1600
100 << 5 = 3200
100 >> 1 = 50
100 >> 2 = 25
100 >> 3 = 12
100 >> 4 = 6
100 >> 5 = 3
```

정답_ 1 a << i; 2 a >> i;

04
연산자 우선순위

연산자 우선순위 표

표 4-6 연산자 우선순위

우선순위	연산자	명칭	순위가 같을 경우 진행 방향
1	( ) [ ] . ->	1차 연산자	→
2	+ - ++ -- ~ ! * &	단항 연산자(변수 또는 상수 앞에 붙음)	←
3	* / %	산술 연산자	→
4	+ -	산술 연산자	→
5	<< >>	비트 시프트 연산자	→
6	< <= > >=	비교 연산자	→
7	== !=	동등 연산자	→
8	&	비트 연산자	→
9	^	비트 연산자	→
10	|	비트 연산자	→
11	&&	논리 연산자	→
12	||	논리 연산자	→
13	?:	삼항 연산자	→
14	= += -= *= /= %= &= ^= |= <<= >>=	대입 연산자	←
15	,	콤마 연산자	→

05
상수

상수 개요

- **상수의 정의 : 변경이 불가능한 데이터**
  - ✓ 이름을 지니지 않는 리터럴(literal) 상수와 이름이 있는 심볼릭(symbolic) 상수

```
int main(void)
{
    int num = 30 + 40;
    . . . .
}
```

연산을 위해서는 30, 40과 같이 프로그램상에 표현되는 숫자도
메모리 공간에 저장되어야 한다.
이렇게 저장되는 값은 이름이 존재하지 않으니 변경이 불가능한 상수이다.
따라서 리터럴 상수라 한다.

리터럴 상수의 자료형

- **리터럴 상수도 기본 자료형이 지정되어 있음**

```
int main(void)
{
    printf("literal int size: %d \n", sizeof(7));
    printf("literal double size: %d \n", sizeof(7.14));
    printf("literal char size: %d \n", sizeof('A'));
    return 0;
}
```

실행결과

```
literal int size: 4
literal double size: 8
literal char size: 4
```

리터럴 상수도 자료형이 결정되어야 메모리 공간에 저장이 될 수 있다.
위 예제의 실행결과는 다음 사실을 의미한다.

- 정수는 기본적으로 int형으로 표현된다.
- 실수는 기본적으로 double형으로 표현된다.
- 문자는 기본적으로 int형으로 표현된다.

접미사를 이용한 다양한 상수의 표현 (1/2)

```
int main(void)
{
    float num1 = 5.789;    // 경고 메시지 발생
    float num2 = 3.24 + 5.12;    //경고 메시지 발생
    return 0;
}
```

실수는 double형 상수로 인식이 되어
데이터 손실에 대한 경고 메시지 발생

```
float num1 = 5.789f;        // 경고 메시지 발생 안 함
float num2 = 3.24F + 5.12F;    // 소문자 f 대신 대문자 F를 써도 된다!
```

접미사를 통해서 상수의 자료형을 변경
할 수 있다.

접미사를 이용한 다양한 상수의 표현 (2/2)

- **명시적으로 상수의 자료형을 표현**

접미사	자료형	사용의 예
U	unsigned int	unsigned int n = 1025U
L	long	long n = 2467L
UL	unsigned long	unsigned long n = 3456UL
LL	long long	long long n = 5768LL
ULL	unsigned long long	unsigned long long n = 8979ULL

접미사	자료형	사용의 예
F	float	float f = 3.15F
L	long double	long double f = 5.789L

심볼릭 상수 : const 상수

- **자주 사용되거나 고유한 상수에 변수명과 같은 상수명을 붙일 수 있음**

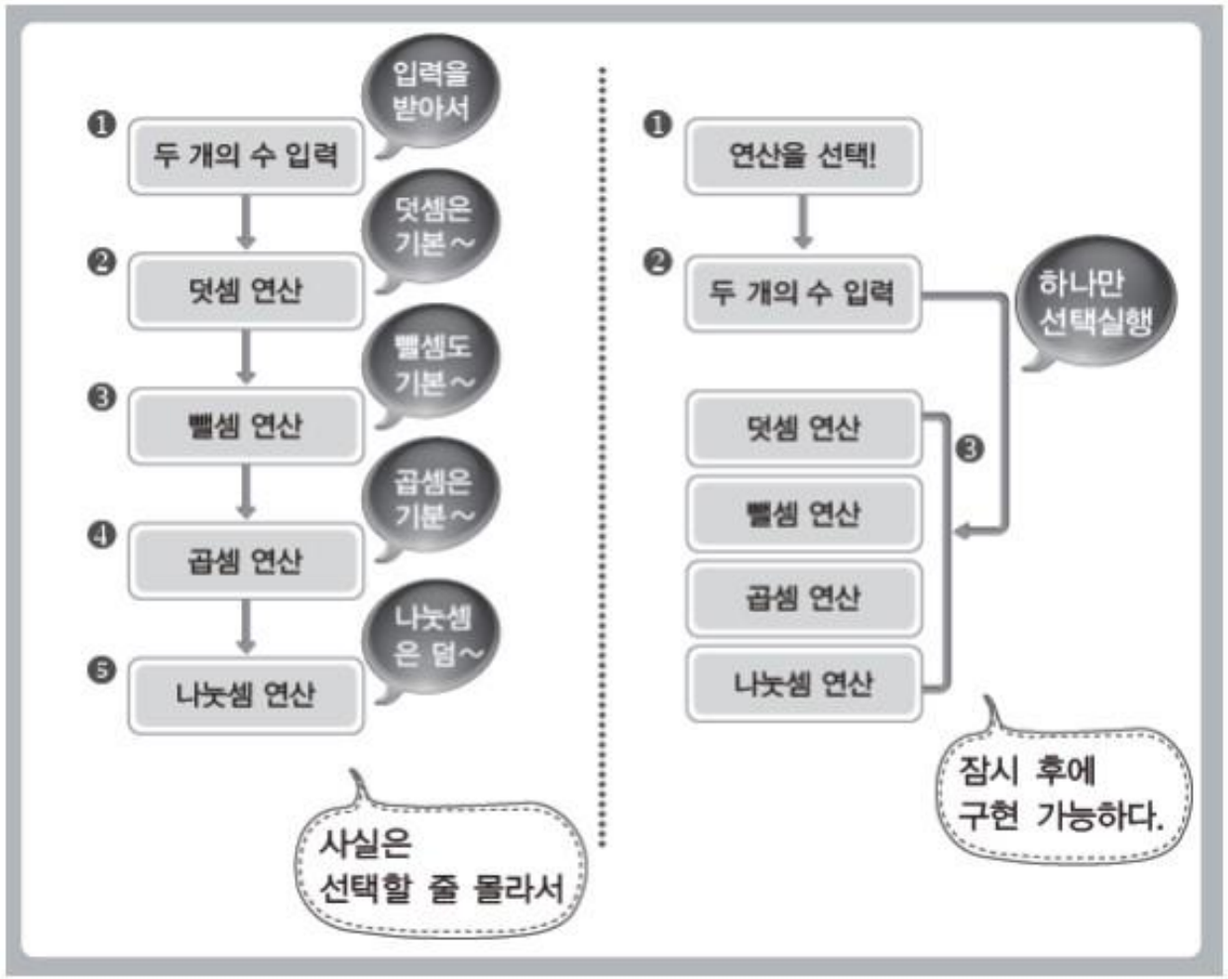
```
int main(void)
{
    const int MAX=100;     // MAX는 상수! 따라서 값의 변경 불가!
    const double PI=3.1415;    // PI는 상수! 따라서 값의 변경 불가!
    . . . .
}
```

```
int main(void)
{
    const int MAX;     // 쓰레기 값으로 초기화 되어버림
    MAX=100;    // 값의 변경 불가! 따라서 컴파일 에러 발생!
    . . . .
}
```

06
if 문

조건식 : 프로그램 흐름 제어

- **흐름의 분기가 필요한 이유 : 계산기 프로그램의 예**



분기하지 못하면 프로그램 사용자는
사칙연산 중 하나를 선택하지 못한다!

기본 if문 (1/5)

- **조건문(선택 제어문)에는 if문과 switch문이 있음**

  ➔ 조건에 따라서 다른 내용이 실행되도록 실행의 흐름을 제어하는 명령문

- **'if'라는 단어의 의미는 '만약 ~라면'으로, C에 서도 '만약' 이라는 어떤 조건을 내세울 때 if문을 사용**

  ✓ 조건식이 참이면 '실행할 문장'을 실행하고 조건식이 거짓이면 실행하지 않음

```
if(조건식)
  실행할 문장;
```

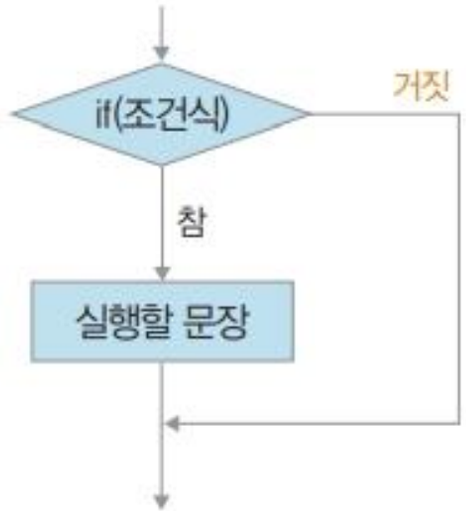


그림 5-1 if문의 형식과 순서도

기본 if문 (2/5)

- **a가 100보다 작으면 메시지를 출력**

기본 5-1 기본 if문 사용 예 1

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 99;
06
07    if(a < 100)                    ---- a가 100보다 작으므로 참이다.
08       printf("100보다 작군요..\n");
09 }
```

실행 결과

```
100보다 작군요..
```

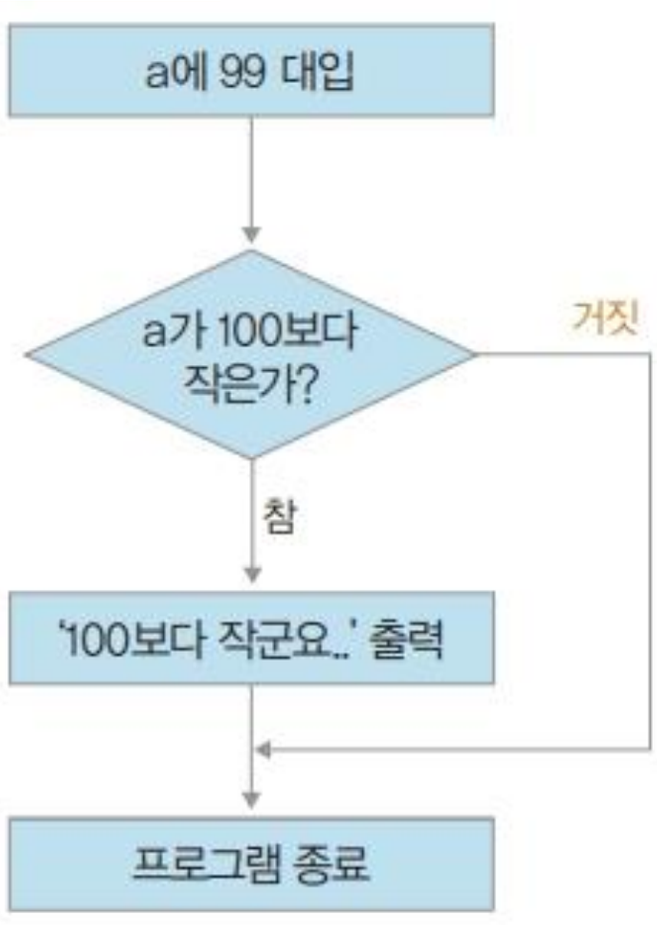


그림 5-2 [기본 5-1]의 실행 과정

기본 if문 (3/5)

- **a가 100보다 작으면 메시지를 출력**

기본 5-2 기본 if문 사용 예 2 5-2.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 200;
06
07    if(a < 100)
08       printf("100보다 작군요..\n");
09       printf("거짓이므로 이 문장은 안 보이겠죠?\n");
10
11    printf("프로그램 끝! \n");
12 }
```

----- 7행이 참일 경우에 수행할 것으로 예상된다.

------ 7행이 거짓이면 8~9행을 수행하지 않고 11행만 수행할 것으로 예상된다.

실행 결과

```
거짓이므로 이 문장은 안 보이겠죠?
프로그램 끝!
```

기본 if문 (4/5)

- **기본 5-2 복기**
  - ✓ 5행의 a가 200이므로 7행의 조건식은 거짓이 되어 8~9행 을 건너뛰고 11행을 실행할 것으로 예상했는데 9행도 실행
  - ➔ 이는 바로 '행 바꿈의 함정' 때문, 다음과 같이 행 바꿈을 수정해서 다시 실행

```
01  #include <stdio.h>
02
03  void main( )
04  {
05     int a = 200;
06
07     if(a < 100)
08        printf("100보다 작군요..\n");
09
10     printf("거짓이므로 이 문장은 안 보이겠죠?\n");
11     printf("프로그램 끝! \n");
12  }
```

기본 if문 (5/5)

- **기본 5-2에서 8~9행을 if문의 실행문으로 가져가고 싶다면?**

기본 5-3 기본 if문 사용 예 3 5-3.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 200;
06
07    if(a < 100)
08    {
09       printf("100보다 작군요..\n");
10       printf("거짓이므로 앞의 문장은 안 보이겠죠?\n");
11    }
12
13    printf("프로그램 끝! \n");
14 }
```

----- 7행이 참이면 블록으로 감싼 부분이 모두 수행된다.

실행 결과

```
프로그램 끝!
```

if~else문 (1/4)

- **참일 때 실행하는 문장과 거짓일 때 실행하는 문장이 다른 경우 ➔ if~else문을 사용**
  - ✓ 조건식이 참이라면 '실행할 문장 1'을 실행하고, 그렇지 않으면 '실행할 문장 2' 실행
  - ➔ if문을 2개 사용하여 대체할 수 있으나 if-else를 사용하면 가독성 향상

```
if(조건식)
  실행할 문장 1;
else
  실행할 문장 2;
```

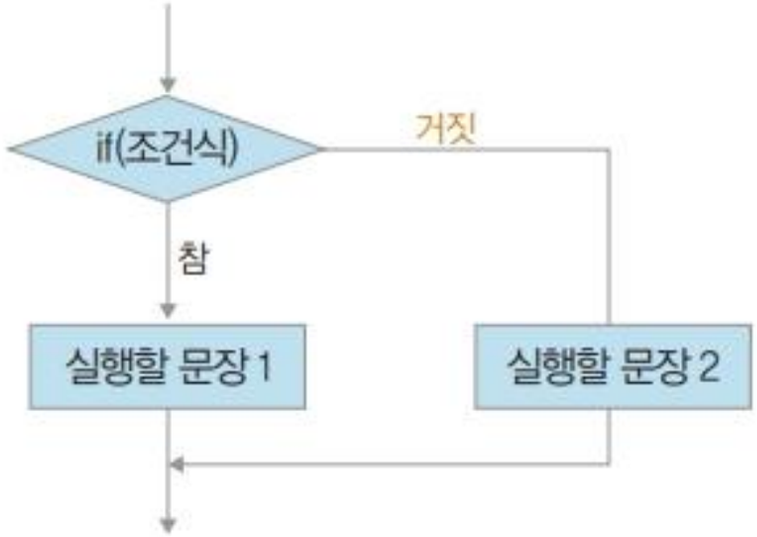


그림 5-3 if~else문의 형식과 순서도

if~else문 (2/4)

- **a가 100보다 크거나 또는 작거나 같을 경우 다른 메시지 출력**

기본 5-4 if~else문 사용 예

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 200;
06
07    if(a < 100)
08       printf("100보다 작군요..\n");      ----- 7행이
09    else
10       printf("100보다 크군요..\n");      ----- 7행이 거짓이면(a가 100보다 크거나 같으면) 실행된다.
11 }
```

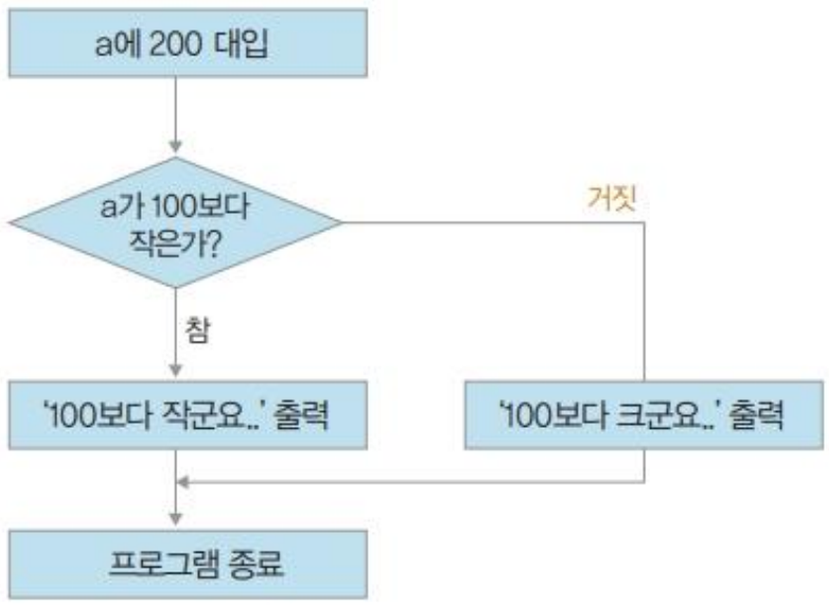


그림 5-4 [기본 5-4]의 실행 과정

실행 결과

```
100보다 크군요..
```

if~else문 (3/4)

- **블록을 이용하여 a의 크기에 따라 다른 메시지 출력**

기본 5-5 블록을 활용한 if~else문 사용 예 1 5-5.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int a = 200;
06
07    if(a < 100)
08    {                                  ----- 7행이 참이면(a가 100보다 작으면) 실행된다.
09       printf("100보다 작군요..\n");
10       printf("참이면 이 문장도 보이겠죠?\n");
11    }
12    else
13    {                                  ----- 7행이 거짓이면(a가 100보다 크거나 같으면)
14       printf("100보다 크군요..\n");          실행된다.
15       printf("거짓이면 이 문장도 보이겠죠?\n");
16    }
17
18    printf("프로그램 끝! \n");
19 }
```

if~else문 (4/4)

- **입력 받은 정수의 짝수 또는 홀수 여부를 출력**

응용 5-6 블록을 활용한 if~else문 사용 예 2 5-6.c

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a;
06
07    printf("정수를 입력하세요 : ");
08    scanf("%d", &a);                    ----- 정수를 입력받는다.
09
10    if(__1__ == 0)                      ----- 입력된 값을 2로 나눈 나머지값이 0일 경우
11    {                                   ----- 짝수이면(2로 나눈 나머지값이 0이면) 실행된다.
12       printf("짝수를 입력했군요..\n");
13    }
14    else
15    {                                   ----- 홀수이면(2로 나눈 나머지값이 0이 아니면) 실행된다.
16       printf("홀수를 입력했군요..\n");
17    }
18 }
```

실행 결과

```
정수를 입력하세요 : 125
홀수를 입력했군요..
```

07

중첩 if문

중첩 if문 (1/6)

- **if문을 한 번 실행하고, 그 결과에 다시 if문을 실행하는 것을 '중첩 if문(또는 중복 if문)'이라고 함**
  - ✓ '조건식 1'이 참이면 'if(조건식 2)'를 실행하고 '조건식 2'가 참이면 '실행할 문장 1'을 실행하는 식

```
if(조건식 1){
  if(조건식 2)
     실행할 문장 1;
  else
     실행할 문장 2;
}
else
     실행할 문장 3;
```

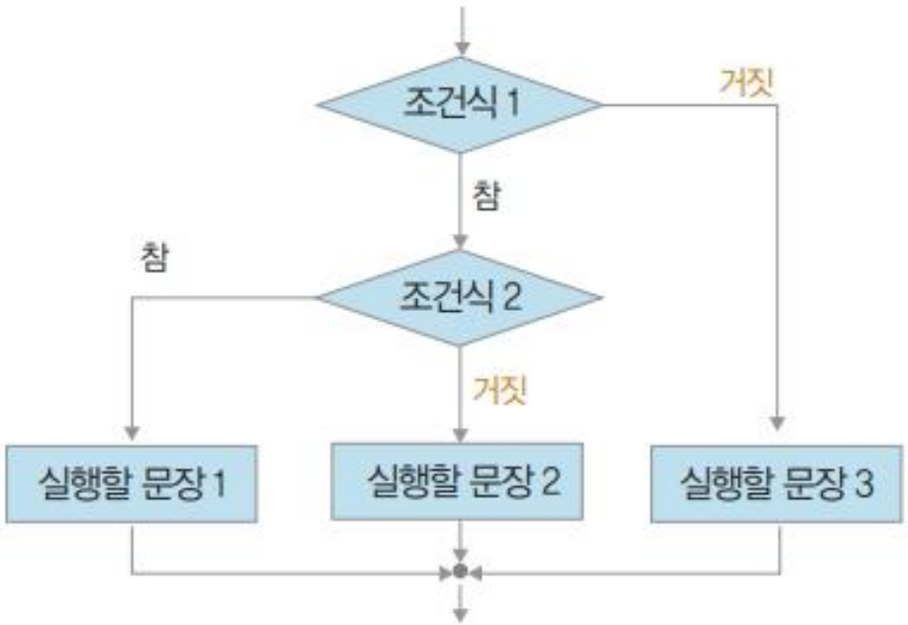


그림 5-5 중첩 if문의 형식과 순서도

중첩 if문 (2/6)

- **50보다 크고 100보다 작으면 메시지를 출력**

기본 5-7 중첩 if문 사용 예 1 — 5-7.c

```
01 #include <stdio.h>
02
03 void main( )
04 {
05    int  a = 75;
06
07    if(a > 50)
08    {                                  ----- 7행이 참이면(a가 50보다 크면) 실행된다.
09       if(a < 100)
10       {                               ------ 7행이 참이고(a가 50보다 크고) 9행이 참이면(a가 100보다 작으면) 실행된다.
11          printf("50보다 크고 100보다 작군요..\n");
12       }
13       else
14       {                               ------ 7행이 참이고(a가 50보다 크고) 9행이 거짓이면(a가 100보다 크거나 같으면) 실행된다.
15          printf("와~ 100보다 크군요..\n");
16       }
17    }
```

중첩 if문 (3/6)

- **50보다 크고 100보다 작으면 메시지를 출력**

```
18    else
19    {
20        printf("에게~ 50보다 작군요..\n");
21    }
22 }
```

---- 7행이 거짓이면(a가 50보다 작거나 같으면) 실행된다.

실행 결과

```
50보다 크고 100보다 작군요..
```

중첩 if문 (4/6)

- **입력 받은 성적의 범위에 따라 학점을 출력**

응용 5-8 중첩 if문 사용 예 2 5-8.c

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a;
06
07    printf("점수를 입력하세요 : ");
08    scanf("%d", &a);                ----- 점수를 입력한다(100점 만점).
09
```

중첩 if문 (5/6)

- **입력 받은 성적의 범위에 따라 학점을 출력**

```
10   if(__1__)                  ---- 입력한 점수가 90점 이상이면 A 학점을 출력한다.
11     printf("A");
12   else
13     if(__2__)                ---- 입력한 점수가 80점 이상이면 B 학점, 70점 이상이면
14       printf("B");                C 학점, 60점 이상이면 D 학점을 출력한다.
15     else
16       if(a >= 70)
17         printf("C");
18       else
19         if(a >= 60)
20           printf("D");
21         else                   ---- 입력한 점수가 60점 미만이면 F 학점을 출력한다.
22           printf("F");
23
24   printf(" 학점 입니다. \n");
25 }
```

빈칸 1 a >= 90 빈칸 2 a >= 80

실행 결과

```
점수를 입력하세요 : 77
C 학점 입니다.
```

중첩 if문 (6/6)

- **응용 5-8 복기**

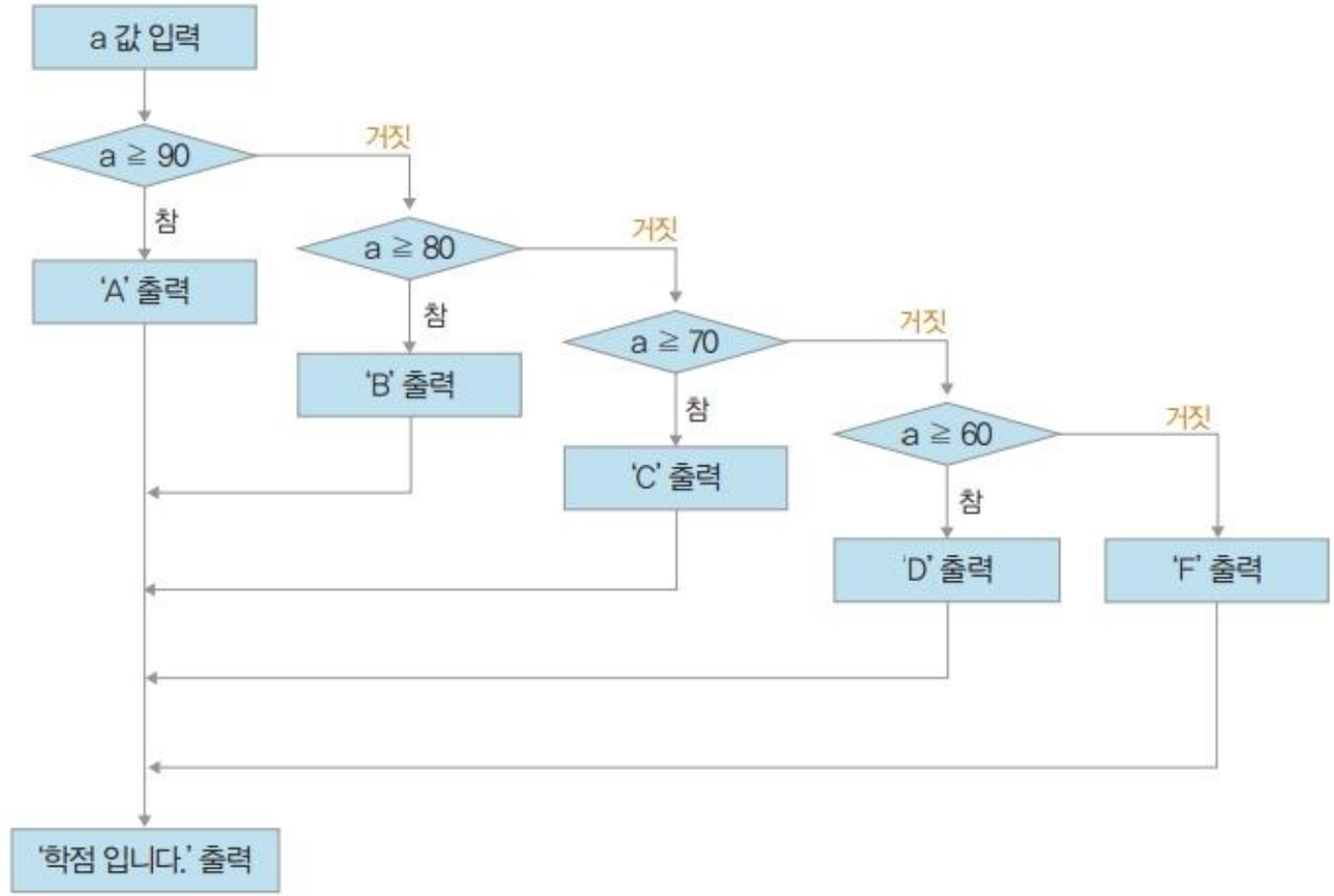


그림 5-6 [응용 5-8]의 실행 과정

예 제

[예제 01] 입력된 두 실수의 산술 연산

예제 설명 실수를 입력받아 두 수의 다양한 연산을 출력하는 프로그램이다.

힌트_ 나머지를 구할 때는 강제 형 변환을 사용한다.

%를 출력하려면?
"%%" 사용

실행 결과

```
첫 번째 계산할 값을 입력하세요 ==> 10
두 번째 계산할 값을 입력하세요 ==> 20
 10.00 + 20.00 = 30.00
 10.00 - 20.00 = -10.00
 10.00 * 20.00 = 200.00
 10.00 / 20.00 =  0.50
 10 % 20 = 10
```

[예제 01] 입력된 두 실수의 산술 연산

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    float a, b;                                  ---- 실수형 변수를 선언한다.
06    float result;
07
08    printf("첫 번째 계산할 값을 입력하세요 ==> ");
09    scanf("%f", &a);                             ---- 실수를 입력받는다.
10    printf("두 번째 계산할 값을 입력하세요 ==> ");
11                                                 ---- 실수를 입력받는다.
12
13    result = a + b;                              ---- 실수의 덧셈을 수행한다.
14    printf(" %5.2f + %5.2f = %5.2f \n", a, b, result);
15    result = a - b;                              ---- 실수의 뺄셈을 수행한다.
16    printf(" %5.2f - %5.2f = %5.2f \n", a, b, result);
17    result = a * b;                              ---- 실수의 곱셈을 수행한다.
18    printf(" %5.2f * %5.2f = %5.2f \n", a, b, result);
19    result = a / b;                              ---- 실수의 나눗셈을 수행한다.
20    printf(" %5.2f / %5.2f = %5.2f \n", a, b, result);
21                                                 ---- 나머지 연산을 위해 실수를 정수로 강제 형 변환한다.
22
23 }
```

[예제 02] 윤년 계산 프로그램

예제 설명 입력된 연도가 윤년인지 계산하는 프로그램이다.

❶ 4로 나누어 떨어지고 100으로 나누어 떨어지지 않으면 윤년이다.
❷ 400으로 나누어 떨어지는 해도 윤년에 포함된다.

실행 결과

```
연도를 입력하세요. : 2024
2024 년은 윤년입니다.
```

[예제 02] 윤년 계산 프로그램

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int year;
06
07    printf("연도를 입력하세요. : ");
08                                          ---- 계산할 연도를 입력한다.
09
10    if(                                                          )
11       printf("%d 년은 윤년입니다. \n", year);
12    else
13       printf("%d 년은 윤년이 아닙니다. \n", year);
14 }
```

윤년은 입력한 연도가 4로 나누어 떨어지고
100으로는 나누어 떨어지지 않아야 한다.
또는 400으로 나누어 떨어져도 된다.

[예제 03] 입력된 두 수의 산술 연산

예제 설명 단순 if문을 활용하여 두 수의 +, −, *, /, % 연산을 수행하는 프로그램이다.

실행 결과

```
첫 번째 수를 입력하세요 : 5
계산할 연산자를 입력하세요 : *
두 번째 수를 입력하세요 : 7
5 * 7 = 35 입니다.
```

[예제 03] 입력된 두 수의 산술 연산

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a, b;                                  ----- 입력받을 정수 2개와 연산자
06    char ch;                                         문자 1개를 선언한다.
07
08    printf("첫 번째 수를 입력하세요 : ");
09    scanf("%d", &a);                           ----- 계산할 첫 번째 숫자를
10    printf("계산할 연산자를 입력하세요 : ");            입력한다.
11    scanf(" %c", &ch);                         ----- 연산자를 입력한다.
12    printf("두 번째 수를 입력하세요 : ");
13    scanf("%d", &b);                           ----- 계산할 두 번째 숫자를
14                                                     입력한다.
15    if(ch == '+')                              ----- 기본 if문을 사용한 연산을
16       printf("%d + %d = %d 입니다. \n", a, b, a+b);     수행한다.
17
18    if(ch == '-')
19       printf("%d - %d = %d 입니다. \n", a, b, a-b);
20
21
22
23
24
25
26
27
28
29 }
```

%c앞에 공백이 있음

[예제 04] 중복 if문을 이용한 산술 연산

예제 설명 중복 if문을 활용하여 두 수의 +, −, *, /, % 연산을 수행하는 프로그램이다.

실행 결과

```
첫 번째 수를 입력하세요 : 12
계산할 연산자를 입력하세요 : /
두 번째 수를 입력하세요 : 5
12 / 5 = 2.400000 입니다.
```

```
첫 번째 수를 입력하세요 : 88
계산할 연산자를 입력하세요 : &
두 번째 수를 입력하세요 : 77
연산자를 잘못 입력했습니다.
```

- **If문을 중첩해서 사용 시 if~ else if~ else 문을 사용**
  - ✓ if ~ else{ if ~ else } ➔ if ~ else if ~ else로 사용 가능

[예제 04] 중복 if문을 이용한 산술 연산

```
01 #define  _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03 void main( )
04 {
05    int a, b;
06    char ch;
07
08    printf("첫 번째 수를 입력하세요 : ");
09    scanf("%d", &a);
10    printf("계산할 연산자를 입력하세요 : ");
11    scanf(" %c", &ch);
12    printf("두 번째 수를 입력하세요 : ");
13    scanf("%d", &b);
14
15    if(ch == '+')                          ----- 중복 if문을 사용한
16       printf("%d + %d = %d 입니다. \n", a, b, a+b);   연산을 수행한다.
17
18
19
20
21
22
23
24
25    else                                   ----- +, -, *, /, % 외의
26       printf("연산자를 잘못 입력했습니다. \n");        문자를 입력하면
27 }                                                 오류 메시지를 보여준다.
```

Q & A