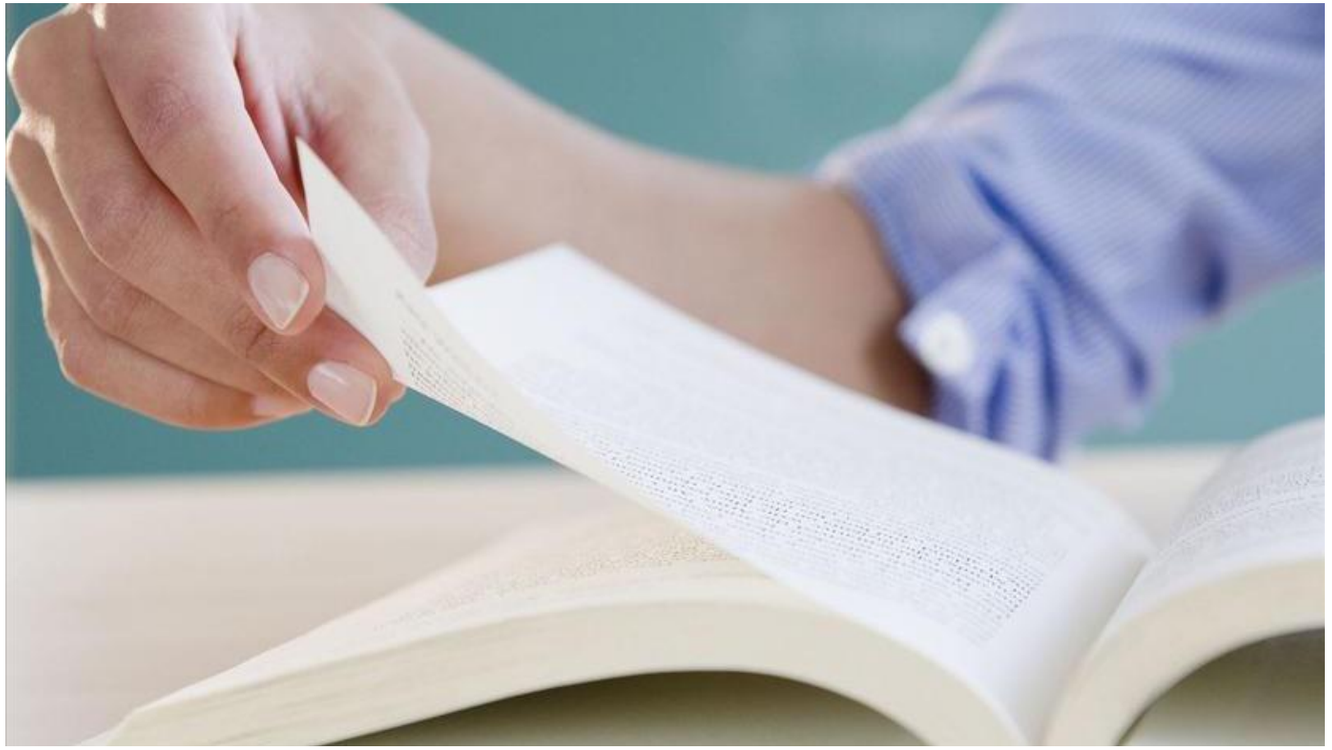
Introduction to computer programming A
LAB3&4

贾艳红 Jana
Email:jiayh@mail.sustech.edu.cn

LAB OBJECTIVES

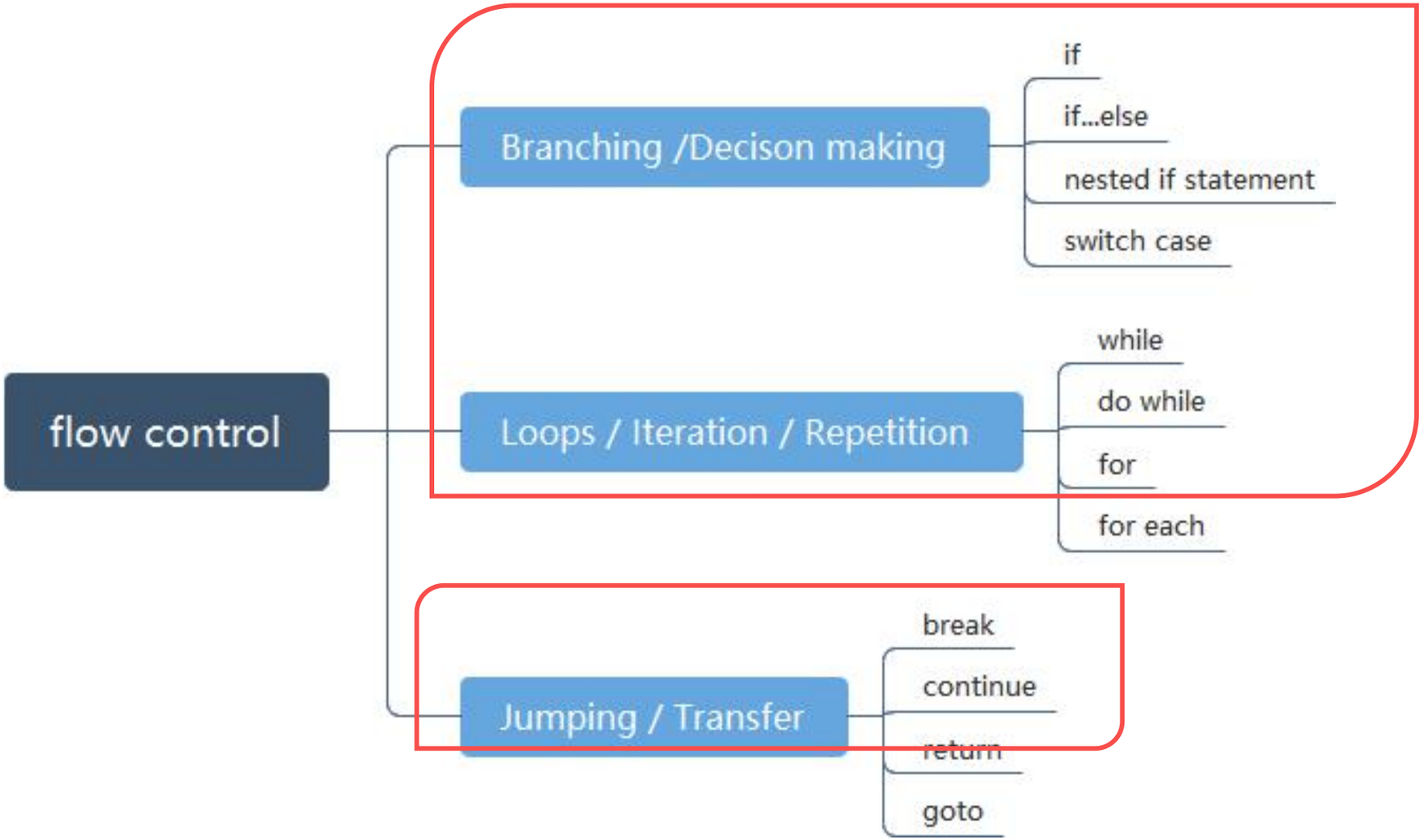
1. Learn how to use the **while** repetition statement to execute statements in a program repeatedly
2. Learn how to use the **do...while**, **for** repetition statement to execute statements in a program
3. Learn how to use the **switch** selection statements to choose among alternative actions.
4. Learn how to use the **break** and **continue** statements in a program

knowledge points



- **Loop Control**
- **break、contine**
- **The switch statement**

Flow Control



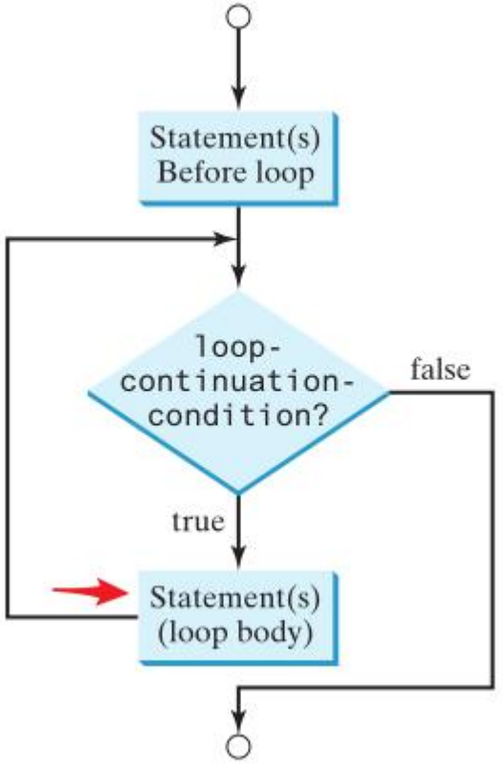
Loop Control

No.	Loop & Description
1	while loop
	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	do...while loop
	Like a while statement, except that it tests the condition at the end of the loop body.
2	for loop
	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.

Loop Control - while Loops

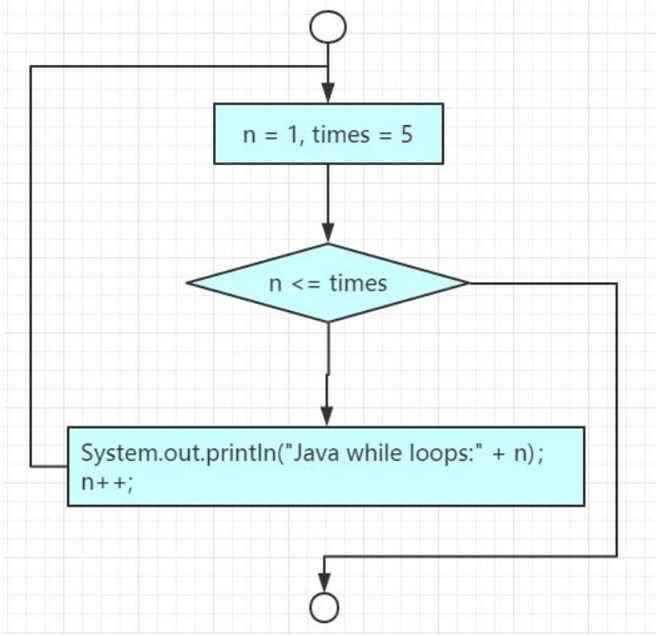
The syntax for the `while` loop is as follows:

```
while (loop-continuation-condition) {
  // Loop body
  Statement(s);
}
```



Example:

```
public class SimpleWhileDemo {

    public static void main(String[] args) {
        /* local variable Initialization */
        int n = 1, times = 5;
        /* while loops execution */
        while (n <= times) {
            System.out.println("Java while loops:" + n);
            n++;

        }
    }
}
```



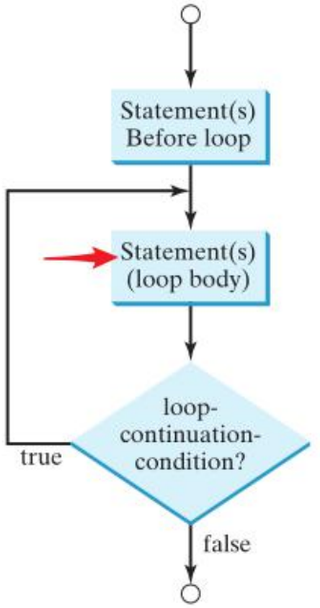
Console

<terminated> SimpleWhileDemo [Java

```
Java while loops:1
Java while loops:2
Java while loops:3
Java while loops:4
Java while loops:5
```

Loop Control - do-while Loops

```
do {
  // Loop body;
  Statement(s);
} while (loop-continuation-condition);
```



Example:

```
public class SimpleDoWhileDemo {

    public static void main(String[] args) {
        /* local variable Initialization */
        int n = 1, times = 5;

        /* do-while loops execution */
        do {
            System.out.println("Java do while loops:" + n);
            n++;
        } while (n <= times);

    }

}
```

Console

<terminated> SimpleWhileDemo [Java

```
Java while loops:1
Java while loops:2
Java while loops:3
Java while loops:4
Java while loops:5
```

Compare the Difference Between While and do-while

Compare the difference between While statements and do-while statements. What does the following two programs output?

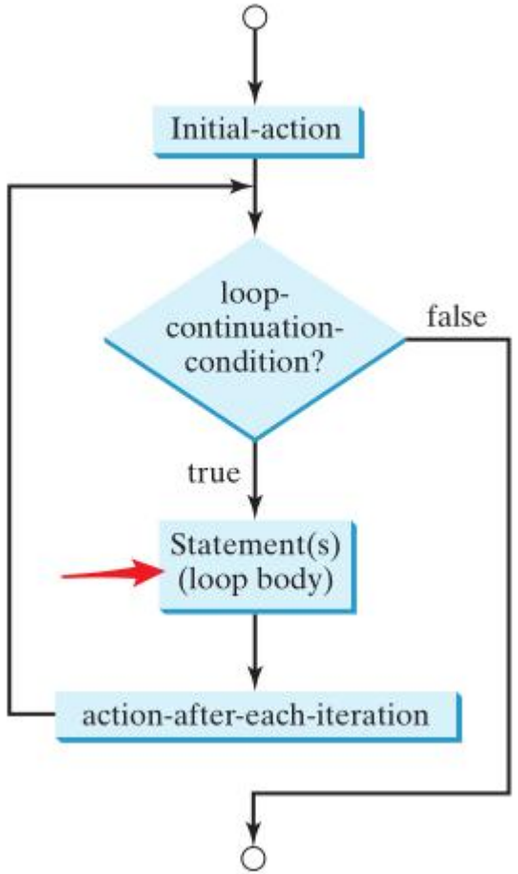
```java
public class SimpleWhileDemo {

    public static void main(String[] args) {
        /* local variable Initialization */
        int n = 1, times = 0;
        /* while loops execution */
        while (n <= times) {
            System.out.println("Java while loops:" + n);
            n++;
        }
    }
}
```

```java
public class SimpleDoWhileDemo {

    public static void main(String[] args) {
        /* local variable Initialization */
        int n = 1, times = 0;
        /* do-while loops execution */
        do {
            System.out.println("Java do while loops:" + n);
            n++;
        } while (n <= times);

    }

}
```

Loop Control - for Loops

The syntax of a **for** loop is as follows:

```
for (initial-action; loop-continuation-condition;
     action-after-each-iteration) {
  // Loop body;
  Statement(s);
```

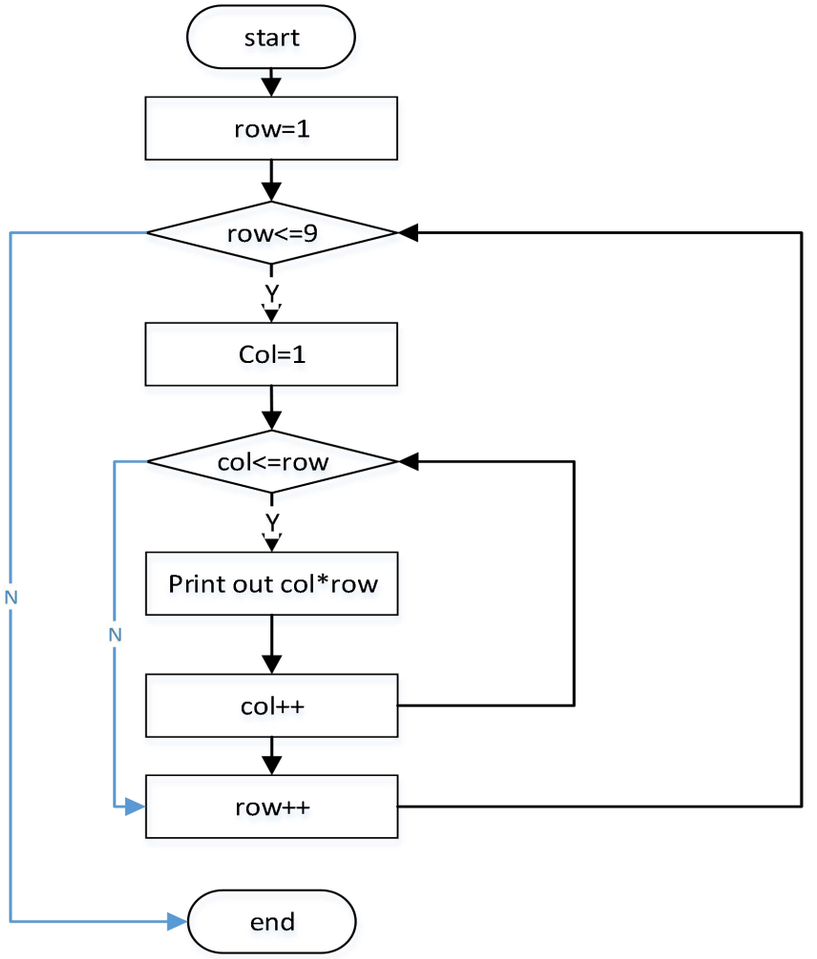


Example:

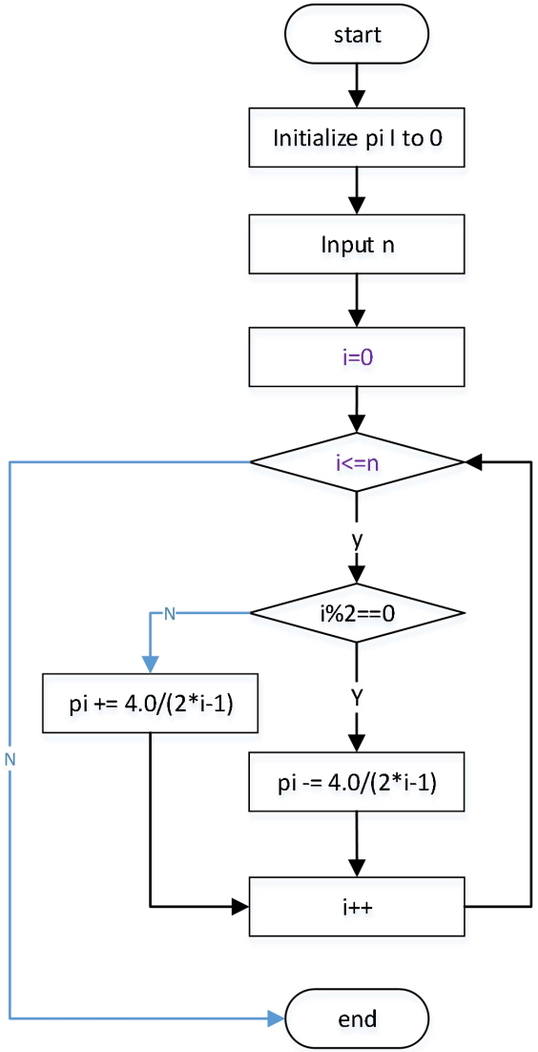
```
public class SimpleForDemo {

    public static void main(String[] args) {
        /* local variable Initialization */
        int n = 1, times = 5;
        /* for loops execution */
        for (n = 1; n <= times; n = n + 1) {
            System.out.println("Java for loops:" + n);
        }

    }

}
```

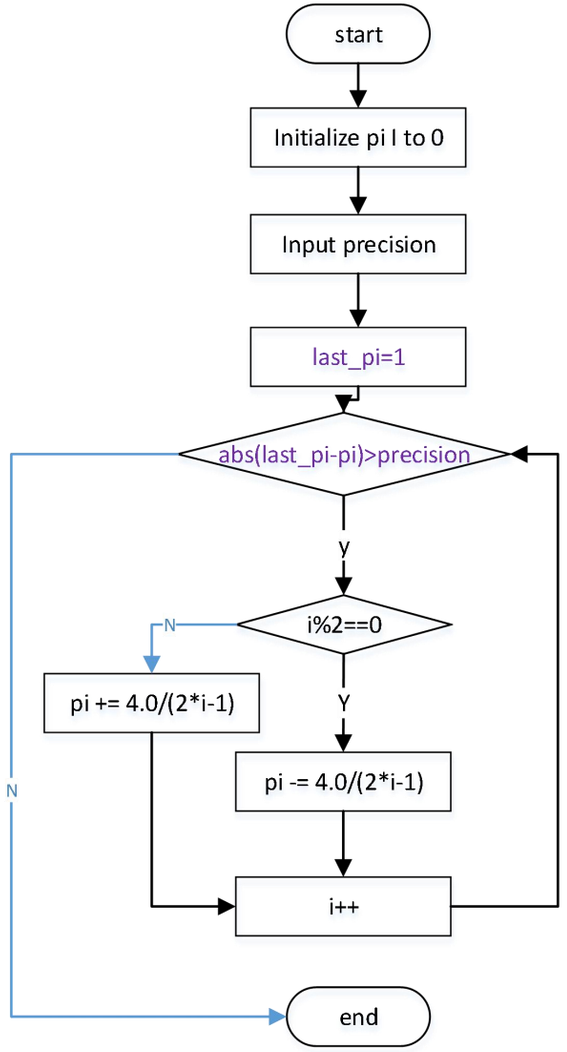
```
Java for loops:1
Java for loops:2
Java for loops:3
Java for loops:4
Java for loops:5
```

Lab exercise1



Lab exercise3 & lab exercise4





Difference Between for and while loop

They can both do the same things:

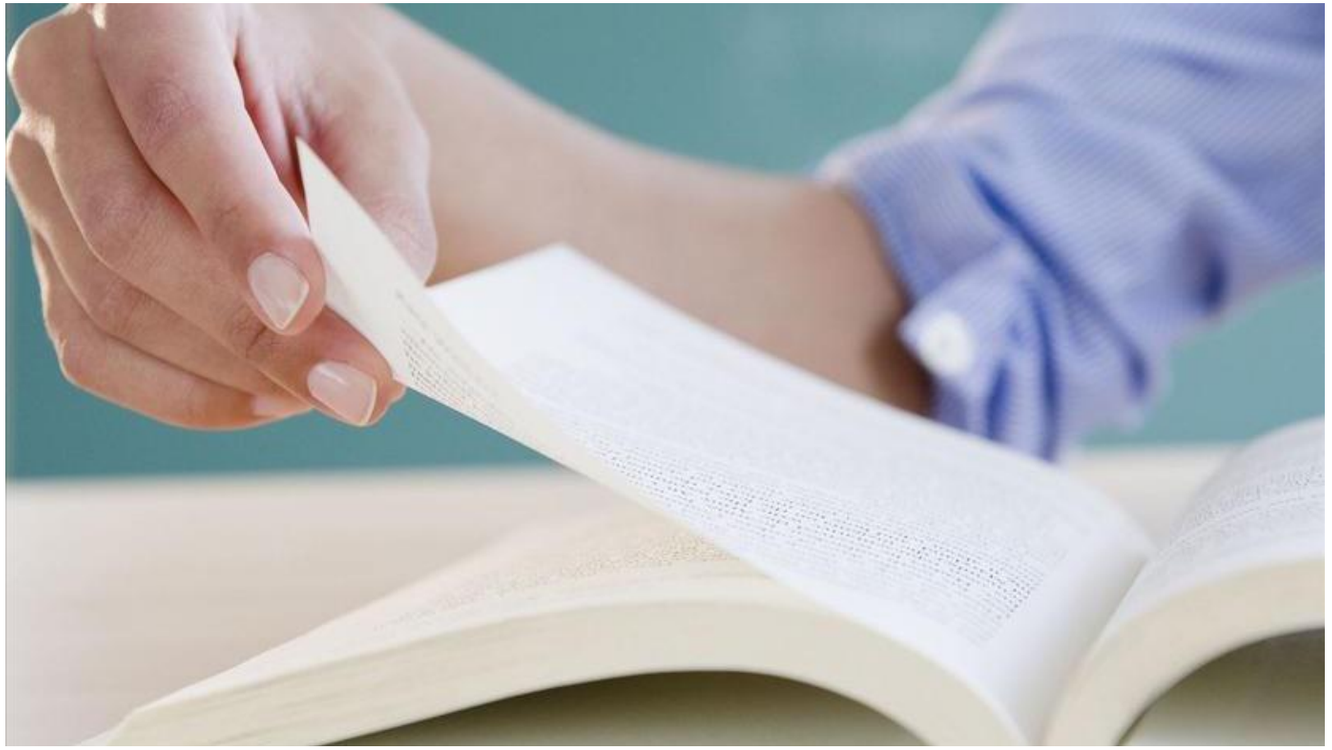
```
for (int i = 0; i < 3; i++) {
    //this goes around 3 times
}
```

```
int x = 0;
while (x < 5) {
    x++;
}

//this loop goes around 5 times, but it's not really known at the time it's created.
```

but in general if you **know how many times** you will loop use a **for**, other wise use a while.

```
import java.util.Scanner;
public class Diff_For_While {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        char ch ;
        boolean again = true;
        while(again)
        {
            System.out.print("Do you want to go again ?: ");
            ch = in.next().charAt(0);
            if(ch == 'n')
                again = false;
        }
    }
```

knowledge points



- Loop Control
- break、contine
- The switch statement

break statement

Example: Use of break statement

```
public class TestBreakJumping {

    public static void main(String[] args) {
        for (int j = 0; j < 10; j++) {
            if (j == 5) {
                break;
            }
            System.out.println(j);
        }
        System.out.println("outside of for loop");
    }

}
```

```
0
1
2
3
4
outside of for loop
```

- Inside the switch case to come out of the switch block.
- Within the loops to break the loop execution based on some condition.

Invalid, break without switch or loop!!!

continue Jumping statement

Example: To print odd numbers.

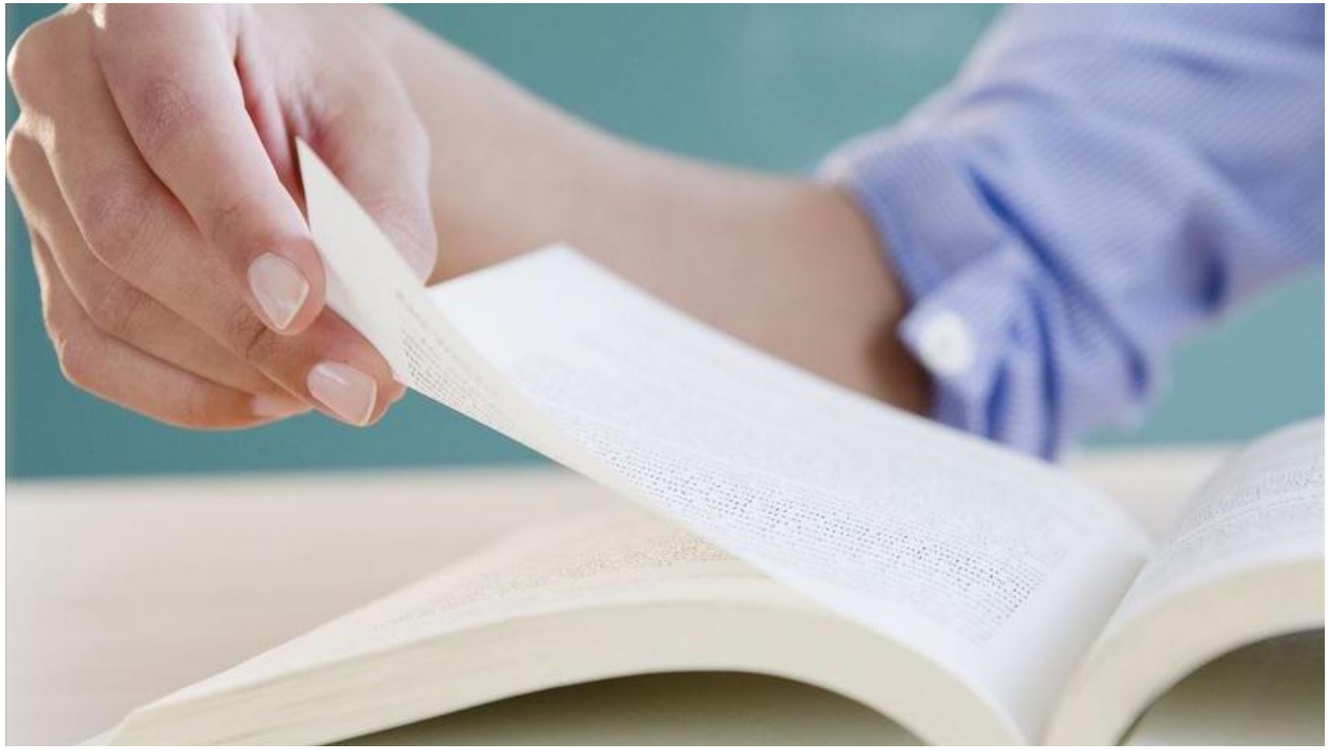
```
public class TestContinueJumping {

    public static void main(String[] args) {
        for (int j = 1; j <= 10; j++) {
            if (j % 2 == 0) {
                continue;
            }
            System.out.println(j);
        }
    }

}
```

```
1
3
5
7
9
```

This statement is used only within looping statements!!!

- When the continue statement is encountered, then it skip the current iteration and the next iteration starts.
- The remaining statements in the loop are skipped. The execution starts from the top of loop again.
- We can use continue statement to skip current iteration and continue the next iteration inside loops.

knowledge points

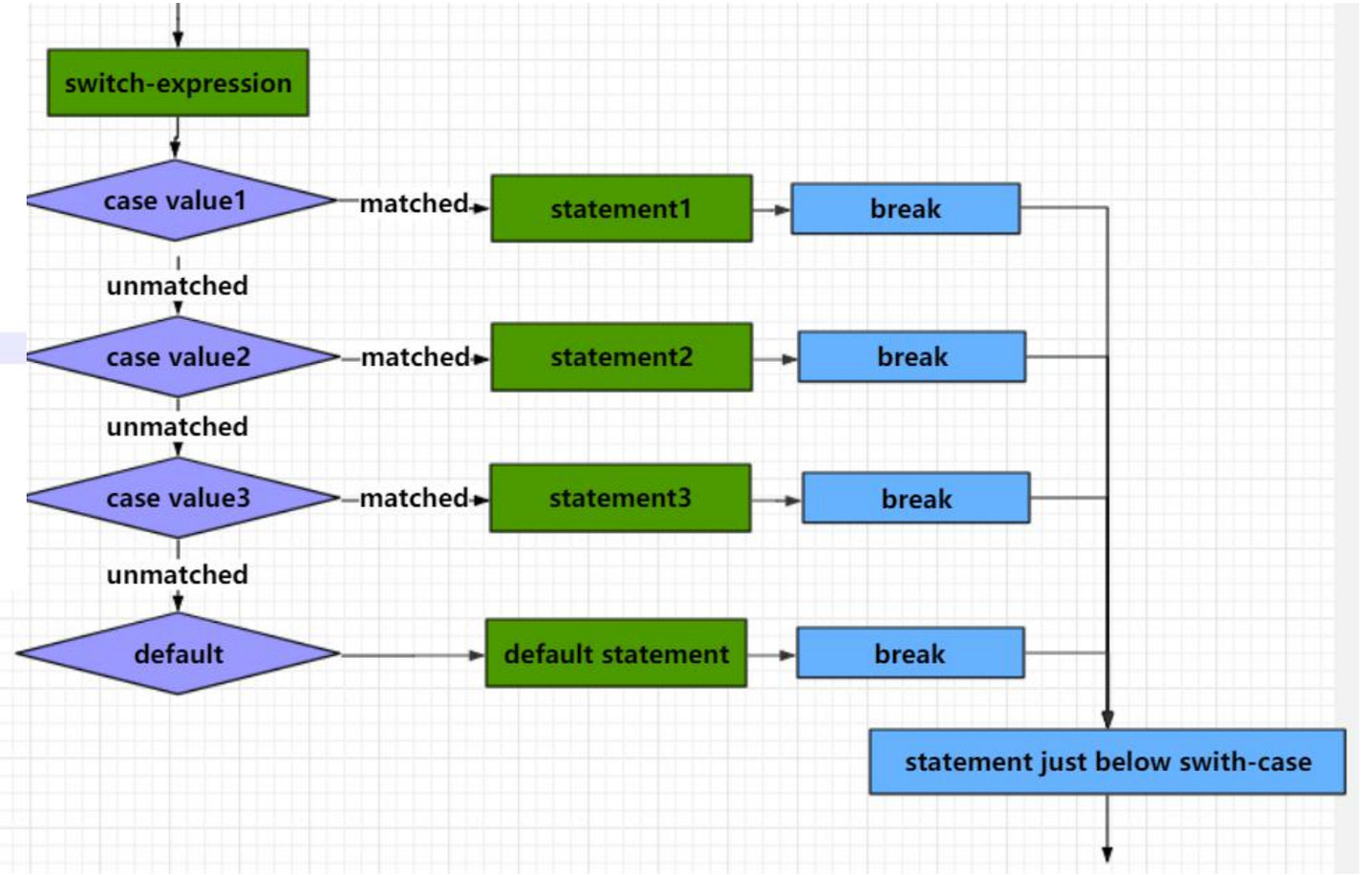


- Loop Control
- break、contine
- The switch statement

The *switch* statement

Syntax:

```
// The switch-expression must yield a value
// of char,byte,short,int,or String type
switch (switch-expression)
{
  case value1:
    //execute statement1
    statement1;
    break;
  case value2:
    //execute statement1
    statement2;
    break;
  ...
  case valueN:
   //execute statementN
    statementN;
    break;
  default:
    //execute statementDefault
    statementDefault;
}
```



switch...case

Example: Write a program print bonus amount based on the grade of employee. An employee can be of grade A, B, C or default (anything other than A, B & C)

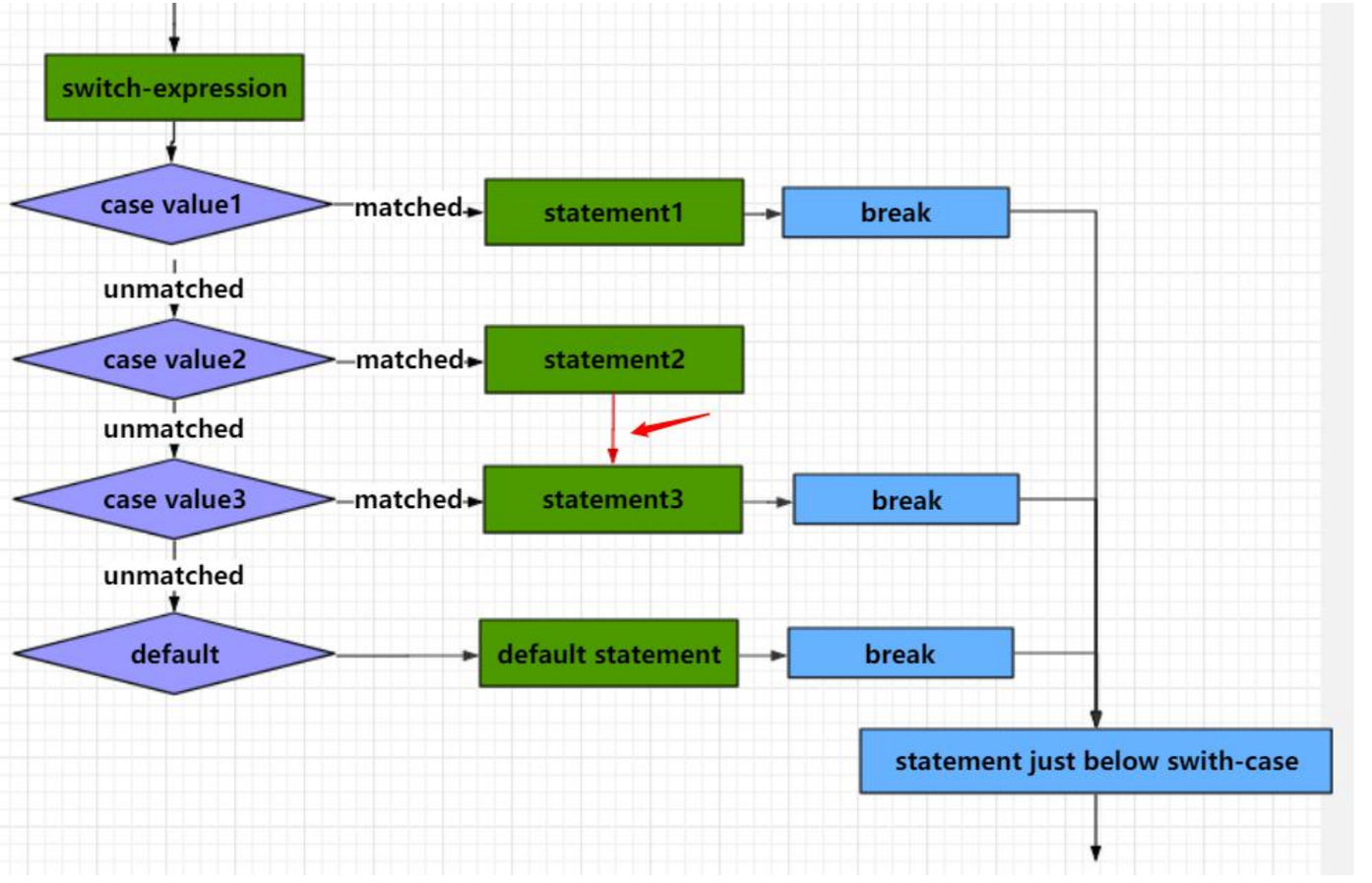
```
2  public class TestSwitchCase {
3
4      public static void main(String[] args) {
5          char Grade ='B';
6          switch (Grade){
7          case 'A':
8              System.out.println("You are Grade A Employee: Bonus= "+ 2000);
9              break;
10         case 'B':
11             System.out.println("You are Grade B Employee: Bonus= "+ 1000);
12             break;
13         case 'C':
14             System.out.println("You are Grade C Employee: Bonus= "+ 500);
15             break;
16         default:
17             System.out.println("You are Default Employee: Bonus= "+ 100);
18             break;
19         }
20     }
21
22 }
```

```
You are Grade B Employee: Bonus= 1000
```

The *switch* statement

```
switch (switch-expression)
{
  case value1:
    //execute statement1
    statement1;
    break;
  case value2:
    //execute statement1
    statement2;
    //break;
  ...
  case valueN:
   //execute statementN
    statementN;
    break;
  default:
    //execute statementDefault
    statementDefault;
}
```

If the break statement is omitted



Difference between if and switch

Piece #1

```
if(opt == 1){
    //add
    result = number1+number2;
}
if(opt == 2){
    //sub
    result = number1-number2;
}
if(opt == 3){
    //multiply
    result = number1*number2;
}
if(opt == 4){
    //divide
    result = number1/number2;
}
```

Piece #2

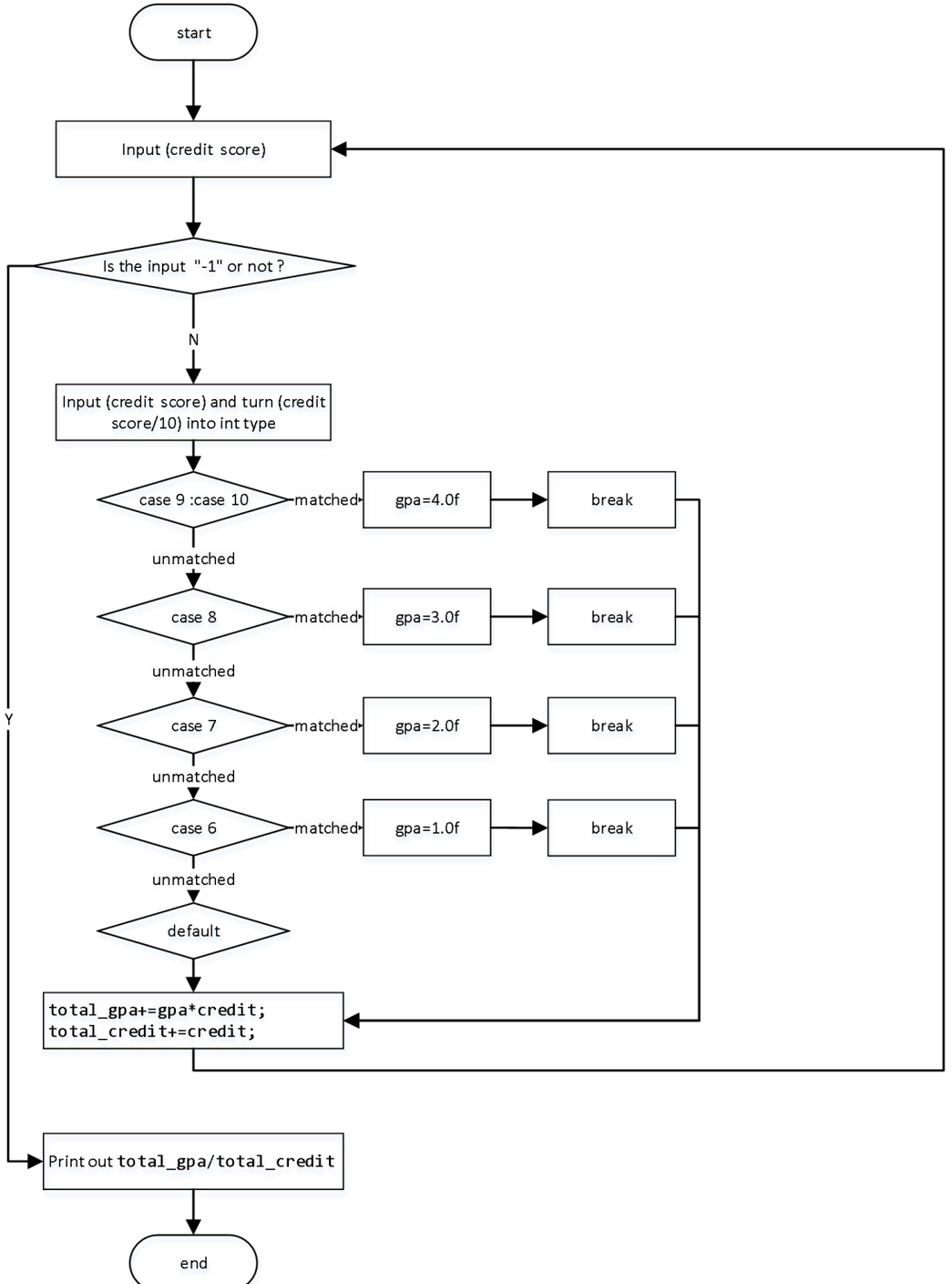
```
if(opt == 1){
    //add
    result = number1+number2;
}else if(opt == 2){
    //sub
    result = number1-number2;
}else if(opt == 3){
    //multiply
    result = number1*number2;
}else if(opt == 4){
    //divide
    result = number1/number2;
}
```

Piece #3

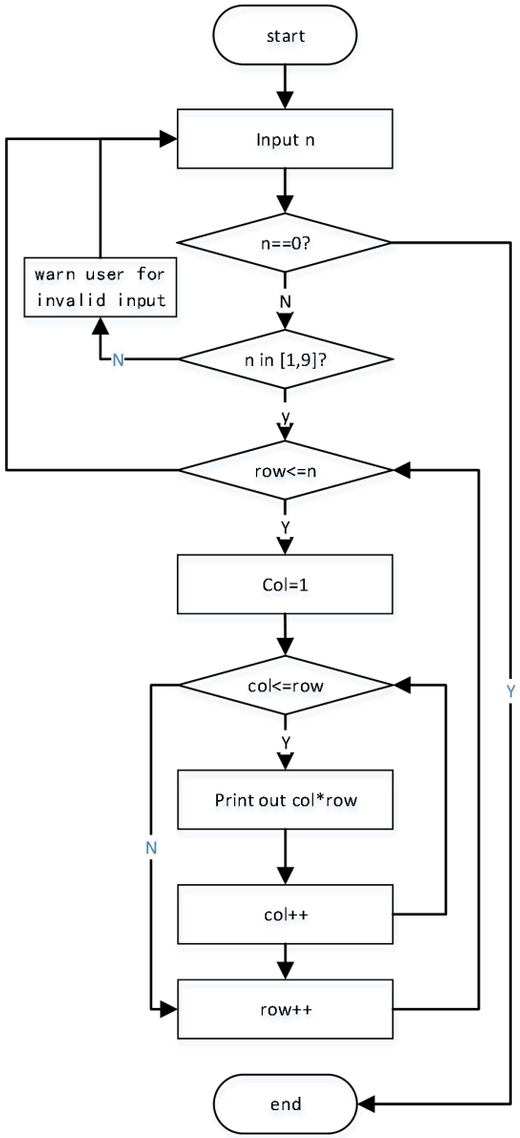
```
switch(opt){
    case 1:
        //add
        result = number1+number2;
        break;
    case 2:
        //sub
        result = number1-number2;
        break;
    case 3:
        //multiply
        result = number1*number2;
        break;
    case 4:
        //divide
        result = number1/number2;
        break;
    default:
        printf("The operator must be one of 1,2,3, and 4\n");
        return; //退出
}
```

use switch if you have three or more alternatives

Lab4 1-2


start
Input (credit score)
Is the input "-1" or not ?
N
Y
Input (credit score) and turn (credit score/10) into int type
case 9 :case 10
matched
gpa=4.0f
break
unmatched
case 8
matched
gpa=3.0f
break
unmatched
case 7
matched
gpa=2.0f
break
unmatched
case 6
matched
gpa=1.0f
break
unmatched
default
total_gpa+=gpa*credit;
total_credit+=credit;
Print out total_gpa/total_credit
end

Lab4-4


start
Input n
n==0?
warn user for invalid input
N
n in [1,9]?
N
Y
row<=n
Y
Col=1
col<=row
Y
Print out col*row
col++
row++
N
Y
end

4 Exercises

Complete the exercises in the **2021S-Java-A-Lab3&4.pdf** and submit to the blackboard as required.

贾艳红 Jana
Email:jiayh@mail.sustech.edu.cn