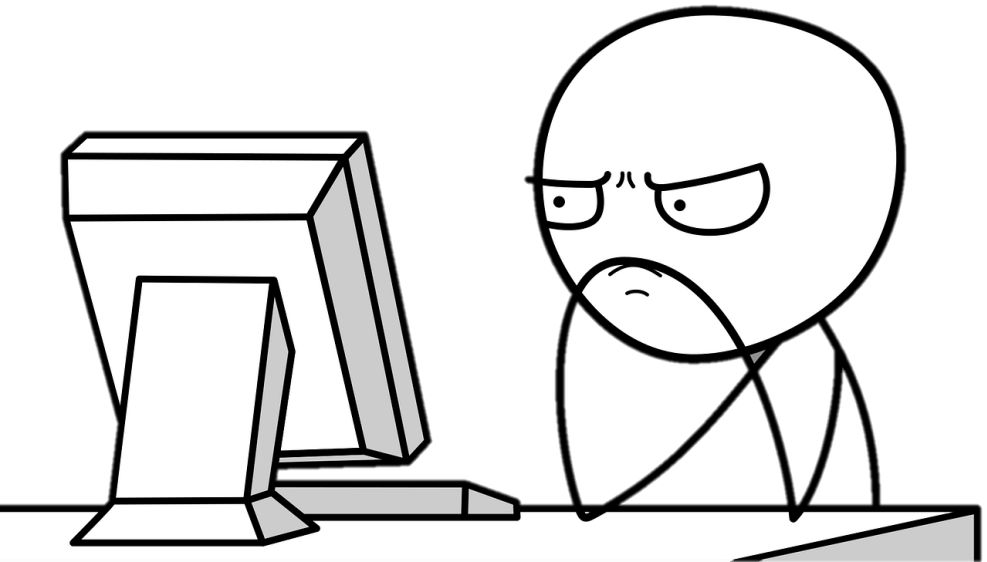

컴퓨터 프로그래밍
Computer Programming


06
조건

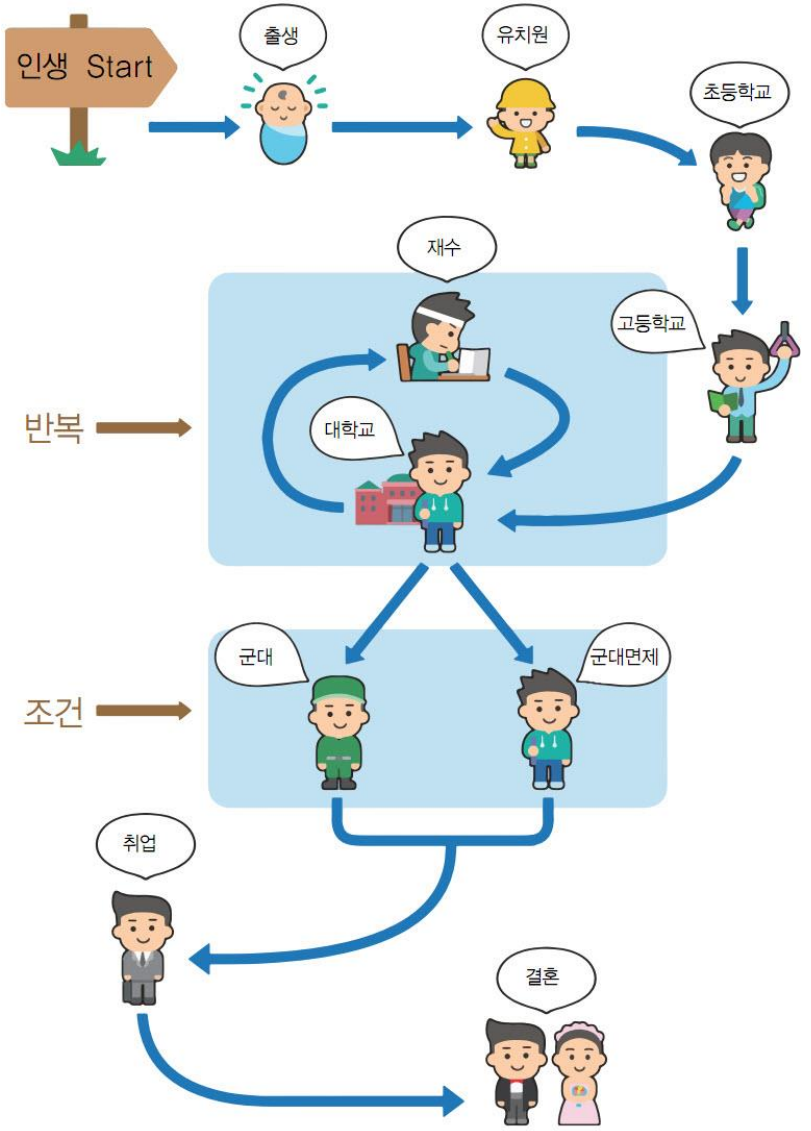
목차

1. 제어문



제어문의 종류

- 순차적 실행
  - 지금까지 배워 온 프로그램의 실행 순서의 원칙: 순차적sequential 실행
  - main 함수 내부에서 배치된 문장이 순차적으로 실행되는 흐름
- 비순차적 실행의 제어문
  - 순차적 실행만으로 프로그램을 모두 작성한다면 매우 비효율적
  - 프로그램의 실행 순서를 제어하는 제어문control statement 제공
  - 선택과 반복 등 순차적인 실행을 변형



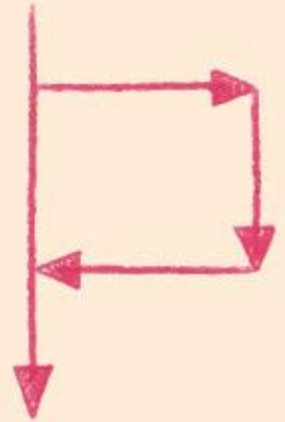
조건선택

- 두 개 또는 여러 개 중에서 한 개를 선택하도록 지원하는 구문

조건선택

조건에 대한 선택 구문

- if
- if else
- if else if
- nested if
- switch



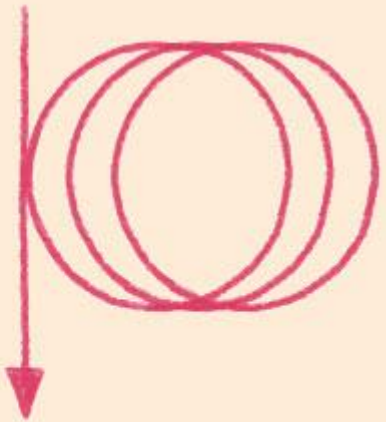
반복(순환)

- 정해진 횟수 또는 조건을 만족하면 정해진 몇 개의 문장을 여러 번 실행하는 구문

반복 순환

반복조건에 따라 일정영역의 반복 구문

- for
- while
- do while



분기처리

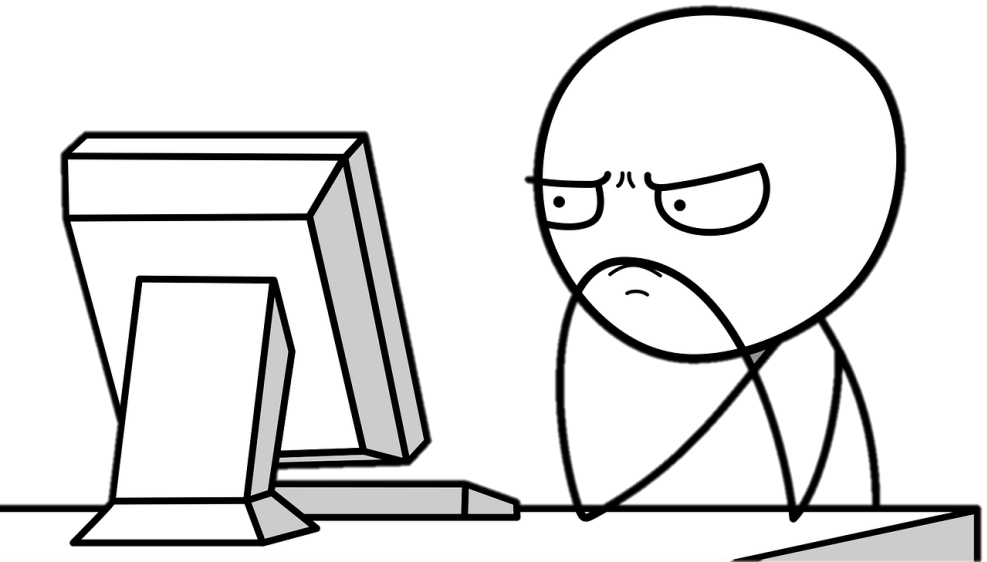
- break 문
  - 작업을 수행 도중 조건에 따라 반복이나 선택을 빠져 나가기
- continue 문
  - 일정구문을 실행하지 않고 다음 반복을 실행
- goto 문
  - 지정된 위치로 이동
- return 문
  - 작업 수행을 마치고 이전 위치로 돌아가는 구문

분기처리

지정된 영역으로 실행을 이동하는 구문

- break
- continue
- goto
- return

2. 조건에 따른 선택 if 문



조건에 따른 결정

- 일상생활에서 조건에 따라 해야 할 내용이 결정되는 사례는 매우 많을 것
- 여러분은 성적에 따라: 조건
  - 장학금을 받을 수도 있고 못 받을 수도 있음

```
평균평점 >= 3.5
```

어느 학교는 평균평점이 3.5는 넘어야 장학금을 받을 수 있다.

```
석차 <= 0.05 * 학생수
```

또 우리 학교는 학과 석차가 상위 5% 이어야 장학금을 받을 수 있다고 한다.

조건에 따라 선택이 발생하는 일상생활에서의 사례

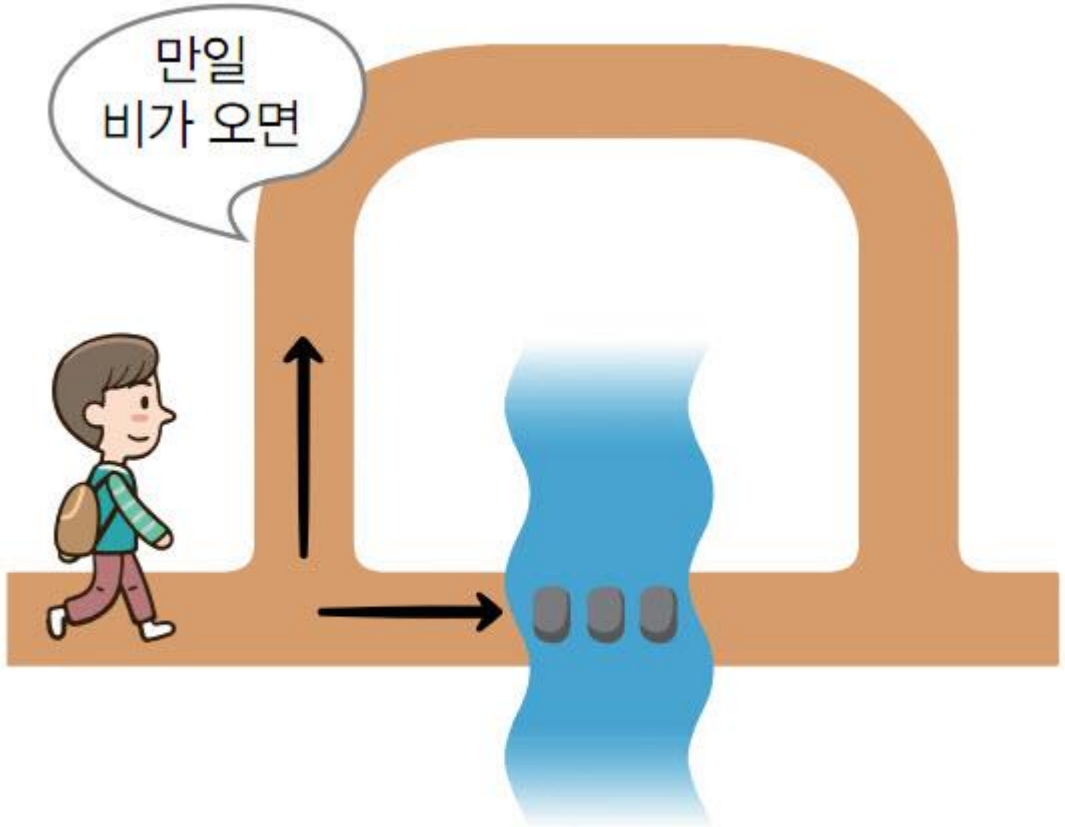
조건 선택의 예	기준변수	조건 표현의 의사코드
온도가 32도 이상이면 폭염 주의를 출력	온도 temperature	만일 (temperature >= 32) printf("폭염 주의");
낮은 혈압이 100이상이면 '고혈압 초기'로 진단	혈압 low_pressure	만일 (low_pressure >= 100) printf("고혈압 초기");
속도가 40km와 60km 사이이면 "적정 속도"라고 출력	속도 speed	만일 (40 <= speed && speed <= 60) printf("적정 속도");

운전면허 필기시험에서 60점 이상이면 합격, 아니면 불합격 출력	시험 성적
	point
		만일 (point >= 60)
		  printf("면허시험 합격");
		아니면
		  printf("면허시험 불합격");

남성이면 체력 테스트에서 80이상이면 합격이고, 아니면 불합격, 여성이면 70이상이면 합격, 아니면 불합격	여성, 남성
	type
	
	체력 점수
	point
		만일 남성이면 (type == 1)
		  만일 (point >= 80)
		  printf("남성: 합격");
		  아니면
		  printf("남성: 불합격");
		아니고 만일 여성이면 (type == 2)
		  만일 (point >= 70)
		  printf("여성: 합격");
		  아니면
		  printf("여성: 불합격");

조건에 따른 선택 if 문장

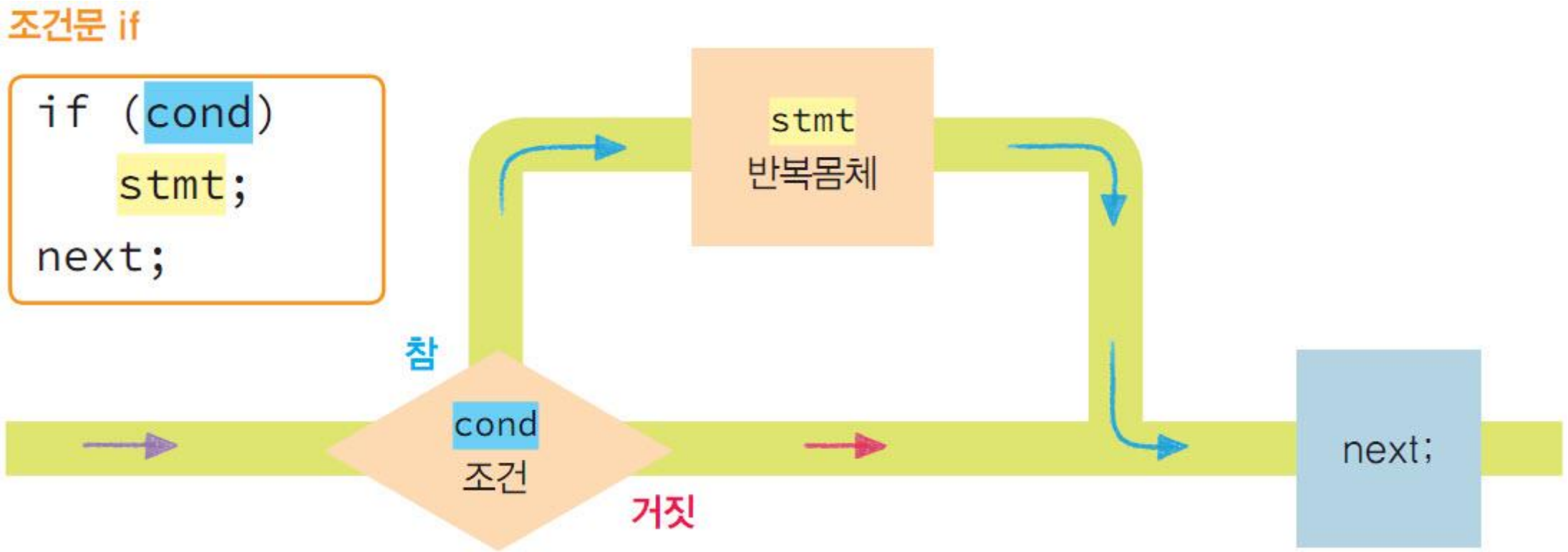
- 조건에 따른 선택을 지원하는 구문
- 일상의 예: 길의 선택
  - 만일 "비가 온다면" 개울이 없는 길 A 선택
  - "비가 오지 않는다면" 개울이 있지만 지름길인 길 B로 갈 것
- 바로 이러한 조건에 따른 문장이 if문



가장 간단한 if문의 형태

- if (cond) stmt;
  - 조건식 cond가 0이 아니면(참)
    - stmt를 실행
  - 0이면(거짓)
    - stmt를 실행하지 않음
  - 문장 stmt는 여러 문장이라면 블록으로 구성
  - if 문이 종료되면
    - 그 다음 문장 실행
  - 조건식 (cond)는 반드시 괄호 필요
  - 문장 stmt은 반드시 들여쓰기

if 문과 두개의 길



if 문

블록이면 다음과 같이 블록 시작 {을 if열에 맞추고 조건 문장들을 들여쓰기 마지막에 행에 다시 블록 종료 }를 시작 열에 맞춘다.

조건문 if

```
if (cond)
    stmt;
next;
```

cond가 만족되면 실행되는 문장

```
if (grade >= 3.2)
{
    printf("회사에 지원할 수 있습니다.\n");
}
printf("졸업을 축하합니다.\n");
```

Source Code #01: if.c

```c
// file: if.c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    double temperature;

    printf("현재 온도 입력: ");
    scanf("%lf", &temperature);

    if (temperature >= 32.0)
    {
        printf("폭염 주의보를 발령합니다.\n");
        printf("건강에 유의하세요.\n");
    }
    printf("현재 온도는 섭씨 %.2f 입니다.\n", temperature);

    return 0;
}
```

```
현재 온도 입력: 32.1
폭염 주의보를 발령합니다.
건강에 유의하세요.
현재 온도는 섭씨 32.10 입니다.
```

조건 if문 주의사항

- if (cond); stmt;와 같이 조건식 다음에 세미콜론을 쓰는 경우
  - 문법오류가 없으며 if 문의 조건으로 인해 실행되는 문장은 하나도 없고, 이후 stmt는 무조건 실행
  - 즉 if 조건을 만족하지 않아도 stmt;가 항상 실행
  - 실제 코딩과정에서 문법오류가 발생하지 않아 논리적인 문제를 찾기 쉽지 않음
- if cond stmt;와 같이 조건식에 괄호를 빼먹는 경우
  - 문법오류가 표시되어 바로 수정할 수 있음

학점이 3.2 미만이라도 다음 두 문장은 항상 실행되는 결과를 낳는다.

```
if (grade >= 3.2);
   printf("회사에 지원할 수 있습니다.\n");

printf("졸업을 축하합니다.\n");
```

if 문에서 조건식 grade >= 3.2에는 괄호가 필요하다. 문법오류로 인한 오류풍선이 발생하여 쉽게 알 수 있다.

```
if  grade >= 3.2
   printf("회사에 지원할 수 있습니다.\n");

printf("졸업을 축하합니다.\n");
```

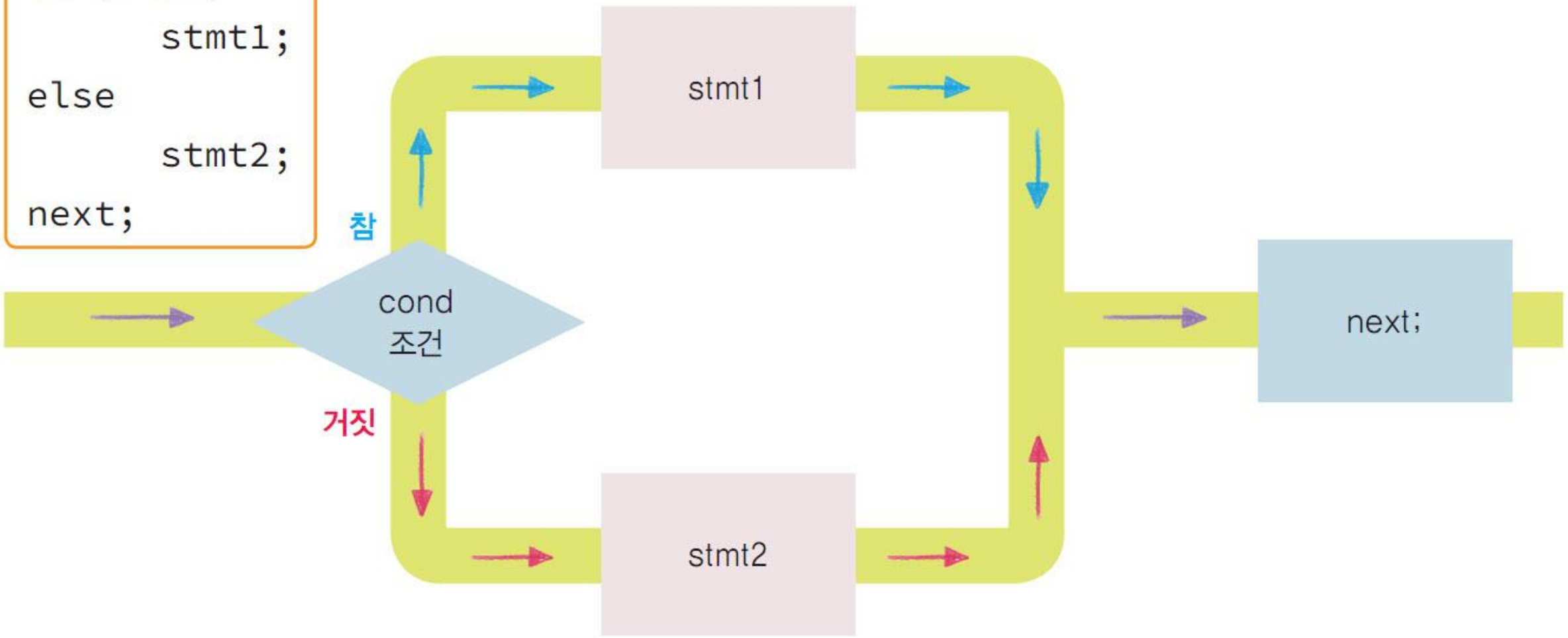
if else 문장

- 조건 만족 여부에 대한 선택 if else
  - 조건문 if (cond) stmt1; else stmt2;
  - 조건 cond를 만족하면 stmt1을 실행
  - 조건 cond를 만족하지 않으면 stmt2를 실행
  - stmt1과 stmt2 둘 중의 하나를 선택하는 구문
- 조건식: 나머지 연산자 %의 사용
  - 조건식 (n % 2): 0이 아니면(참) 홀수이고 0이면(거짓) 짝수
- 조건문 if else에서 주의해야 할 점
  - (조건식)은 괄호가 필요
  - 조건식에서 등호를 대입으로 잘못 쓰는 것에 주의
  - 즉 (n == 100)을 (n = 100)로 잘못 쓰면 항상 참으로 인식
  - stmt2 부분이 여러 문장이면 {여러 문장들}의 블록으로 구성

if else문의 제어흐름

조건문 if else

```
if (cond)
        stmt1;
else
        stmt2;
next;
```



조건문 if else

조건문 if else

```
if (cond)
        stmt1;
else
        stmt2;
next;
```

```
if (n % 2 == 0)
    printf("짝수");
else
    printf("홀수");

printf("입니다.\n");
```

```
if (n % 2)
    printf("홀수");
else
    printf("짝수");

printf("입니다.\n");
```

if else 문장

- 조건식에서 (n != 0)와 (n)은 같은 식
- 마찬가지로 (n == 0)와 (!n)도 같은 식

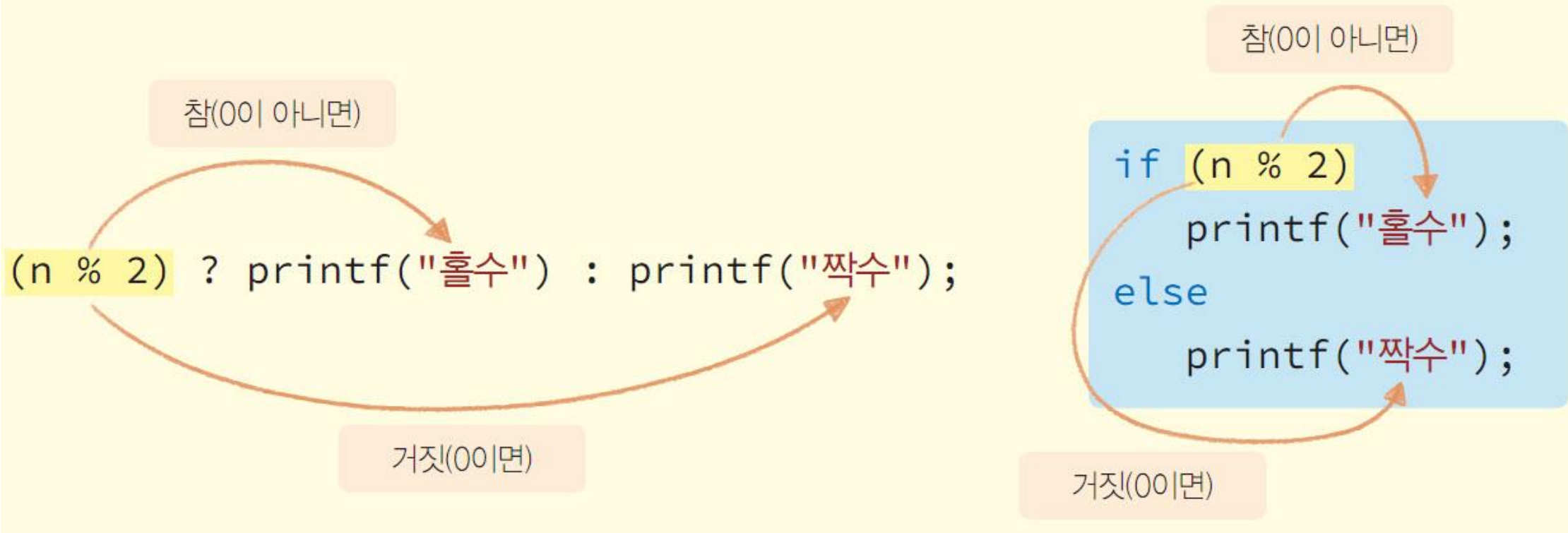
조건식	설명	예	

(n != 0)
 	
(n)
	n이 0이 아니어야 참인 연산식이므로 연산식 (n)과 같음	`if (n % 2 != 0)`
		`  printf("홀수");`
		`else`
		`  printf("짝수");`
			`if (n % 2)`
			`  printf("홀수");`
			`else`
			`  printf("짝수");`

(n == 0)
 	
(!n)
	n이 0이어야 참인 연산식이므로 연산식 (!n)과 같음	`if (n % 2 == 0)`
		`  printf("짝수");`
		`else`
		`  printf("홀수");`
			`if (!(n % 2))`
			`  printf("짝수");`
			`else`
			`  printf("홀수");`

if와 조건연산자

- 조건 문장 if는 이미 배운 조건연산자와 매우 유사해서 좀 더 편리하게 활용



조건문 if else 문

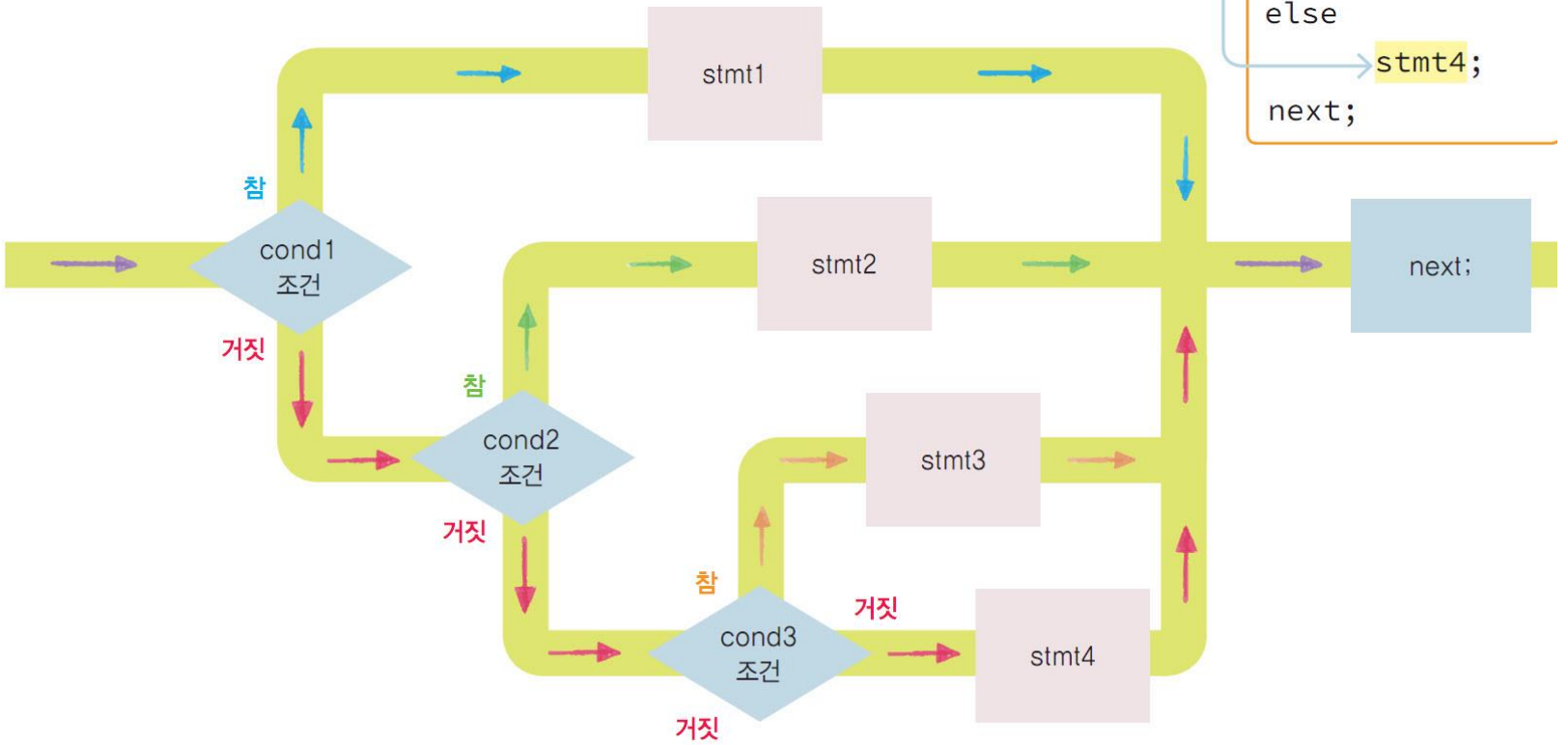
- else 이후에 if else를 필요한 횟수만큼 반복 가능
- 다음 문장에서 stmt1에서 stmt4에 이르는 여러 문장 중에서 실행되는 문장은 단 하나라는 것을 기억

조건문 if else if 문의 제어흐름

조건문 if else if

```
if (cond1)
    stmt1;
else if (cond2)
    stmt2;
else if (cond3)
    stmt3;
else
    stmt4;
next;
```

여러 개의 문장중 단 하나만 실행된다.



조건문 반복된 if else의 예

- 점수가 90이상이면 학점이 A
  - 80 이상에서 90 미만이면 B, 70 이상에서 80 미만이면 C
  - 60 이상에서 70 미만이면 D, 마지막으로 60 미만이면 F로 학점을 처리하는 모듈
- 가장 처음의 if는 if (point>=90)으로 시작
- 그 다음 if는 else 이후의 if이므로 else if (point>=80)으로 표현
  - 80 이상에서 90 미만인 조건(90>point && point>=80)이 만족

조건문 반복된 if else if else를 사용한 성적 처리

이 조건식은 첫 if의 조건식인 (point >= 90)이 만족되지 않고 체크되는 것이므로 결국 (!(point >= 90) && (point >= 80))이므로 80 이상에서 90 미만인 조건 (90>point && point>=80)이 만족된다.

```
if (point >= 90)
   printf("A\n");
else if (point >= 80)
   printf("B\n");
else if (point >= 70)
   printf("C\n");
else if (point >= 60)
   printf("D\n");
else
   printf("F\n");
```

필요하면 이와 같이 블록 사용이 가능하다.

```
if (point >= 90)
{
   printf("A\n");
}
else if (point >= 80)
{
   printf("B\n");
}
else if (point >= 70)
{
   printf("C\n");
}
else if (point >= 60)
{
   printf("D\n");
}
else
{
   printf("F\n");
}
```

Source Code #02: ifelseif.c

```c
// file: ifelseif.c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    double gpa;

    printf("평균평점 입력: ");
    scanf("%lf", &gpa);

    if (gpa >= 4.3)
        printf("성적이 최고 우수한 학생입니다.\n");
    else if (gpa >= 3.8)
        printf("성적이 매우 우수한 학생입니다.\n");
    else if (gpa >= 3.0)
        printf("성적이 우수한 학생입니다.\n");
    else
        printf("성적이 3.0 미만인 학생입니다.\n");

    return 0;
}
```

```
평균평점 입력: 4.3
성적이 최고 우수한 학생입니다.
```

```
평균평점 입력: 3.3
성적이 우수한 학생입니다.
```

```
평균평점 입력: 3.9
성적이 매우 우수한 학생입니다.
```

```
평균평점 입력: 2.7
성적이 3.0 미만인 학생입니다.
```

Lab #01: 두 실수의 대소에 따라 다양한 연산을 수행하여 그 결과를 출력

- 두 실수를 입력 받아 두 실수의 연산 값이 출력되는 프로그램
  - 만일 x > y 이면 x / y 연산값 출력
  - 만일 x < y 이면 x + y 연산값 출력
  - 만일 x == y 이면 x * y 연산값 출력
- 결과
  - 두 실수를 입력: 32.765 3.987
  - 연산 결과: 8.22

```c
// file: tworeal.c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    double x = 0, y = 0, result = 0;

    printf("두 실수를 입력: ");
    scanf("%lf %lf", &x, &y);

    
    {
        result = x / y;
    }
    else if (x == y)
    {
        result = x * y;
    }
    
    {
        result = x + y;
    }

    printf("연산 결과: %.2f\n", result);

    return 0;
}
```

Source Code #03: nestedif.c

```c
// file: nestedif.c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    int type, point;

    printf("번호를 선택: 1(1종면허), 2(2종면허): ");
    scanf("%d", &type);
    printf("필기시험 점수 입력: ");
    scanf("%d", &point);

    if (type == 1)
    {
        if (point >= 70)
            printf("1종면허 합격\n");
        else
            printf("1종면허 불합격\n");
    }
    else if (type == 2)
    {
        if (point >= 60)
            printf("2종면허 합격\n");
        else
            printf("2종면허 불합격\n");
    }

    return 0;
}
```

```
번호를 선택: 1(1종면허), 2(2종면허): 1
필기시험 점수 입력: 67
1종면허 불합격
```

```
번호를 선택: 1(1종면허), 2(2종면허): 1
필기시험 점수 입력: 77
1종면허 합격
```

```
번호를 선택: 1(1종면허), 2(2종면허): 2
필기시험 점수 입력: 58
2종면허 불합격
```

```
번호를 선택: 1(1종면허), 2(2종면허): 2
필기시험 점수 입력: 63
2종면허 합격
```

중첩된 if 문

```
if ( type == 1 )
{
    문장1;
}
else if ( type == 2 )
{
    문장2;
}
```

문장 1

```
if ( point >= 70 )
   printf("1종면허 합격\n");
else
   printf("1종면허 불합격\n");
```

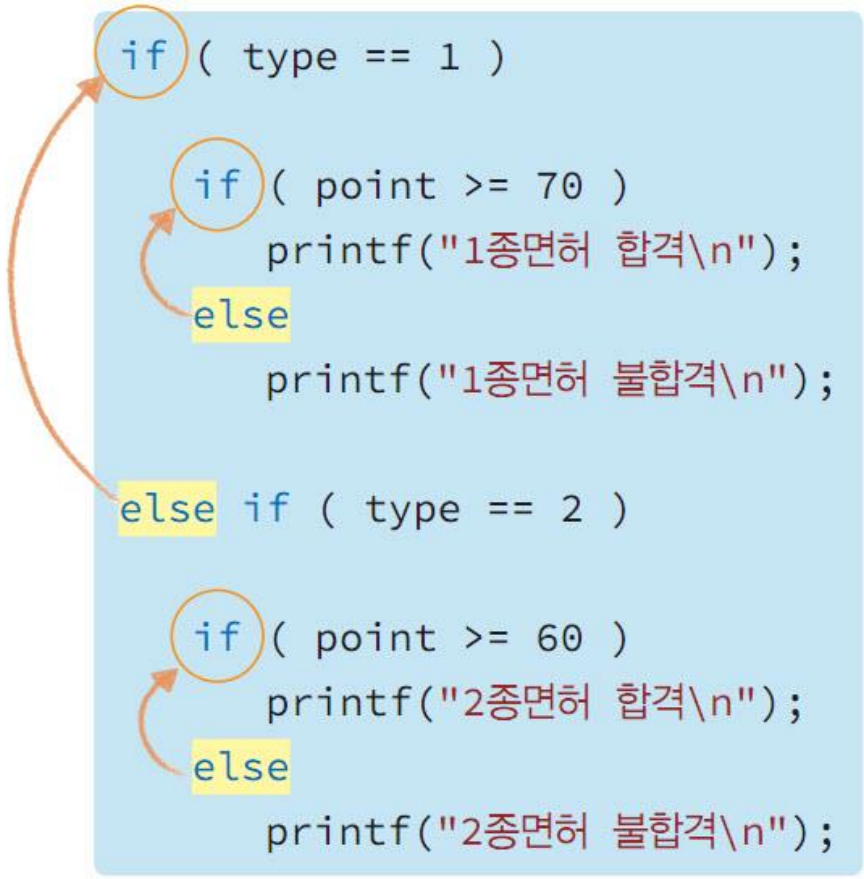
문장 2

```
if ( point >= 60 )
   printf("2종면허 합격\n");
else
   printf("2종면허 불합격\n");
```

블록 표시와 else

- 블록이 없는 경우 else 문장이 어느 if문의 소속인지 정확히 판단
  - else는 문법적으로 같은 블록 내에서 else가 없는 가장 근접한 상위의 if문에 소속된 else로 해석
  - else의 혼란을 방지하려면 블록을 이용하는 것을 권고

```
if ( type == 1 )
{
    if ( point >= 70 )
        printf("1종면허 합격\n");
    else
        printf("1종면허 불합격\n");
}
else if ( type == 2 )
{
    if ( point >= 60 )
        printf("2종면허 합격\n");
    else
        printf("2종면허 불합격\n");
}
```



else 구문의 주의

```
int age = 30;

if (age >= 20)
    if (age >= 65)
        printf("당신은 어르신입니다.\n");
else
    printf("당신은 미성년자입니다.\n");
```

```
int age = 30;

if (age >= 20)
{
    if (age >= 65)
        printf("당신은 어르신입니다.\n");
    else
        printf("당신은 미성년자입니다.\n");
}
```


```
int age = 30;

if (age >= 20)
{
    if (age >= 65)
        printf("당신은 어르신입니다.\n");
}
else
    printf("당신은 미성년자입니다.\n");
}
```

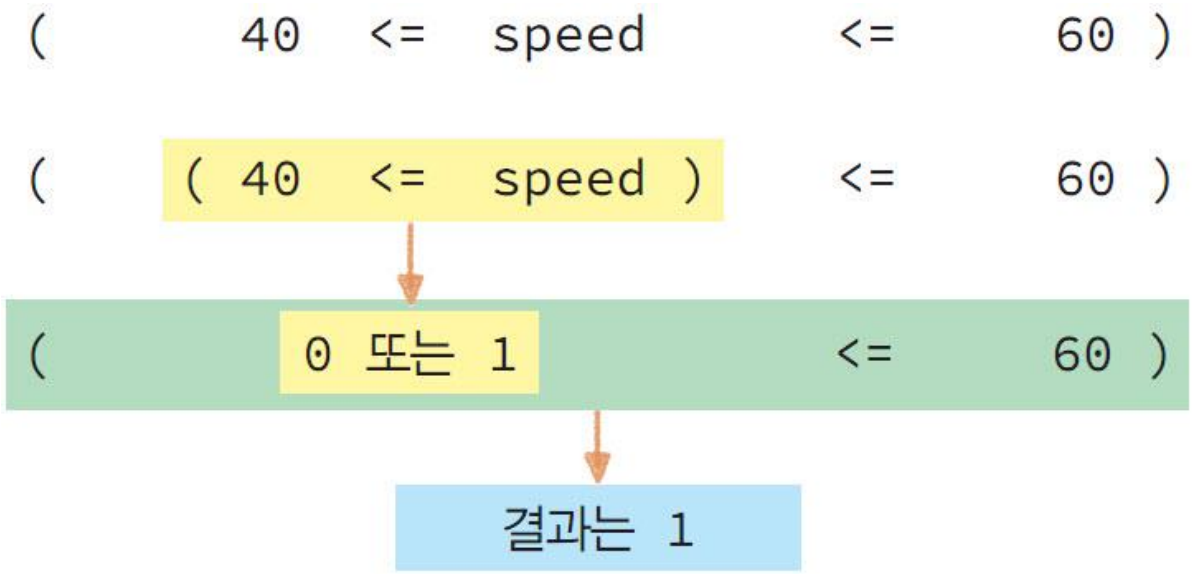
조건에 따라 선택이 발생하는 사례와 구현

조건 표현	if 형태	기준변수	다양한 if 문으로 구성
온도가 32도 이상이면 폭염주의를 출력	if	온도 temperature	`if (temperature >= 32)` `  printf("폭염주의");`
속도가 40km와 60km 사이이면 "적정속도"라고 출력	if	속도 speed	`if (40 <= speed && speed <= 60)` `  printf("적정속도");`

운전면허 필기시험에서 60점 이상이면 합격, 아니면 불합격 출력	if else	시험성적
		point
			`if (point >= 60)`
			`  printf("면허시험 합격");`
			`else`
			`  printf("면허시험 불합격");`

주의해야 할 조건 연산식

- 속도 speed가 40km와 60km 사이라는 조건식
  - (40 <= speed && speed <= 60)
  - (40 <= speed <= 60)로 사용한다면 잘못된 조건식



if와 조건연산자

구현 내용	조건연산자	if

두 수의 최대값 구하기	`max = x > y ? x : y;`	`if (x > y)`
		`  max = x;`
		`else`
		`  max = y;`

두 수의 최소값 구하기	`min = x > y ? y : x;`	`if (x > y)`
		`  min = y;`
		`else`
		`  min = x;`

절대값 구하기	`abs = x >= 0 ? x : -x;`	`if (x >= 0)`
		`  abs = x;`
		`else`
		`  abs = -x;`

홀수와 짝수 구하기	`a % 2 ? printf("홀수") : printf("짝수");`	`if (a % 2)`
		`  printf("홀수");`
		`else`
		`  printf("짝수");`

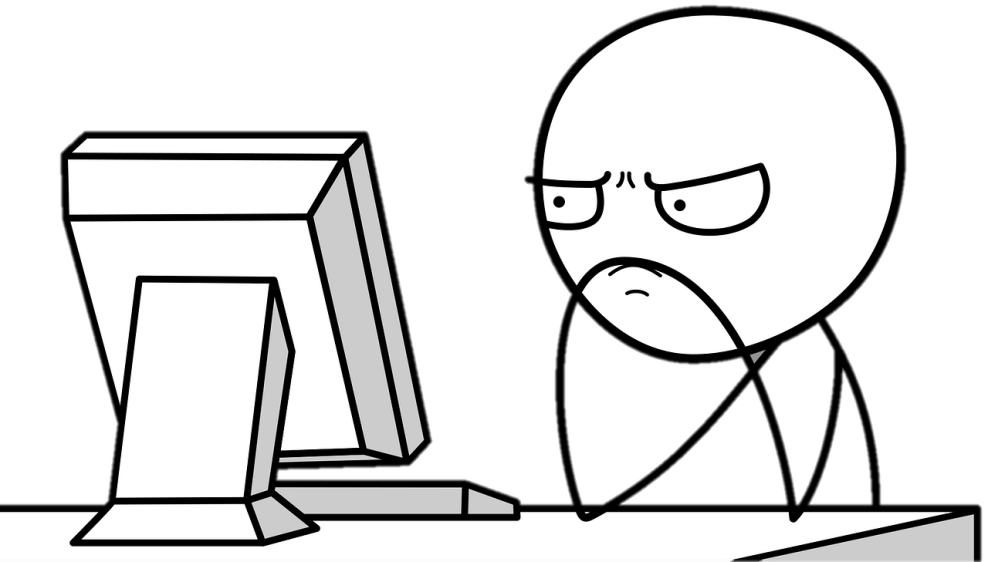
Lab #02: 표준입력으로 받은 세 정수의 최대값을 출력

- 표준입력으로 받은 세 정수에서 최대값이 출력되는 프로그램
  - 먼저 조건식 x > y 이 참이면 x와 y의 최대값, 거짓이면 x와 y의 최소값

- 결과
  - 세 정수를 입력: 10 30 20
  - 최대 수: 30

```c
// file: maxof3.c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    int x, y, z;

    printf("세 정수를 입력: ");
    scanf("%d %d %d", &x, &y, &z);

    if (x > y)
    {
        if (          )
            printf("최대 수: %d\n", x);
        else
            printf("최대 수: %d\n", z);
    }
    else
    {
        if (          )
            printf("최대 수: %d\n", y);
        else
            printf("최대 수: %d\n", z);
    }

    return 0;
}
```

3. 다양한 선택 switch 문

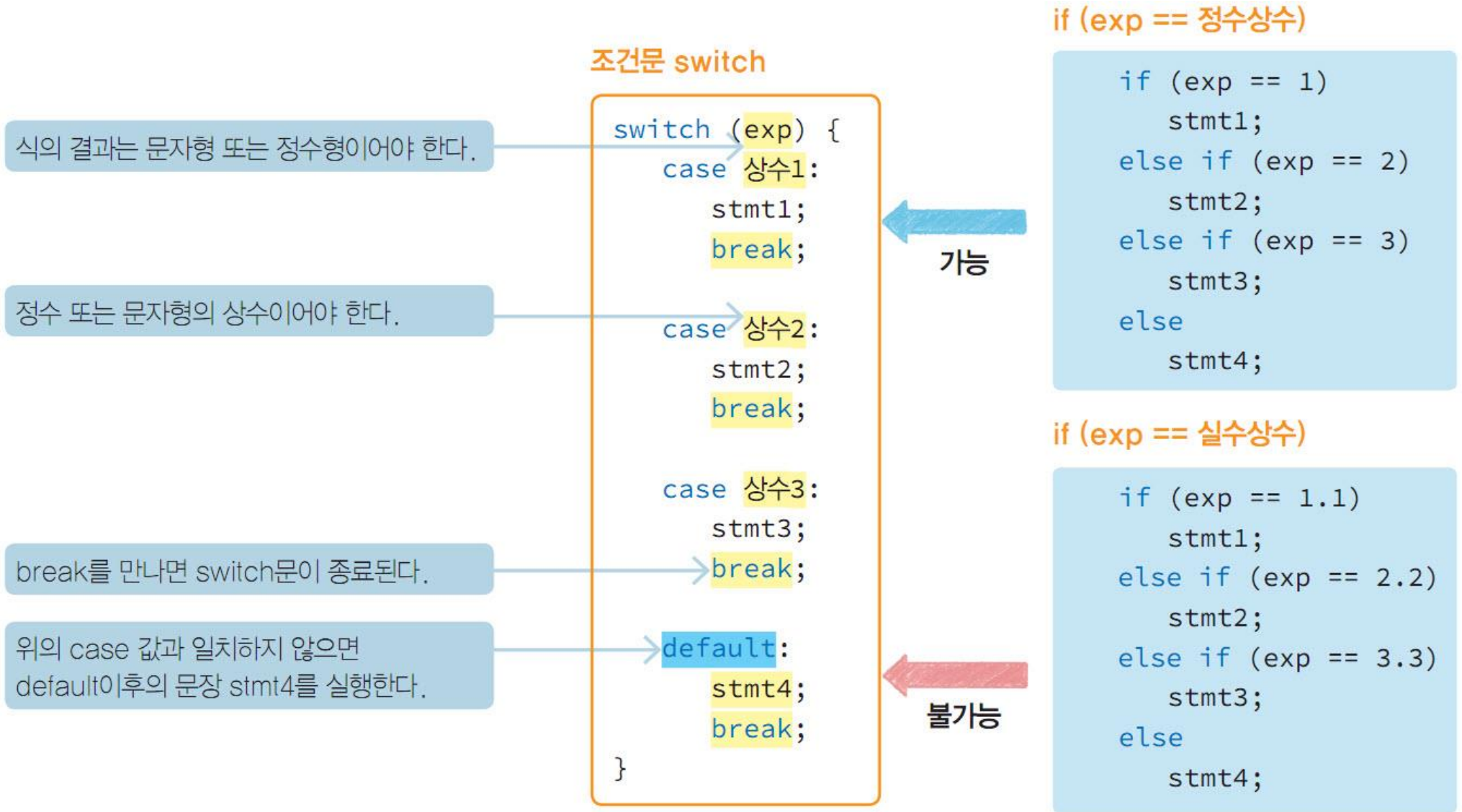


switch 문장 개요

- 연산식의 정수 또는 문자 선택
  - 다중선택 구문인 switch문을 사용
  - 문장 if else가 여러 번 계속 반복되는 구문을 좀 더 간략하게 구현
  - 특히 if의 조건식이 정수와 등호식이라면 보다 간편한 switch문의 사용이 가능

- 주어진 연산식이 문자형 또는 정수형
  - 그 값에 따라 case의 상수값과 일치하는 부분의 문장들을 수행하는 선택 구문

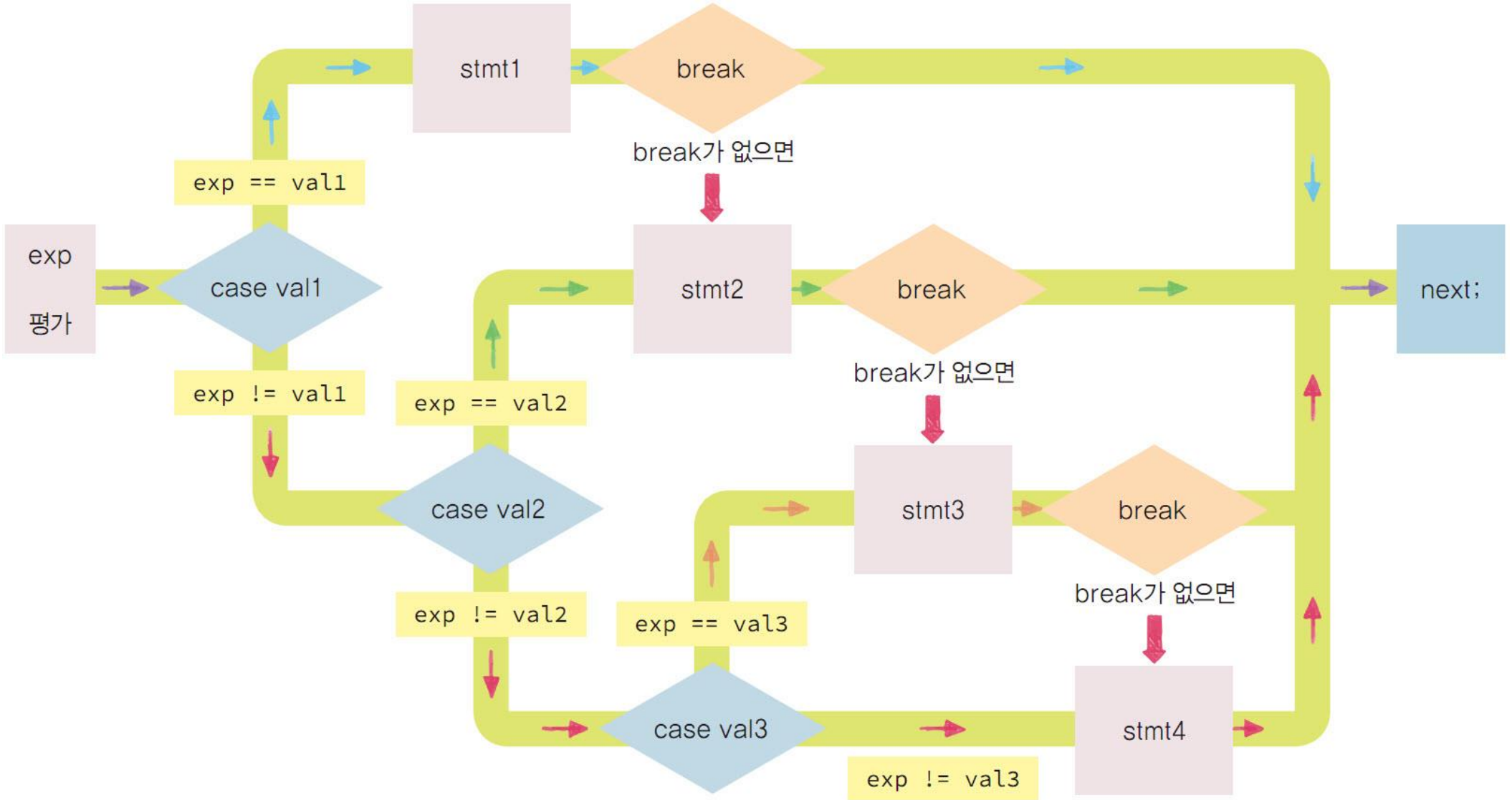
Switch문 문장구조



switch 문

- switch (exp) { …} 문: switch, case, break, default는 키워드
  - 표현식 exp 결과값: 반드시 문자 또는 정수
    - case의 값과 일치하는 항목의 문장 stmt1을 실행한 후 break를 만나 종료
    - case 다음의 value 값은 변수가 올 수 없으며, 상수식(constant expression)으로 그 결과가 정수 또는 문자 상수 (값은 중복 불가능)
    - break 문이 없으면 break 문을 만나기 전까지 다음 case 의 내부로 무조건 이동하여 내부 문장을 실행
- default는 선택적
  - 일치된 case 값을 만나지 못하여 default를 만나면 default 내부의 문장을 실행
  - default의 위치
    - 어디에도 위치 가능
    - 중간에 위치하면서 break문이 없으면 하부 case 내부 분장을 무조건 실행하므로 주의가 필요

Switch문의 제어흐름



Source Code #04: switch.c

```c
// file: switch.c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    double x, y;
    int op;

    printf("두 실수 입력: ");
    scanf("%lf %lf", &x, &y);
    printf("연산종류 번호선택 1(+), 2(-), 3(*), 4(/): ");
    scanf("%d", &op);

    switch (op) {
    case 1:
        printf("%.2f + %.2f = %.2f\n", x, y, x + y);
        break;
    case 2:
        printf("%.2f - %.2f = %.2f\n", x, y, x - y);
        break;
    case 3:
        printf("%.2f * %.2f = %.2f\n", x, y, x * y);
        break;
    case 4:
        printf("%.2f / %.2f = %.2f\n", x, y, x / y);
        break;

    default:
        printf("번호를 잘못 선택했습니다.\n");
        break; //생략가능
    }

    return 0;
}
```

```
두 실수 입력: 3.765 6.987
연산종류 번호선택 1(+), 2(-), 3(*), 4(/): 1
3.77 + 6.99 = 10.75
```

```
두 실수 입력: 4.82 3.987
연산종류 번호선택 1(+), 2(-), 3(*), 4(/): 2
4.82 - 3.99 = 0.83
```

```
두 실수 입력: 3.986 4.826
연산종류 번호선택 1(+), 2(-), 3(*), 4(/): 3
3.99 * 4.83 = 19.24
```

```
두 실수 입력: 87.354 6.98
연산종류 번호선택 1(+), 2(-), 3(*), 4(/): 4
87.35 / 6.98 = 12.51
```

Source Code #05: seasonswitch.c

```c
// file: seasonswitch.c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    int  month;

    printf("년도의 월(month)을 입력: ");
    scanf("%d", &month);

    switch (month) {
    case 4: case 5:
        printf("%d월은 봄입니다.\n", month);
        break;

    case 6: case 7: case 8:
        printf("%d월은 여름입니다.\n", month);
        break;

    case 9: case 10: case 11:
        printf("%d월은 가을입니다.\n", month);
        break;

    case 12: case 1: case 2: case 3:
        printf("%d월은 겨울입니다.\n", month);
        break;

    default:
        printf("월(month)을 잘못 입력하셨습니다.\n");
    }

    return 0;
}
```

- 표준입력으로 1년 중 해당하는 월을 입력받아 그 달에 맞는 계절을 출력하는 프로그램

OR 개념 처리

- case 문 내부에 break 문이 없다면
  - 일치하는 case 문을 실행하고
  - break 문을 만나기 전까지 다음 case 내부 문장을 실행
- case 4, 5와 같은 나열은 문법오류가 발생

여러 개의 정수 선택에 따른 case 처리 방법

```
switch ( month )
{
    case 4 : case 5 :
        printf("%d월은 봄입니다.\n", month);
        break;

    case 6 : case 7 : case 8 :
        printf("%d월은 여름입니다.\n", month);
        break;

    ...

    default :
        printf("월(month)을 잘못 입력하셨습니다.\n");
}
```

```
case 4, 5 : //오류발생
    ...
    break;

case 6, 7, 8 : //오류발생
    ...
    break;
```

Lab #03: 표준입력으로 받은 세 정수의 최대값을 출력

- 표준입력으로 받은 세 정수에서 최대값이 출력되는 프로그램
  - 먼저 조건식 x > y 의 결과를 switch 문을 이용
  - 조건연산자를 이용하여 두 수 중에서 최대값을 출력
- 결과
  - 세 정수를 입력: 5 10 8
  - 최대값: 10

```c
// file: simplemaxof3.c
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

int main(void)
{
    int x, y, z;

    printf("세 정수를 입력: ");
    scanf("%d %d %d", &x, &y, &z);

    switch ((x > y))
    {
    case 0:
        printf("최대 값: %d\n", [redacted]);
        break;

    case 1:
        printf("최대 값: %d\n", [redacted]);
        break;
    }

    return 0;
}
```

Source Code #06: scoreswitch.c

```c
// file: scoreswitch.c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    int  score;

    printf("점수 입력: ");
    scanf("%d", &score);

    switch (score / 10) {
    case 10: case 9:
        printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'A');
        break;
    case 8:
        printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'B');
        break;
    case 7:
        printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'C');
        break;
    case 6:
        printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'D');
        break;

    default:
        printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'F');
    }

    return 0;
}
```

- 표준 입력된 성적에 따라 성적 'A'에서 'F'까지 부여
- 점수를 10으로 나눈 연산식 (score / 10)을 활용
- switch 문으로 성적처리 가능

```
점수 입력: 100
점수가 100 점으로 성적이 A 입니다.
```

```
점수 입력: 88
점수가 88 점으로 성적이 B 입니다.
```

```
점수 입력: 75
점수가 75 점으로 성적이 C 입니다.
```

```
점수 입력: 62
점수가 62 점으로 성적이 D 입니다.
```

점수에 따른 성적처리를 위한 연산값

점수 예	점수 범위	(score / 10) 연산값	성적처리
100, 98, 95, 90	90 <= 점수 <= 100	9 또는 10	'A' 부여
80, 85, 88, 89	80 <= 점수 < 90	8	'B' 부여
80, 85, 88, 89	70 <= 점수 < 80	7	'C' 부여
80, 85, 88, 89	60 <= 점수 < 70	6	'D' 부여
30, 55, 58, 59	점수 < 60	그 외	'F' 부여

default의 위치

- 일반적으로 switch 문에서 default는 생략 가능
  - 그 위치도 제한이 없음
  - default를 위치시킨 이후에 다른 case가 있다면 break를 반드시 입력
  - 만일 break문이 제거되면 case 상수 외에도 다시 상수 10과 9에 기술된 내부 문장이 실행되는 논리 오류 발생

default 위치에 따른 break 추가

```
switch (score / 10) {
default:
    printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'F');
    break; //반드시 필요한 break

case 10: case 9:
    printf("점수가 %d 점으로 성적이 %c 입니다.\n", score, 'A');
    break;
    ...
}
```

Lab #04: 열거 상수로 switch 문 활용

- 표준입력으로 받은 정수에 대응하는 열거 상수로 switch문에서 분기를 처리하는 프로그램
  - 삼원색을 표현하는 열거상수로 RED, GREEN, BLUE를 정의
  - 세 정수(R[0], G[1], B[2]) 중의 하나를 입력
  - switch의 case 상수로 열거 상수를 이용
- 결과
  - 세 정수(R[0], G[1], B[2]) 중의 하나를 입력: 0
  - Red

```c
// enumswitch.c
#define _CRT_SECURE_NO_WARNINGS //scanf() 오류를 방지하기 위한 상수 정의

#include <stdio.h>

int main(void)
{
    enum color { RED, GREEN, BLUE };
    int input;

    printf("세 정수(R[0], G[1], B[2]) 중의 하나를 입력: ");
    scanf("%d", &input);

    switch (input) {
    case RED:
        printf("Red\n");
        break;

    case GREEN:
        printf("Green\n");
        break;

    case BLUE:
        printf("Blue\n");
        break;

    default:
        printf("잘못된 입력\n");
    }

    return 0;
}
```


I
PROGRAMMING