C 언어

연산자

- 산술연산자
- 비교연산자
- 논리연산자
- 증감연산자

산술 연산자

연산자	기호	의미	예
덧셈	+	x와 y를 더한다	x+y
뺄셈	-	x에서 y를 뺀다.	x-y
곱셈	*	x와 y를 곱한다.	x*y
나눗셈	/	x를 y로 나눈다.	x/y
나머지	%	x를 y로 나눌 때의 나머지값	x%y

산술 연산자의 종류

산술 연산자

- Binary expression
  - 형식 : 피연산자 – 연산자 – 피연산자 (operand – operator – operand)
  - Operator가 operand사이에 있음

  - Binary expression 예
    - a+7, 3+4, b–11 등 ← additive expression (+, –)
    - 10*12, a /4, 5%2 등 ← multiplicative expression (*, /, %)
      - %(modulo)는 나머지를 구하는 연산자

실습예제

INPUT

- 2개의 정수 입력
- 원하는 연산 (1 = 더하기, 2=빼기, 3=곱하기, 4=나누기)

Function

- 각 연산을 수행하고 결과값을 리턴 하는 함수
  - 더하기 : a + b
  - 빼기 : a - b
  - 곱하기 : a * b
  - 나누기 : a / b

증감 연산자

- 증가연산(prefix, postfix)과 일반산술연산(infix)의 비교

```c
#include <stdio.h>

int main(void){
        int x = 10;

        printf("%d\n", x++);
        printf("%d\n", x);
        printf("%d\n", --x);
        printf("%d\n", x);

        return;
}
```

printf	X
	10
10	11
11	11
10	10
10	10

```c
#include <stdio.h>

int main(void){
        int x = 10;

        printf("%d\n", x+1);
        printf("%d\n", x);
        printf("%d\n", x-1);
        printf("%d\n", x);

        return;
}
```

printf	X
	10
11	10
10	10
9	10
10	10

증감 연산자

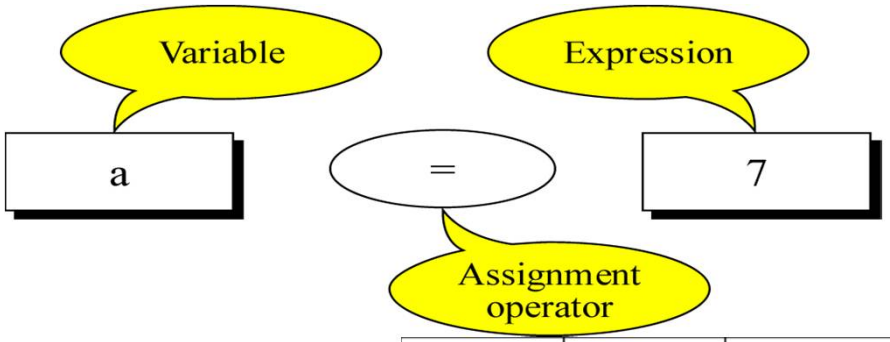
- 증감연산 사용 이유
  - 증감연산자를 이용하면 프로그램 형태가 간결
  - 기계어 코드와 일대일 대응되므로 실행속도가 개선

- 주의점
  - 연산자의 위치에 따라 evaluation value가 다르므로 주의 요구
  - 증감연산자는 ++, --자체가 연산자 기호 ➔ 중간에 공백이 들어가면 안됨
  - 산술연산이나 관계, 논리연산보다 그 평가를 먼저 한다.
  - 증감연산자는 피연산자로 변수를 사용할 수 있지만, 상수나 일반 수식을 피연산자로 사용 불가능
    - 다음과 같은 수식은 잘못된 수식

```
int a = 10;
++300;  /* 상수에는 증가 연산자를 사용할 수 없다 */
(a+1)--; /* 일반 수식에는 증가 연산자를 사용할 수 없다 */
```

대입연산자

- Assignment expression
  - 수학에서는 “우측의 값과 좌측의 값이 같다”라는 의미이나,
    C 언어에서는 “우측의 값을 좌측의 저장 장소에 저장하라”라는 의미



  - 분류
    - simple assignment ←
    - 예) a=5, b=x+1, i=i+1등

Contents of Variable x	Contents of Variable y	Expression	Value of Expression	Result of Expression
10	5	x = y + 2	7	x = 7
10	5	x = x / y	2	x = 2
10	5	x = y % 4	1	x = 1

    - Compound assignment
    - 예) x+=y, x*=y, x/=y 등 ←

Compound Expression	Equivalent Simple Expression
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y
x += y	x = x + y
x -= y	x = x - y

논리연산자

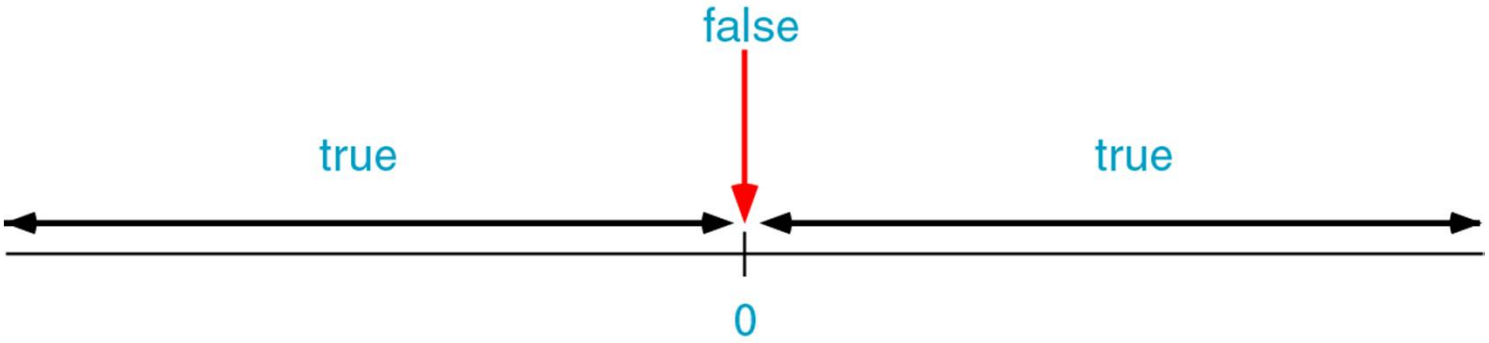
선택(selection) 은 어떤 상황에서 판단을 하여 다른 흐름을 만드는 방법

선택은 논리적 판단에 기반함

그러나 전통적으로 C에는 논리형(logical type)이 없음

- 데이터 값이 '0(zero)'이면 'false', (nonzero) 이면 'true'로 봄
- C99에서 소개된 Boolean 데이타 타입 사용

정수, 실수 등에서는 0, 문자에서는 '\0'가 false에 해당

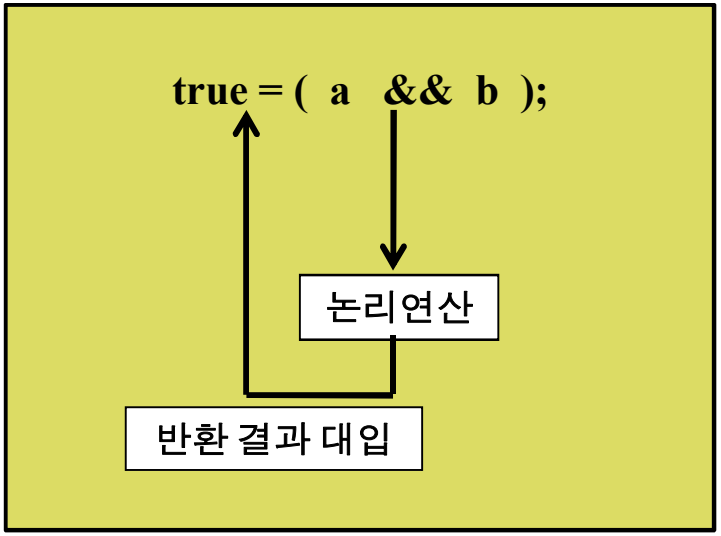


논리연산자

예제프로그램 – 논리연산자

```c
#include <stdio.h>
#include <stdbool.h>

int main(void){
        bool a = true;
        bool b = true;
        bool c = false;

        printf("    %2d AND     %2d : %2d\n", a, b, a && b);
        printf("    %2d AND     %2d : %2d\n", a, c, a && c);
        printf("    %2d AND     %2d : %2d\n", c, a, c && a);
        printf("    %2d OR      %2d : %2d\n", a, c, a || b);
        printf("    %2d OR      %2d : %2d\n", c, a, c || a);
        printf("    %2d OR      %2d : %2d\n", c, c, c || c);
        printf("NOT %2d AND NOT %2d : %2d\n", a, c, !a && !c);
        printf("NOT %2d AND     %2d : %2d\n", a, c, !a && c);
        printf("    %2d AND  NOT%2d : %2d\n", a, c, a && !c);

        return 0;
}
```

a 는 true 이므로 1,
b 도 true 이므로 1.




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

논리연산자

논리연산자

- **and(&&), or(||), not(!)**을 표현하는 연산자
- 두 피연산자의 참과 거짓에 따라 연산의 결과값을 결정
  - **&&** : 피연산자가 모두 참이면 **true**를 반환
  - **||** : 피연산자 중 하나라도 참이면 **true**를 반환
  - **!** : 피연산자가 **true**이면 **false**를, **false**이면 **true**를 반환

논리 연산자의 연산 방법

- 연산에 참여하는 자료 값 중 **0**은 거짓**(false)**을 의미

  1(0이 아닌 값**)**은 참**(true)**을 의미
- 정수만을 이용하는 것은 아니고

  실수나 다른 유형의 자료 값에도 이용
  - 평가의 결과는 반드시 **0(false)**이거나 **1(true)**

논리연산자

연산자 기호	사용 예	의미
&&	x && y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓
||	x || y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓
!	!x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참

x	y	&& 연산	|| 연산	! 연산
		x && y	x || y	! x
0(false)	0	false(0)	false(0)	true(1)
0	1(true)	false(0)	true(1)	true(1)
1	0	false(0)	true(1)	false(0)
1	1	true(1)	true(1)	false(0)

<Logical Operators Truth Table>

관계연산자

관계연산자

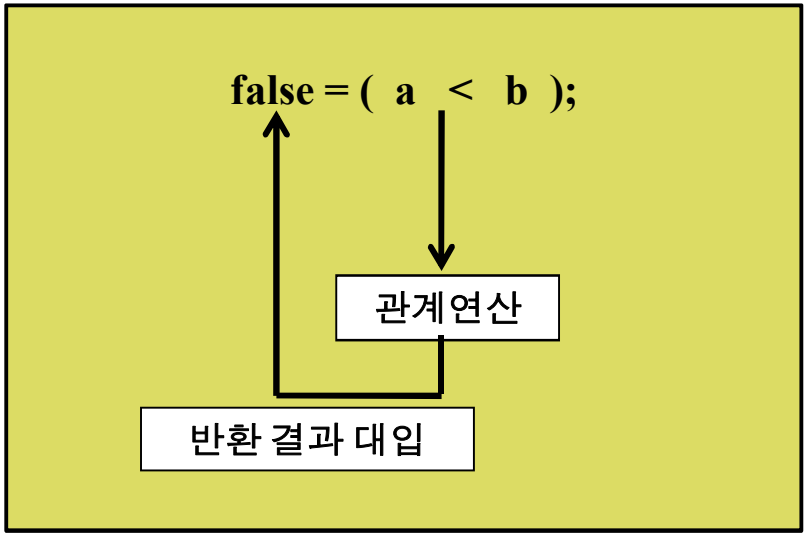
- 두 값의 크기를 비교하는 연산자
  - <, >, >=, <=, ==, !=
  - 두 기호의 순서가 바뀌면 에러 발생(=<, =!)

연산자 기호	의미	사용 예
==	x와 y가 같은가?	x == y
!=	x와 y가 다른가?	x != y
>	x가 y보다 큰가?	x > y
<	x가 y보다 작은가?	x < y
>=	x가 y보다 크거나 같은가?	x >= y
<=	x가 y보다 작거나 같은가?	x <= y

관계연산자

예제프로그램 – 관계연산자

```c
#include <stdio.h>

int main(void){
        int a = 5;
        int b = -3;

        printf("%2d  < %2d is %2d\n",a,b,a <b);
        printf("%2d == %2d is %2d\n",a,b,a==b);
        printf("%2d != %2d is %2d\n",a,b,a!=b);
        printf("%2d  > %2d is %2d\n",a,b,a >b);
        printf("%2d <= %2d is %2d\n",a,b,a<=b);
        printf("%2d >= %2d is %2d\n",a,b,a>=b);

        return 0;
}
```




```
gr120100206@cspro:~/CProgramming/chap5$ vi comparative_operator.c
gr120100206@cspro:~/CProgramming/chap5$ gcc -o comparative_operator comparative_operator.c
gr120100206@cspro:~/CProgramming/chap5$ ./comparative_operator
 5  < -3 is  0
 5 == -3 is  0
 5 != -3 is  1
 5  > -3 is  1
 5 <= -3 is  0
 5 >= -3 is  1
```

관계연산자

Original Expression	! (x < y)	! (x > y)	! (x != y)	! (x <= y)	! (x >= y)	! (x == y)
Simplified Expression	x >= y	x <= y	x == y	x > y	x < y	x != y

<Example of simplifying Operator Complements>

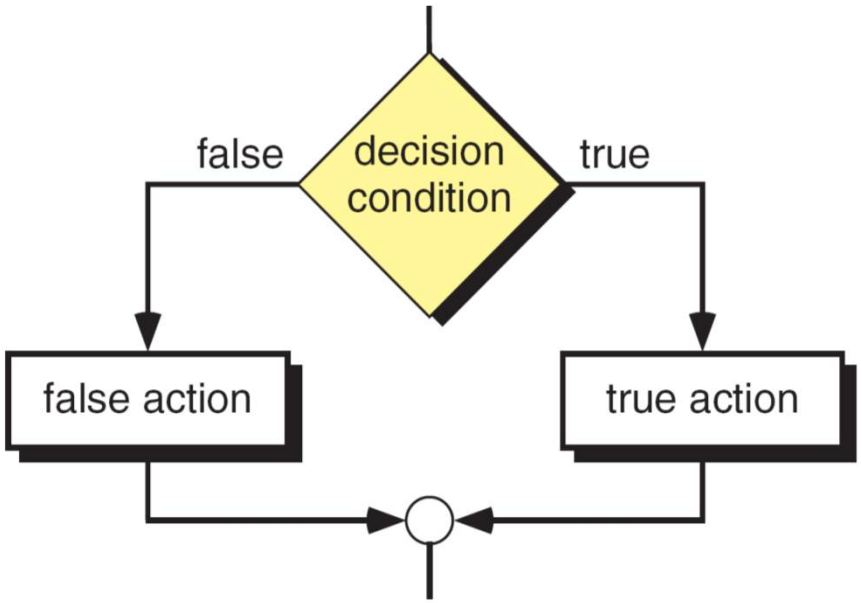
Two-Way Selection

컴퓨터의 basic decision statement는 two-way selection 임

Decision condition의 결과에 따라 제어 흐름 바뀜

two-way decision의 로직

- condition이 true이면 true action을 취함
- condition이 false이면 false action을 취함

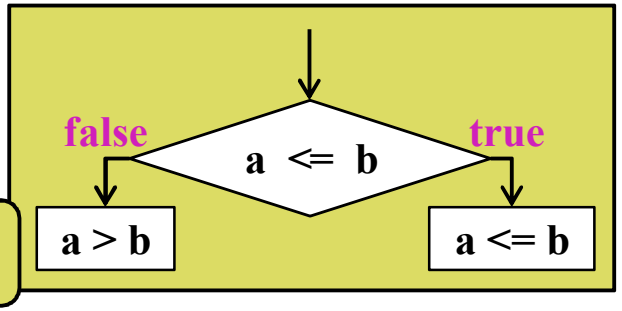


조건문 if, else

예제 프로그램 – if…else 문의 예

```c
#include <stdio.h>

int main(void){
        int a;
        int b;

        printf("Please enter two integers: ");
        scanf("%d %d",&a, &b);

        if(a <= b)
                printf("%d <= %d\n",a,b);
        else
                printf("%d > %d\n",a,b);

        return 0;
}
```

사용자로부터 임의의 값을 입력 받는다.

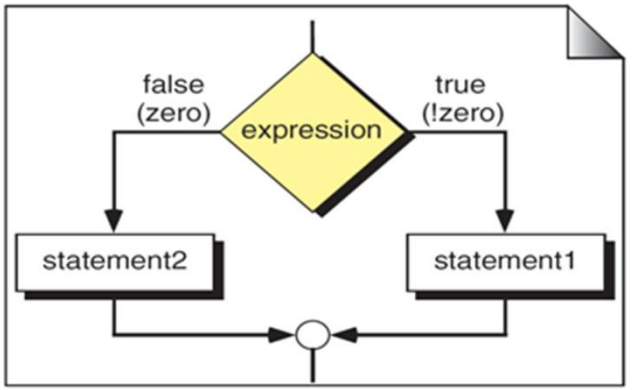



```
gr120100206@cspro:~/CProgramming/chap5$ vi two_way_selection.c
gr120100206@cspro:~/CProgramming/chap5$ gcc -o two_way_selection two_way_selection.c
gr120100206@cspro:~/CProgramming/chap5$ ./two_way_selection
Please enter two integers: 10 15
10 <= 15
gr120100206@cspro:~/CProgramming/chap5$ ./two_way_selection
Please enter two integers: 21 5
21 > 5
```

조건문 if, else

if (condition expression) statements

- "실행 조건"을 만족(true)하는 경우
  - "statement-1" 실행
- "실행 조건"을 불만족(false)하는 경우
  - "statement-2" 실행



(a) Logical Flow

•While 문은 조건이 만족되는 동안 계속해서 반복 실행
•If 문은 선택적으로 한 번만 실행하고 넘어감

If…else문의 statement는 들여쓰기 적용

- 관리자가 코드를 인식하기 쉽게 하기 위해 사용
- 실제 컴퓨터가 들여쓰기를 인식하는 것은 아님

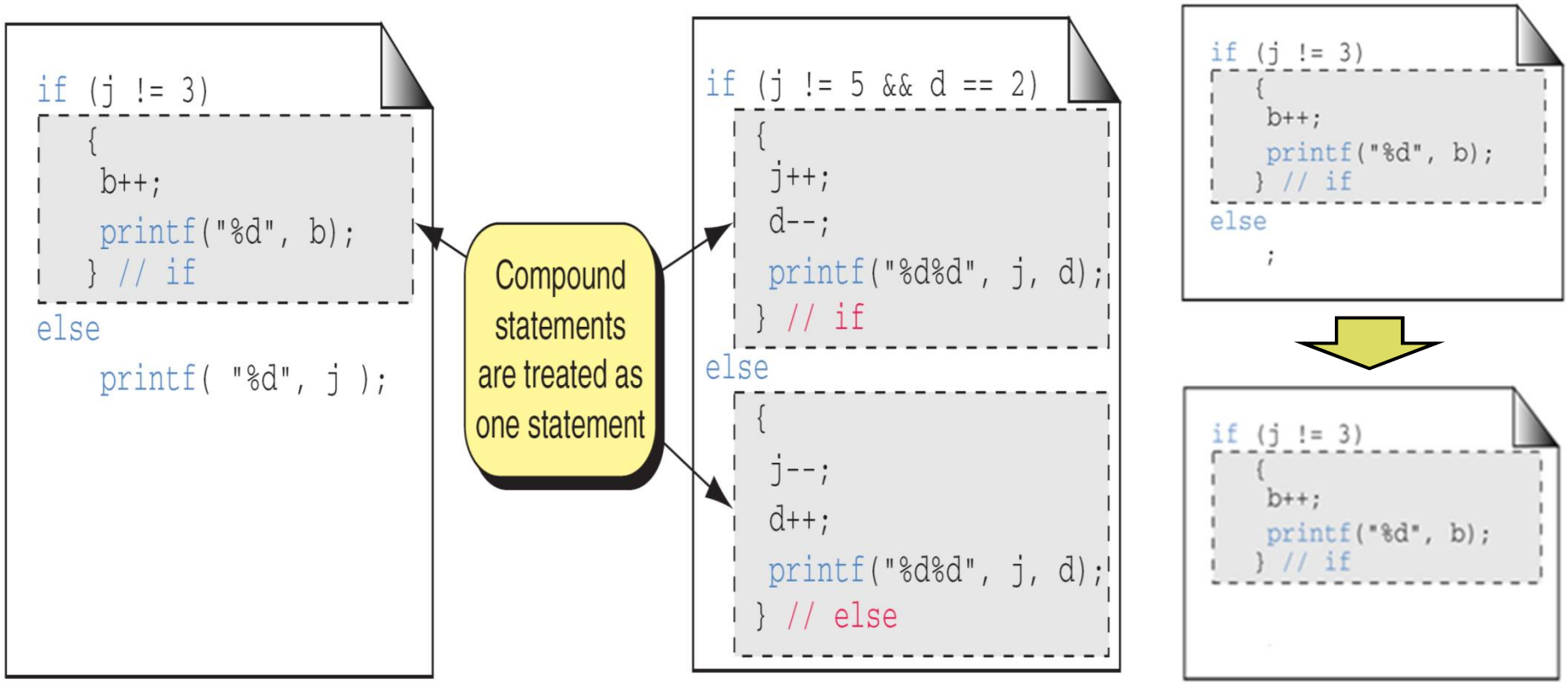
```
if (expression)
    statement1
else
    statement2
```
(b) Code

조건문 if, else

Conditional expression을 검사 후

- 수행하는 statement가 두 줄 이상 일 때는 { }로 묶어줌
- else부분에 들어갈 statement가 없을 경우
  - else 부분 생략 가능 (Null else statement)

조건문 if, else



<Compound statements in an if...else>

<Null else statement>

Nested if

예제 프로그램 – Nested if statement문의 예

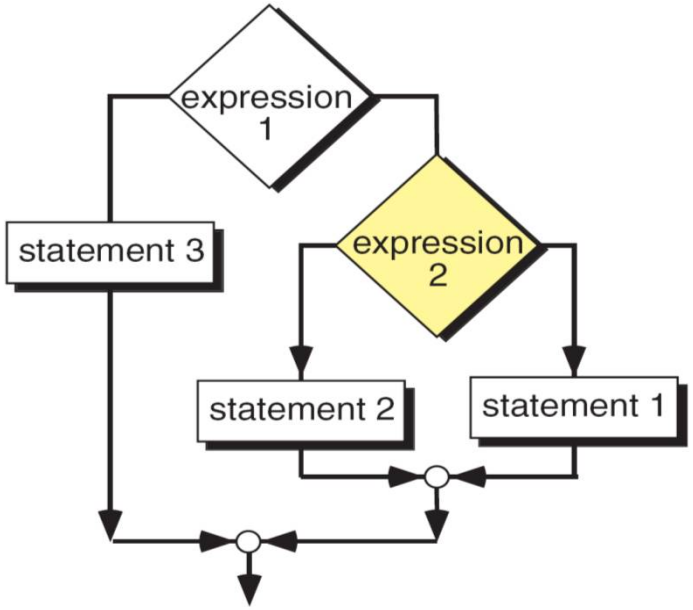
```c
#include <stdio.h>

int main(void){
        int a;
        int b;

        printf("Please enter two integers: ");
        scanf("%d %d",&a, &b);

        if(a <= b)
                if(a < b)
                        printf("%d < %d\n",a,b);
                else
                        printf("%d == %d\n",a,b);
        else
                printf("%d > %d\n",a,b);

        return 0;
}
```

```
gr120100206@cspro:~/CProgramming/chap5$ vi nested_if_statement.c
gr120100206@cspro:~/CProgramming/chap5$ gcc -o nested_if_statement nested_if_statement.c
gr120100206@cspro:~/CProgramming/chap5$ ./nested_if_statement
Please enter two integers: 10 10
10 == 10
gr120100206@cspro:~/CProgramming/chap5$ ./nested_if_statement
Please enter two integers: 10 5
10 > 5
gr120100206@cspro:~/CProgramming/chap5$ ./nested_if_statement
Please enter two integers: 4 7
4 < 7
```

Nested if

Nested if Statements

- if 문 내부에 다른 if문이 중첩(nested)되어 나오는 문장
  - 조건 내에 다른 조건을 넣고 싶을 때 유용한 표현
- 다음은 nested if문의 Logic 흐름과 실제 표기법임



(a) Logic flow

```
if (expression 1)
    if (expression 2)
        statement 1
    else
        statement 2
else
    statement 3
```

(b) Code

Nested if

- if…else 구문에서 여러 개의 if문이 나타난 경우
  else문이 생략되면 의미가 불분명해 질 수 있음
  - 컴퓨터는 if문에 가장 가까운 else를 하나의 쌍으로 처리
  - Dangling else problem 발생
    - [해결책] 의도한 if문을 compound statement { }로 처리

하나의 if..else쌍으로 인식

```
if(alpha == 3)
        if(beta == 4)
                printf("alpha = 3, beta = 4\n");
else
        printf("alpha beta not valid\n");
```

Compound statement { }

```
if(alpha == 3)
{
        if(beta == 4)
                printf("alpha = 3, beta = 4\n");
}
else
        printf("alpha beta not valid\n");
```

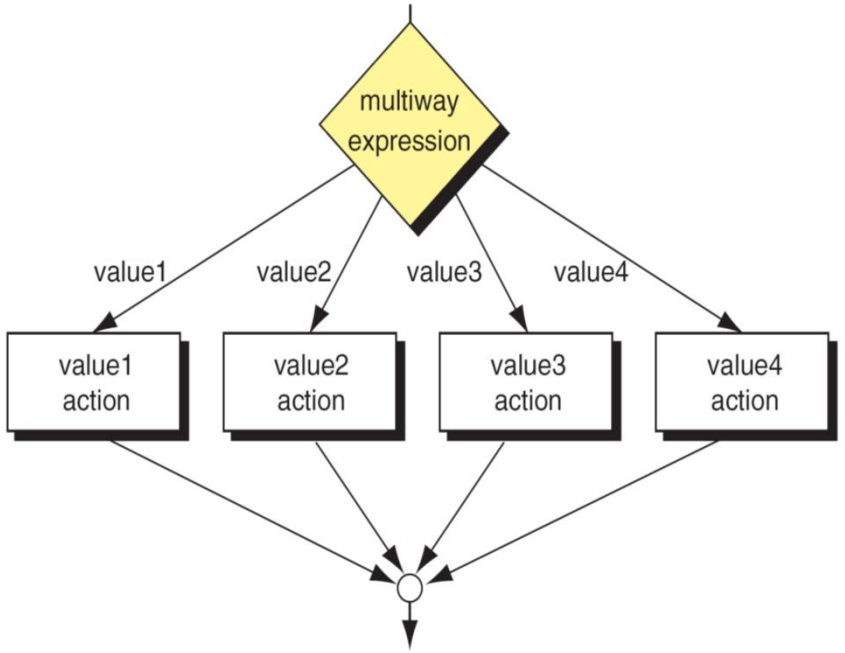
Multiway Selection

Multiway selection에는 크게 switch 문과 else if 문이 존재

Switch문

- **switch 문은 여러 개의 선택을 처리하는 구문**
- **switch 이후의 괄호 ( )사이의 표현 식의 값 중에서 case의 값과 일치하는 것을 처리**
  - **Expression과 value는 반드시 정수나 정수수식이어야 한다.**

switch 문 로직



Switch문

예제 프로그램 – switch문의 예

```c
#include <stdio.h>

char scoreToGrade (int score);

int main(void)
{
        int score;
        char grade;

        printf("Enter the test score (0-100); ");
        scanf("%d", &score);

        grade = scoreToGrade(score);
        printf("The grade is : %c\n", grade);

        return 0;
}
```

```c
char scoreToGrade(int score)
{
        char grade;
        int temp;

        temp = score / 10;
        switch(temp)
        {
                case 10 :
                case 9 : grade='A'; break;
                case 8 : grade='B'; break;
                case 7 : grade='C'; break;
                case 6 : grade='D'; break;
                default : grade='F';
        }
        return grade;
}
```

```
gr120100206@cspro:~/CProgramming/chap5$ vi switch.c
gr120100206@cspro:~/CProgramming/chap5$ gcc -o switch switch.c
gr120100206@cspro:~/CProgramming/chap5$ ./switch
Enter the test score (0-100): 89
The grade is : B
Enter the test score (0-100): 97
The grade is : A
```

Switch문

실행순서

- switch문의 표현식(expression) 계산
- 계산된 값과 일치하는 상수 값(constant)을 갖는 case의 값을 위에서부터 찾음
- 해당 case내부의 문장(statement) 실행
- case 내부에서 break를 만나면 switch문 종료
- 일치된 case문을 만나지 못하여 default를 만나면 default내부의 문장 실행

Default : 반드시 있어야 하지는 않지만 선택되지 않은 모든 경우를 나타낸다.

```
switch (expression)
  {
   case constant-1: statement
                    ⋮
                    statement
   case constant-2: statement
                    ⋮
                    statement
   case constant-n: statement
                    ⋮
                    statement
   default        : statement
                    ⋮
                    statement
  } // end switch
```

Switch문

break문

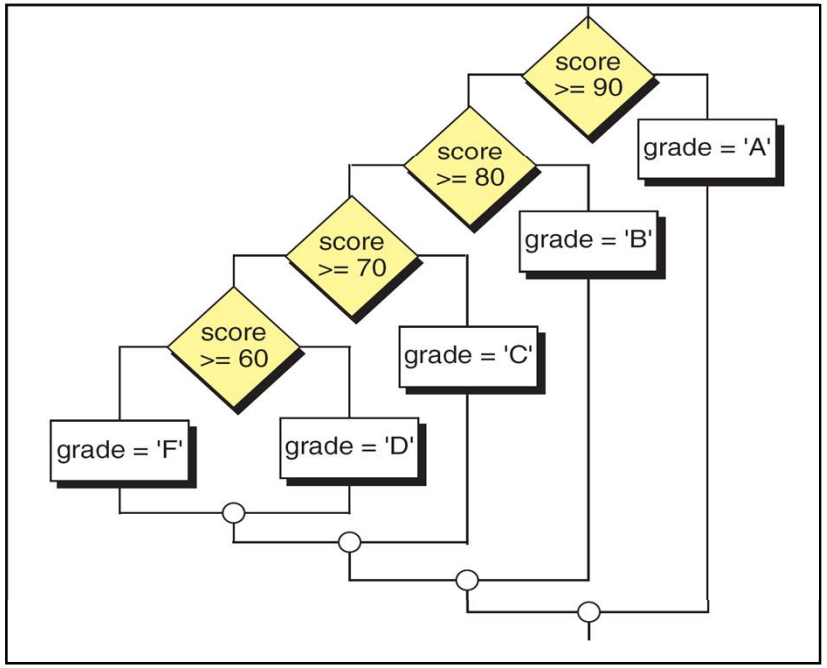
- switch 문에서 break 문을 만나면 무조건 switch 문을 종료
- 그러나 switch 문의 case 문 내부에 break 문이 없다면
  - 일치하는 case 문을 실행하고,
  - 계속해서 break 문을 만나기 전까지 무조건 다음 case 문 내부의 문장을 실행
- break 이후 실행되는 곳은 지금 빠져 나온 structure의 바로 뒷 부분
- break 문을 주로 사용하는 용도
  - Loop으로부터의 탈출
  - switch structure에서 나머지 실행문장을 건너 뛰기 위함

Else if문

예제 프로그램 – else if 문의 예

```
#include <stdio.h>

int main(void) {
        int score=0;
        char grade;

        printf("Enter the test score(0-100); ");
        scanf("%d",&score);

        if(score >= 90)
                grade='A';
        else if(score >= 80)
                grade='B';
        else if(score >= 70)
                grade='C';
        else if(score >=60)
                grade='D';
        else    grade='F';

        printf("The grade is %c\n", grade);
        return 0;
}
```

```
gr120100206@cspro:~/CProgramming/chap5$ vi else_if.c
gr120100206@cspro:~/CProgramming/chap5$ gcc -o else_if else_if.c
gr120100206@cspro:~/CProgramming/chap5$ ./else_if
Enter the test score (0-100): 76
The grade is C
Enter the test score (0-100): 91
The grade is A
```



Else if문

다중판단을 할 경우에 사용 되는 형태

(if…else if…else)

- if의 expression을 검사하여 참일 경우
  - 바로 아래의 statement1 실행
- expression이 거짓일 경우
  - 다음의 else if 로 넘어가서 조건 검사
- 마지막 else는 위의 조건들이 모두 참이 아닐 경우에 statement4 실행
- 맞는 조건을 찾은 다음에는 나머지 조건검사 수행 하지 않음

```
If(expression)
        statement1
else if(expression)
        statement2
else if(expression)
        statement3
else
        statement4
```

Switch vs Else if

switch문과 else if문 모두 조건에 따라 프로그램의 흐름을 분기시키는 목적으로 사용

switch문에는 비교연산이 올 수 없음

switch문으로 구현된 내용은 else if문으로 구현 가능

- 반대의 경우는 구현이 안될 경우가 생김
  - 예 : 비교연산

분기의 횟수가 많을 수록 switch문이 else if문에 비해 간결한 코드 구현이 가능