

컴퓨터 프로그래밍

Flow control

Flow

- Statement
 - program 실행 단위
 - semi-colon(;)으로 구분
- Block
 - set of statements
 - { }로 지정
 - block 내에 선언된 변수는 block 내에서만 유효
- Method 내에서는
 - 코드에 작성된 문장이 순차적으로 실행
 - control statement를 통해 실행 순서를 조정

control Statements

- conditional statements
 - if - else
 - switch
- Repetition statements (loop)
 - while
 - do - while
 - for
- Branching statements
 - break
 - continue
 - return
- Method call

if

- 조건에 따르는 실행 여부 control
- if (condition)
statement(or block)
- Example
 - if ($a+b \neq 0$)
 - {
 - $pa = a/(a+b);$
 - $pb = b/(a+b);$
 - }

if - else

- if (condition)
 statement(or block)
[else if (condition)
 statement(or block)
 :
[else
 statement(or block)]]

```
if (a < b) {  
    min = a;  
    max = b;  
}  
else if (a > b) {  
    min = b;  
    max = a;  
}  
else {  
    System.out.println("a is equal to b");  
}
```

Nested if

- Example

```
— if (num1 < num2)
{
    if (num1 < num3)
        min = num1;
    else
        min = num3;
}
else
{
    if (num2 < num3)
        min = num2;
    else
        min = num3;
}
```

Example

```
import java.util.Scanner;

public class Grade {
    public static void main(String[] args) {
        int score;
        char grade;

        Scanner input = new Scanner(System.in);
        System.out.print("점수 입력: ");
        score = input.nextInt();

        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';

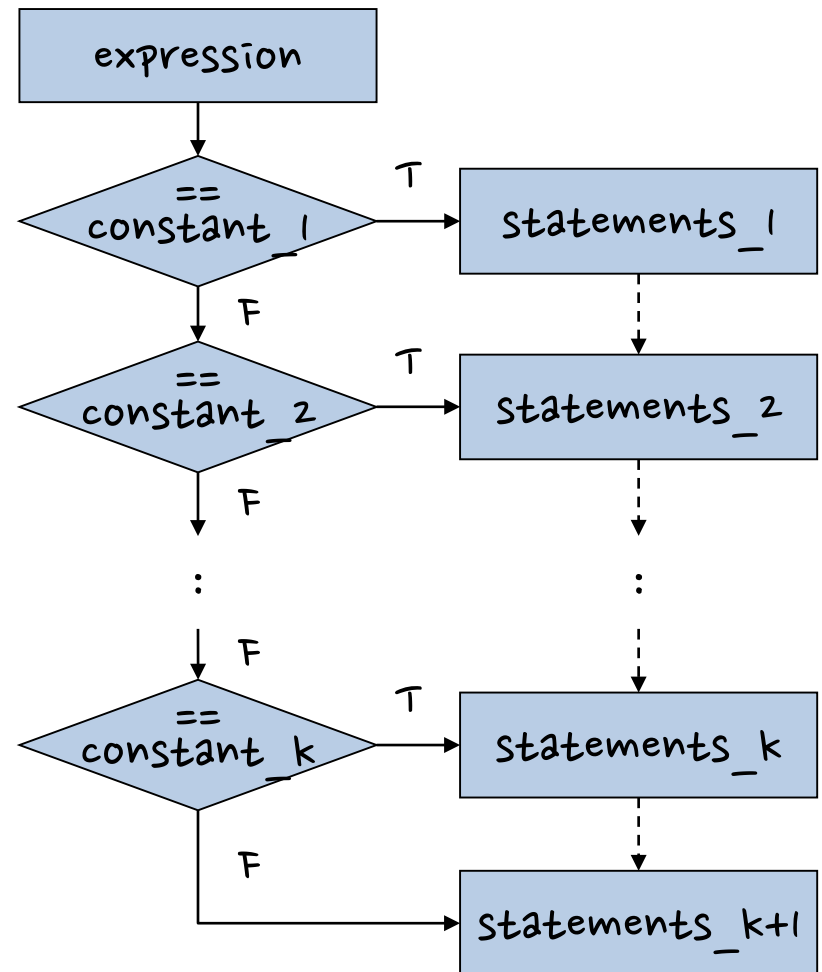
        System.out.println("점수: "+score+"\t 학점: "+grade);
    }
}
```

점수 입력: 85
점수: 85 학점: B

switch - case

- 수식 값에 따른 여러 개의 조건을 검사

- ```
switch (expression) {
 case constant_1:
 statements_1
 case constant_2:
 statements_2
 :
 case constant_k:
 statements_k
 default:
 statements_k+1
}
```





## switch - case

- 각 case에 해당하는 statements만 수행하고 싶다면 break 사용
- switch (expression) {  
    case constant 1:  
        statements<sub>1</sub>  
        break;  
    case constant 2:  
        statements<sub>2</sub>  
        break;  
    :  
    case constant k:  
        statements<sub>k</sub>  
        break;  
    default:  
        statements<sub>k+1</sub>  
}

# Switch - case

- Example

```
import java.util.Scanner;

public class GradeSwitch {

 public static void main(String[] args) {
 int score, category;
 char grade;

 Scanner input = new Scanner(System.in);
 System.out.print("점수 입력: ");
 score = input.nextInt();

 category = score/10;
 switch(category) {
 case 10:
 grade = 'A';
 break;
 case 9:
 grade = 'A';
 break;
 case 8:
 grade = 'B';
 break;
 case 7:
 grade = 'C';
 break;
 case 6:
 grade = 'D';
 break;
 default:
 grade = 'F';
 }
 System.out.println("점수: "+score+"\t 학점: "+grade);
 }
}
```

# while

- 조건이 만족하는 동안 문장을 반복
- while (condition)  
statement(or block)
- Example
  - int count = 1, sum = 0;  
while (count <= 100) {  
sum = sum + count;  
count++;  
}

# while

- Example

```
import java.util.Scanner;

public class Average {
 public static void main(String[] args) {
 int total = 0, score, count = 0;
 float average;

 Scanner input = new Scanner(System.in);
 System.out.print("점수 입력 (0은 끝): ");
 score = input.nextInt();

 while (score != 0) {
 total += score;
 count++;
 score = input.nextInt();
 }

 if (count == 0)
 System.out.println("입력 없음");
 else {
 average = (float)total/count;
 System.out.println("총점: "+total);
 System.out.println("평균: "+average);
 }
 }
}
```

# for

- 조건을 만족하는 동안 문장을 반복
- 정해진 횟수만큼 반복할 때 편리
- `for ( init_expression ; condition ; inc_expression )  
statement(or block)`
- Example
  - `for (count=1 ; count<=100 ; count++)  
sum += count;`

# for

- 수행 절차

- init\_expression

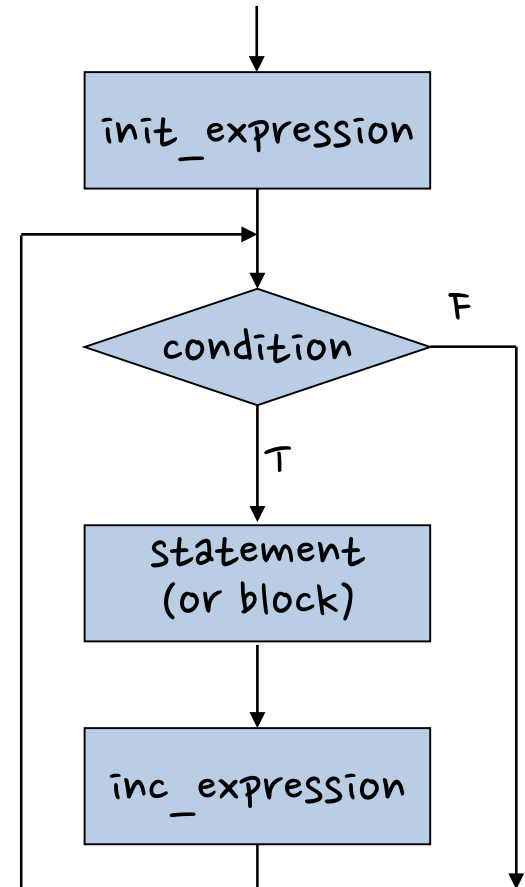
- 초기화를 위한 수식
- 반복 시작 전 한 번만 수행

- condition

- 종료 조건
- statement(or block) 수행 전 연산

- inc\_expression

- condition에 나타나는 변수의 증가/감소를 위한 수식
- statement(or block) 수행 후 연산



## while vs. for

- `init_expression`  
`while (condition) {`  
    `statements`  
    `inc_expression`  
`}`
- `for (init_expression ; condition ; inc_expression) {`  
    `statements`  
`}`
- `inc_expression`이 일정하게 증가/감소하는 연산인 경우  
for의 가독성이 높음

# while vs. for

```
public class ConvertTemperature {
 public static void main(String[] args) {
 float f, c;

 for (f = 0 ; f <= 100 ; f = f+10) {
 c = (f-32)*5/9;
 System.out.println(f+"° F = "+c+"° C");
 }
 }
}
```

```
public class ConvertTemperature {
 public static void main(String[] args) {
 float f, c;

 f = 0;
 while (f <= 100) {
 c = (f-32)*5/9;
 System.out.println(f+"° F = "+c+"° C");
 f = f+10;
 }
 }
}
```



# Nested Loop

- Loop statement(or block)에 다른 loop를 포함
- 외부 루프의 각 반복에 대해서 내부 루프가 완전히 반복되므로 다차원의 반복 작업 수행 가능

```
public class NestedLoop {
 public static void main(String[] args) {
 int i, j;

 for (i = 1 ; i < 10 ; i++) {
 System.out.println(i+" 단");
 for (j = 1 ; j < 10 ; j++)
 System.out.println(i+" x "+j+" = "+i*j);
 System.out.println();
 }
 }
}
```

---

1 단

1 x 1 = 1  
1 x 2 = 2  
1 x 3 = 3  
1 x 4 = 4  
1 x 5 = 5  
1 x 6 = 6  
1 x 7 = 7  
1 x 8 = 8  
1 x 9 = 9

2 단

2 x 1 = 2  
2 x 2 = 4  
2 x 3 = 6  
2 x 4 = 8  
2 x 5 = 10  
2 x 6 = 12  
2 x 7 = 14  
2 x 8 = 16  
2 x 9 = 18

# break / continue

- break

- 수행 중인 block을 즉시 종료 (loop 탈출)

- while ( $i \leq 1000$ ) {  
     $sum += i++$ ;  
    if ( $sum > 10000$ )  
        break;  
}

- continue

- block의 처음으로 이동 (loop 유지)

- for ( $i = 0 ; i \leq 100 ; i++$ ) {  
    if ( $i \% 2 == 0$ )  
        continue;  
     $sum += i$ ;  
}

## 시작

- 임의의 정수를 입력 받아 각 자리수의 합을 구하는 프로그램

정수를 입력하세요 : 12345

자리수의 합 : 15

- 높이를 입력 받아 삼각형 형태를 출력하는 프로그램

높이를 입력하세요 : 5

```
 *


```

시작

- 영문 문자열을 입력 받고 그 문자열에 속한 각 모음의 개수를 출력

영문 스트림을 입력하세요

abcdefghijklmnopqrstuvwxyz

a의 개수 1

e의 개수 1

i의 개수 1

o의 개수 1

u의 개수 1

다른 문자 개수 21

— 문자열 입력은 String class 사용

- e.g String a;  
a = input.next();

— String class의 length(), charAt() method 활용

- e.g. a.length() -> String a의 길이를 반환  
a.charAt(i) -> String a의 i번째 문자를 반환

# Summary

- control statements
  - conditional (if - else, switch - case)
  - Repetition (while, do - while, for)
  - Branching (break, continue)