Mod 1 Basic Computing Concepts (6) Algorithms

6 Algorithms

6.1 What is an Algorithm?

An **Algorithm** is a step by step description of how a problem should be solved. Algorithms can be expressed in many kinds of notation including:

- Natural languages (e.g. Maltese, English etc.)
- Pseudocode
- Flowcharts
- Programming languages

6.2 Flowcharts

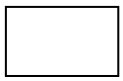
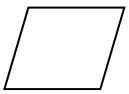
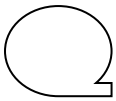
A **Flowchart** describes an algorithm graphically by special symbols each having a particular meaning. There are basically two types of flowcharts:

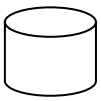
- System flowchart
- Program flowchart

A **System Flowchart** shows how data flows in an information system. A Program Flowchart is a detailed description of the logic of a program.

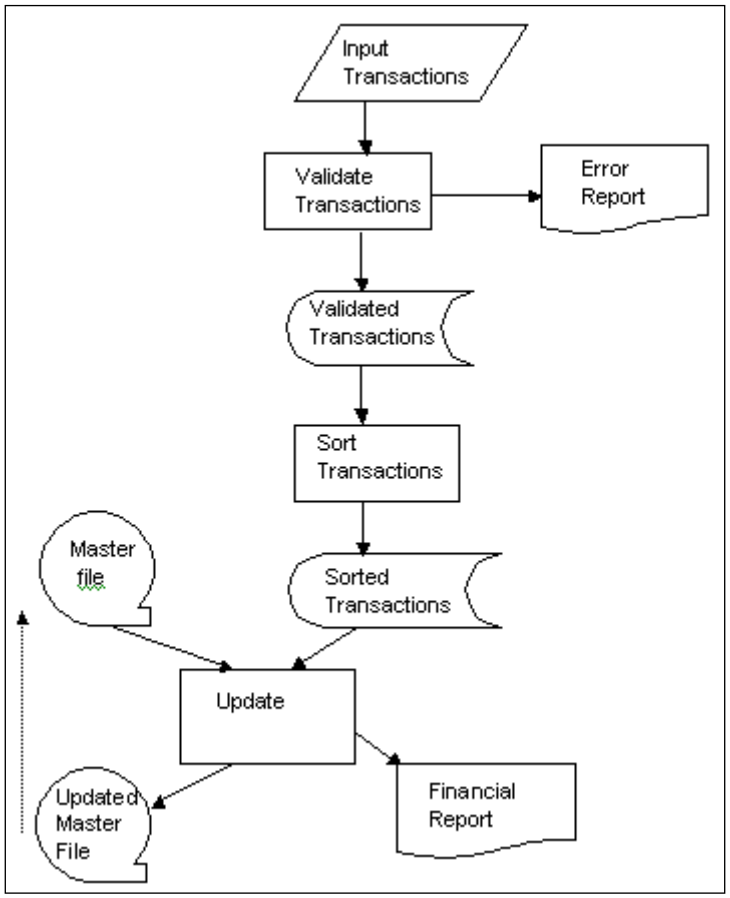
6.2.1 System Flowchart

The following table shows some symbols used in system flowcharts.

Symbol	Symbol Name	Symbol Description
	Process	Shows a statement or a number of statements. A statement directs the computer to perform a specified action.
	Input / Output	Indicates input to or output from a process.
	Document	Printed or written data.
	Sequential Access Storage	For example a file stored on tape.

	Stored Data	For example file stored on disk.
	Magnetic Disk	This symbol depicts a database.

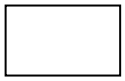
System Flowchart Symbols

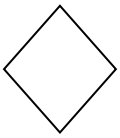
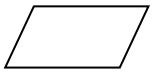


A System Flowchart

6.2.2 Program Flowchart

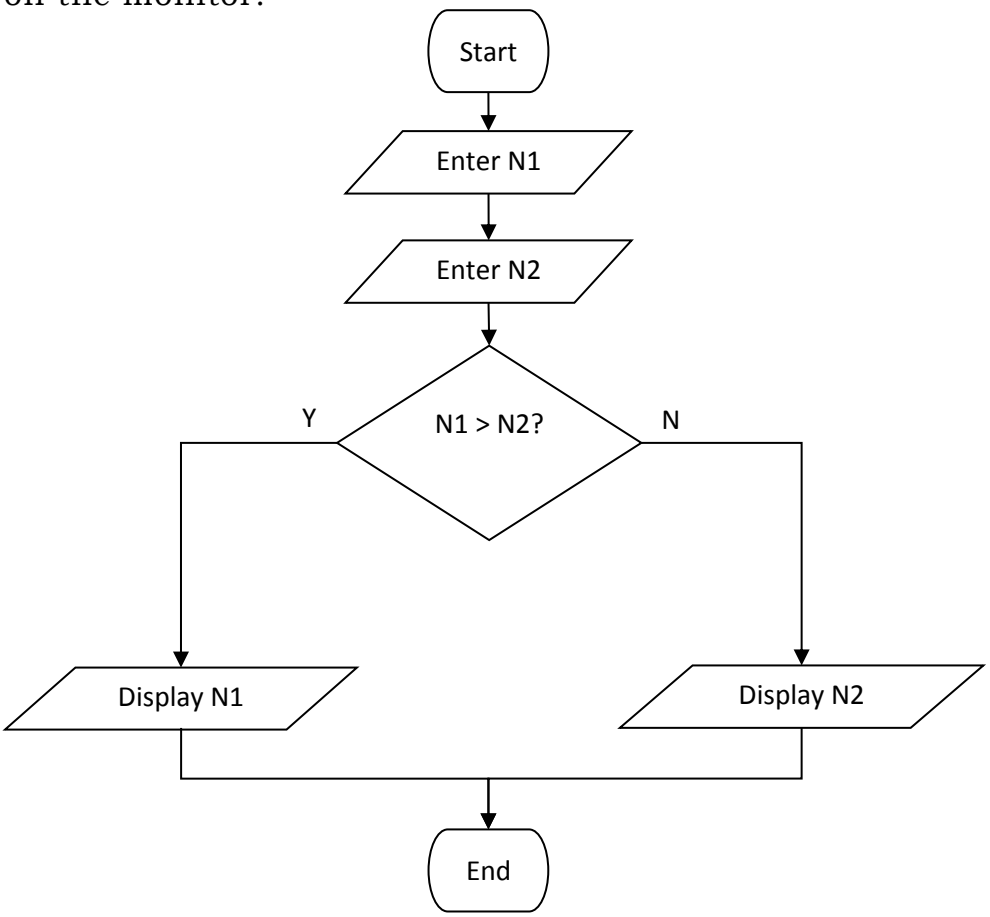
Below is a table showing the basic program flowchart symbols.

Symbol	Symbol Name	Symbol Description
	Terminator	Terminators show the start and stop points in a process
	Process	Shows a statement or a number of statements. A statement directs the computer to perform a specified action.

	Predefined Process (Subroutine)	A predefined process symbol is a marker for another process that is defined elsewhere.
	Decision	Indicates a question or branch in the process flow. Typically a decision shape is used when there are 2 options but can be used when there are more than two options.
	Input / Output	Indicates input to or output from a process.
	Connector	This symbol is used to connect two parts, on the same page, together.
	Off-page connector	This symbol is used to connect two parts of the same flowchart distributed on two pages.

Program Flowchart Symbols

The following diagram shows a simple example of a flowchart. In the algorithm two numbers N1 and N2 are inputted and the largest of them is displayed on the monitor.



A Flowchart

Note in the above algorithm that the first value is given the name N1 and the second value is given the name N2. This is the concept of a

Variable. A variable is created to hold a certain type of value e.g. the variables N1 and N2 hold numbers. Another variable (let's call it NAM) can hold names (strings). In this case N1 or N2 cannot hold names and NAM cannot hold numbers. If a variable (say B) is of type Boolean then it can hold only the values 'true' or 'false'.

6.2.2.1 Exercise

Express the following algorithms as Flowcharts.

a) A program asks the user to give it the name of the user's father. Then it asks for the father's age. It then asks for the user's name and finally his or her age. The program calculates the difference between the ages and then displays the sentence "Your father is ... years bigger than you".

b) A program asks a user to enter the number of sides of a polygon. If the number 4 is entered then the program displays the message "the shape is a quadrilateral. Otherwise it displays the message "the program is not a quadrilateral".

c) A club is organising a buffet. When one books for the buffet one will pay €12 per person if the number of persons being booked is more than 4. Otherwise the price is €15 per person. Write a program that asks the user to enter the number of persons to be booked for the buffet and then the program will display the total amount to be paid.

6.3 Pseudocode

Pseudocode (besides flowcharts and other techniques) is another way to express algorithms. When writing the algorithm in pseudocode the author builds the logic of the algorithm on the logic of high-programming languages. The pseudocode is a actually a draft of a program. It is a program written without using the exact syntax. Later the pseudocode can be transformed into real code with relative ease. Pseudocode is a mixture of program constructs and a natural language (in our case English).

The following algorithm is the one expressed as flowchart in the preceding section. It finds the larger of two numbers.

```
begin
    input number N1
    input number N2
        if N1 is larger than N2
            then display N1
            else display N2
end
```

Pseudocode of a program that finds the larger of two numbers

6.3.1 Exercise

Express the following algorithms in Pseudocode.

a) A bus goes on a trip and makes one stop in the middle. In this stop a number of people leave the bus and other new passengers get on the bus. Write a program that asks how many passengers started the trip, how many passengers stopped in mid-trip and how many passengers went into the bus in mid-trip. The program will then display the number of passengers that arrived at the end of the trip.

b) This program is about whether profit was made from the sale of an object. The program asks the user "how much was the object bought?" and the user enters the price. Then the program asks how much it was sold. The program then displays either the message "a profit was made" or "no profit was made".

c) A businessman organising a meal in a restaurant. Adults pay €10 and children €6. If the total price to be paid exceeds €30, a discount of 10% is given. Write a program that calculates the amount to be paid after being given the number of adults and children.

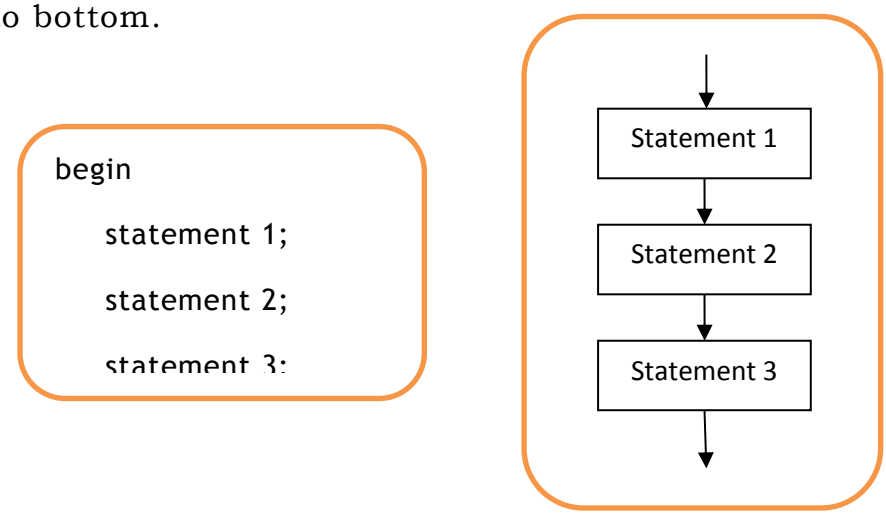
6.4 High Level Language (Java) Structures

A high level language has the following structures:

- Linear structure
- Selection structure
- Repetition structure
- Branch structure

6.4.1 Linear Structure

A **linear structure** means that instructions are executed sequentially from top to bottom.



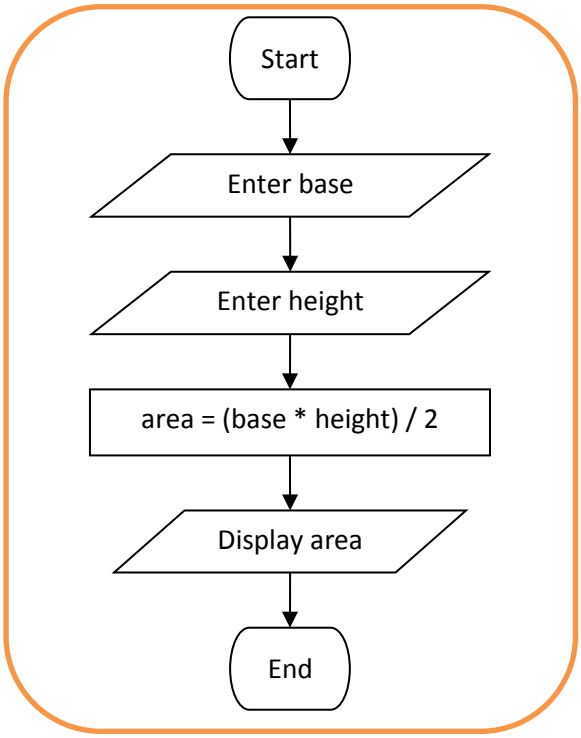
Pseudocode and a flowchart that show the linear structure of a program

Let us give an example. Consider a program that calculates the area of a triangle.

```
begin

    display "Area of a triangle"
    input base
    input height
    area = (base * height) / 2
    display area

end
```



Finding the Area of a Triangle

6.4.1.1 Exercise

(i) Express the following algorithm by means of a flowchart.

A program is given a value for the temperature in Centigrade. The program calculates its Fahrenheit equivalent and displays it on the monitor. The formula for the conversion from Centigrade to Fahrenheit is F=C*9/5+32.

(ii) Express the following algorithm by means of pseudocode.

A program is given the value of a radius. It first calculates the area of circle with that radius and then calculates the volume of a sphere with the same radius. It then displays the results on the monitor.

6.4.2 Selection Structure

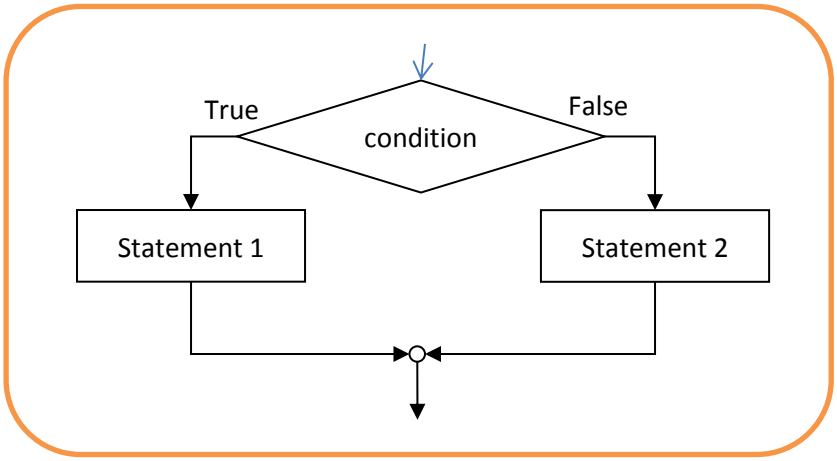
A **selection** (or **selective** or **conditional**) structure contains a **condition** which determines which statement/s to execute. There are two selection structures. These are:

(a) if-then-else statement
(b) switch (or 'case') statement

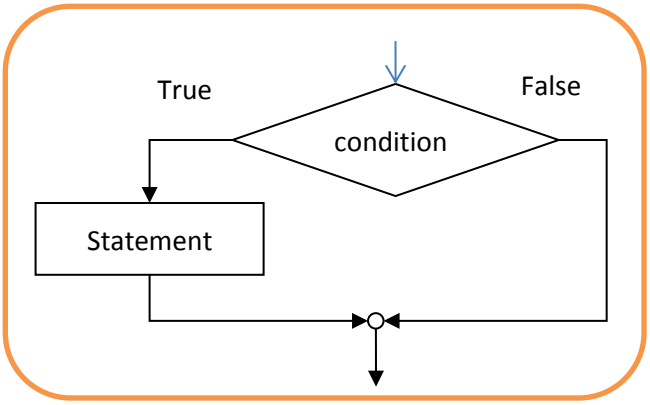
6.4.2.1 If-then-else Structure

The **if-then-else** structure makes the program choose one of two possible statements. The pseudocode and flowchart of the if-then-else structures are shown hereunder.

```
if (condition is true)
      then perform statement 1;
      else perform statement 2;
```

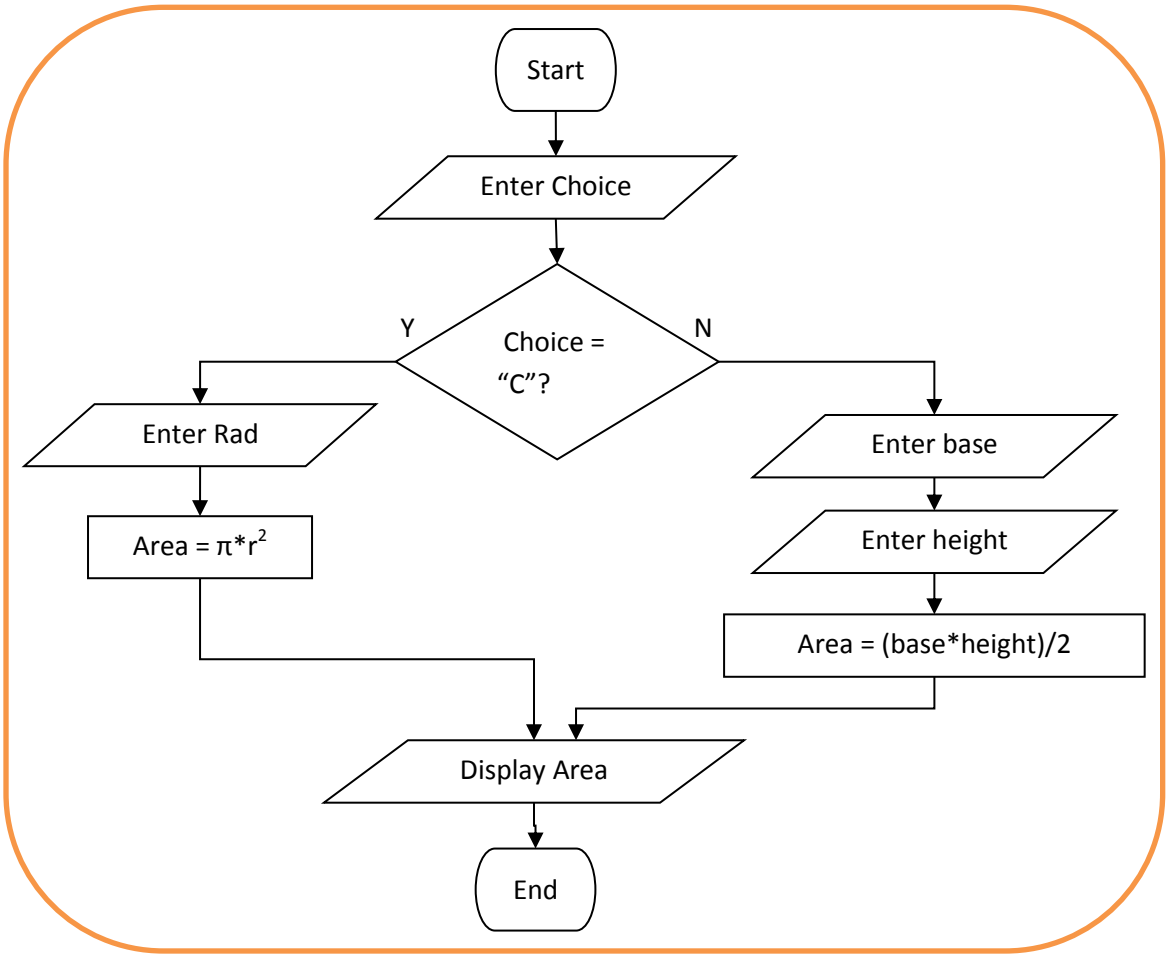



```
if (condition is true)
      then perform statement;
```



if ... then ... else and if ... then expressed as pseudocode and flowchart

Below is an algorithm (flowchart and pseudocode) that makes use of the if-then-else statement:



Which program is the above flowchart representing?

6.4.2.1.1 Exercise

1 Express the flowchart in the above diagram in pseudocode.

2 Express the following algorithms as flowcharts.

(a) A program receives in input a name of a person and his/her age. If the age is less than 18 the program displays the message "you cannot vote". Otherwise it displays the message "you can vote".

(b) A program receives in input a mark. If the mark is less than 50 it displays the word "fail". Otherwise if the mark is less than 75 it displays the word "pass". In the case the mark is 75 or over it displays the word "distinction".

(c) A program receives three numbers and it decides whether they were entered in ascending order or not.

3 Express the following algorithms in pseudocode.

(a) A program receives in input a mark. If the mark is less than 50 the program displays the word "fail". Otherwise it displays the word "pass".

(b) A program that is given three numbers and it displays the largest one.

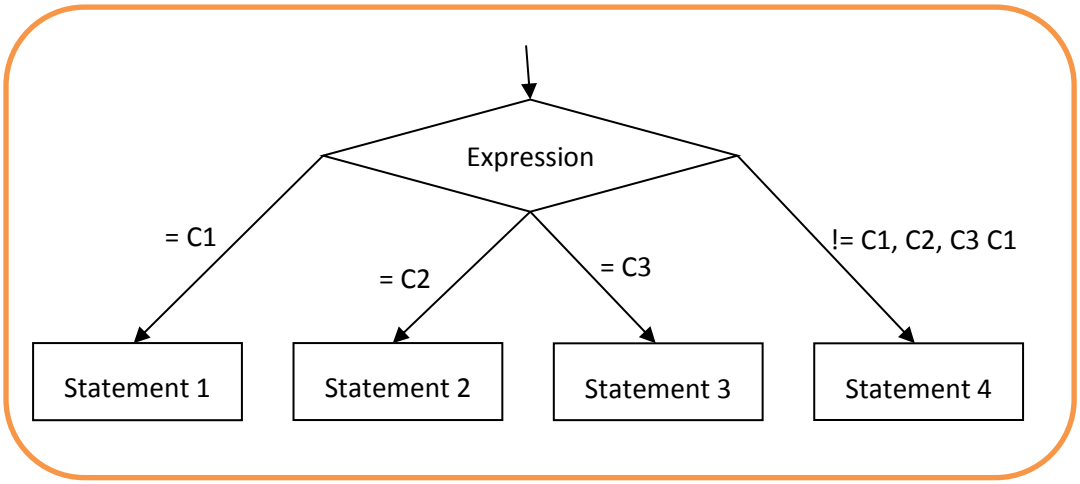
(c) A program receives three numbers in input and displays the numbers in ascending order.

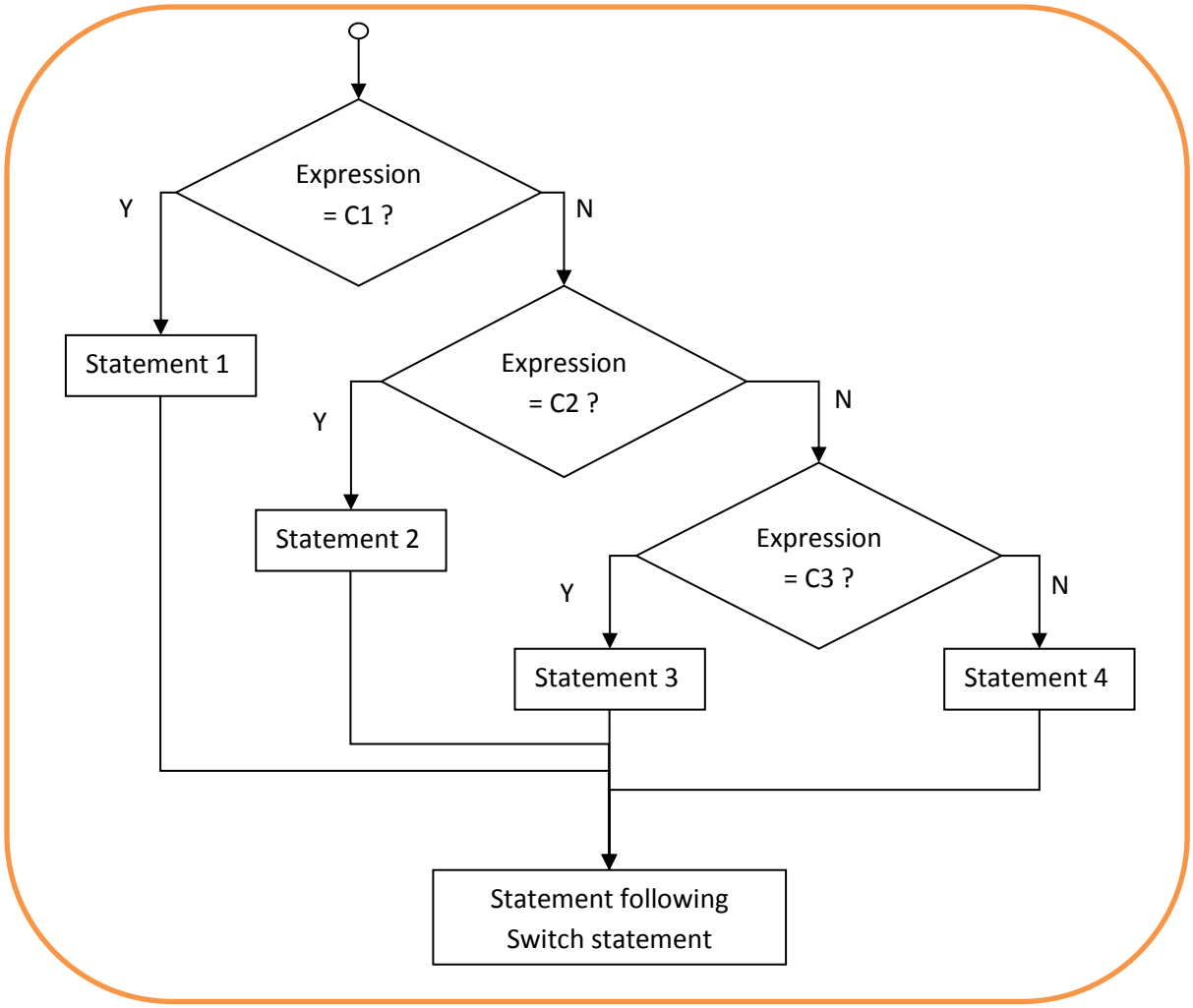
6.4.2.2 Switch Structure

```
switch (expr) {
    case c1:   statements to be performed if expr is
               equal to the value c1
    case c2:   statements to be performed if expr is
               equal to the value c2
    case c3:   statements to be performed if expr is
               equal to the value c3
    . . .
    default:   statements to be performed if expr is not
               equal to any of c1, c2, c3, etc. }
```

Pseudocode of the Switch structure

The **Switch** (or **Case**) statement offers multiple choices. A possible pseudocode is shown in the diagram shown above. The flowchart can be drawn in more than one way. Two examples are shown hereunder.





Flowcharts to represent the Case statement

6.4.3 Repetition Structure

The **repetition** structure is also known as **iterative** or loop **structure**. There are three such structures. In these structures a sequence of statements is repeated if a condition is met. These three structures are the following:

a. while statement
b. do-while (or 'repeat-until') statement
c. for statement

6.4.3.1 While Statement

The following diagram shows a program snippet written in Java making use of the **while loop**. Note the **condition** at the start of the loop. Note also that this piece of programming makes use of a **counter** that is initialised outside the loop and is incremented inside the loop.

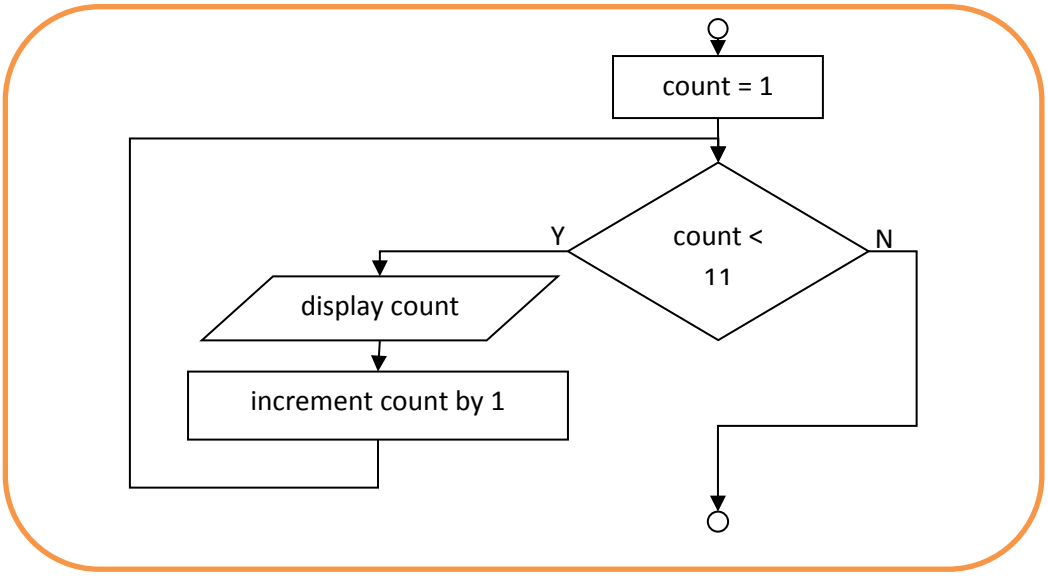
```
int count = 1;
while (count < 11)
{
    System.out.println("Count is: " + count);
    count++;
}
```

Program snipped in Java making use of the while loop

The pseudocode of the above program can be expressed as:

```
give initial value of 1 to 'count'
while ('count' is less than 11)
    begin
        display count
        increment count by 1
    end
```

Pseudocode of the above program



Flowchart of the above program

The 'while loop' is called **pre-tested** because the condition is tested at the start of the loop. The flowchart of this program is shown below.

6.4.3.1.1 Exercise

What does the program in the above program do?

6.4.3.2 Do-While Statement

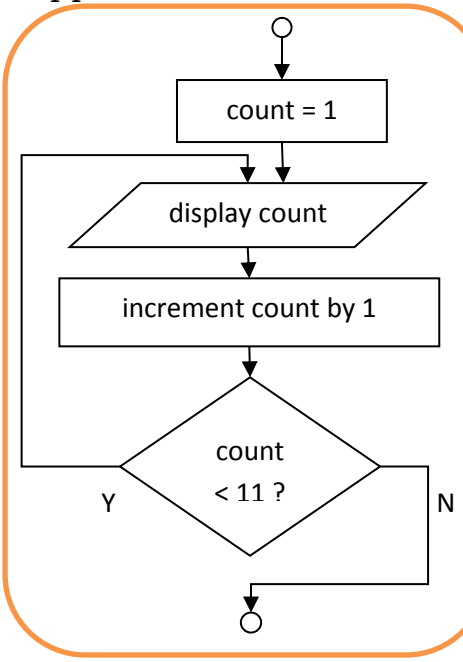
In some languages (e.g. in Pascal) the **do-while** statement is represented as a **repeat-until** statement with the same effect. The following code from a Java program makes use of the do-while statement.

```
int count = 1;
do
{
    System.out.println("Count is: " + count);
    count++;
} while (count < 11);
```

A snippet of a program in Java that makes use of the do...while statement

This kind of loop is called **post-tested** because the test that decides whether the loop should be repeated or not is found at the end of the loop. The pseudocode and flowchart of the above snippet are shown below.

```
initiate 'count' with value 1
  do
     display 'count'
     increment the value of 'count' by 1
  while (the value of 'count' is less than
11)
```



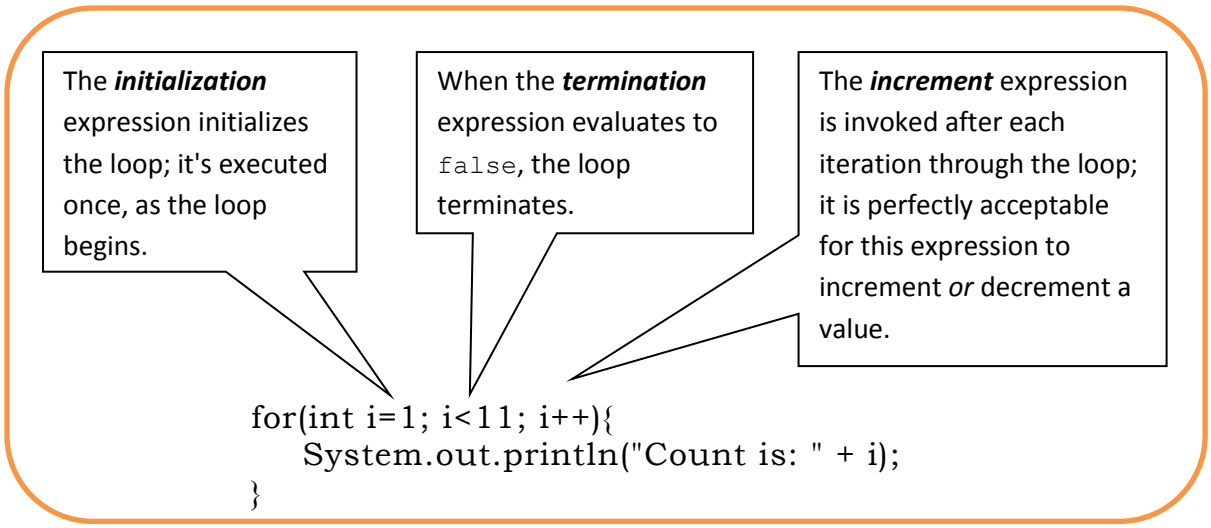
Pseudocode and flowchart of the program above

6.4.3.2.1 Exercise

What does the above program do?

6.4.3.3 For Statement

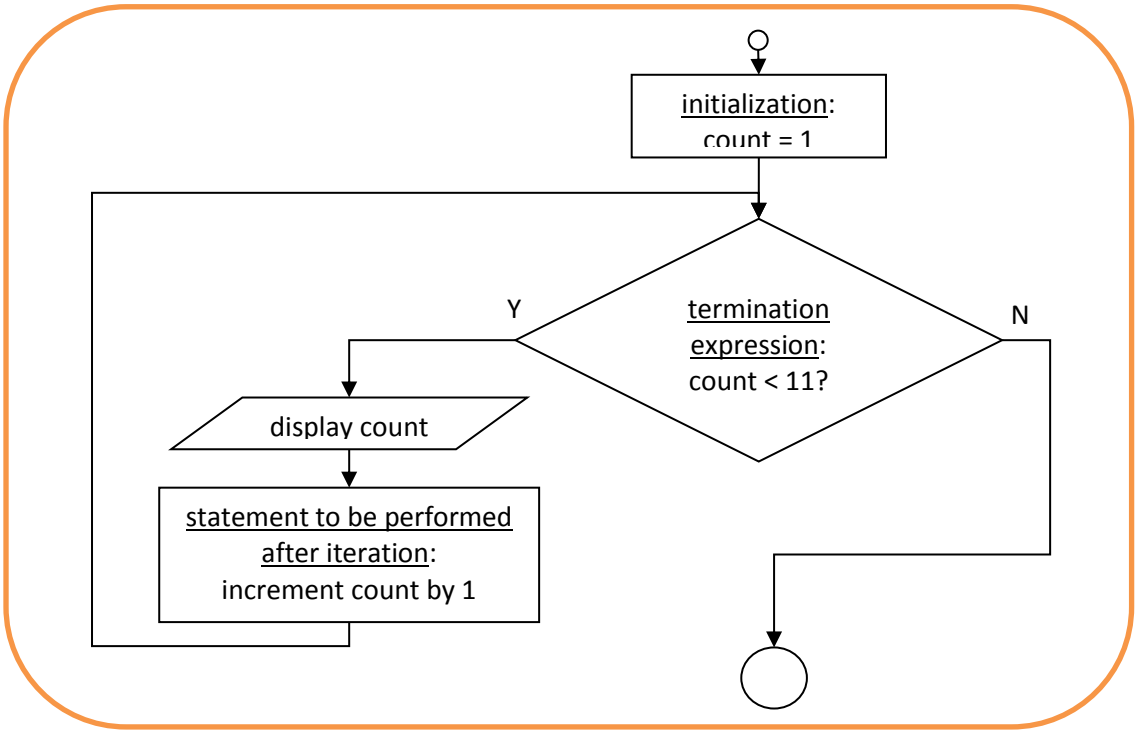
An example of the **for-statement** as used in Java is shown in the diagram below.



Use of the for loop

There are various ways how to express the above code in pseudocode but the best way to do it is to be as close to the code as possible so that once the pseudocode is ready it will be easy to translate it in Java code.

```
for-loop:
    (initialization: i = 1
    termination expression: loop while i<11
    statement after each iteration: increment i by 1)
        begin
            display ( i )
        end
```



Pseudocode and flowchart of the Java snippet shown above

6.4.3.4 Exercise

Represent the following algorithms as indicated in the brackets.

(a) A program displays the values 3, 6, 9 ... 36. (flowchart, while-loop)

(b) A program displays the values 6, 11, 16 ... 66. (pseudocode, do-while)

(c) A program is given a whole number n and then displays the first n multiples of 10 e.g. if it is given 3 it displays 10, 20, 30 (flowchart, for-loop).

(d) A program receives in input a sequence of numbers. When 0 is entered it means that there are no more numbers to input. The program finds the sum of the inputted numbers (pseudocode, while-loop).

(e) A program receives in input a sequence of numbers. When 0 is entered it means that there are no more numbers to input. The program counts how many numbers have been entered. The 0 is not counted (flowchart, do-while loop).

(f) A program receives in input a sequence of numbers. When 0 is entered it means that there are no more numbers to input. The program finds the average of the inputted numbers. 0 is not counted as one of the numbers (pseudocode, for-loop).

(g) A program receives in input a sequence of numbers. When 0 is entered it means that there are no more numbers to input. The program finds the total of the positive numbers (flowchart, while-loop).

6.4.4 Branch Statements

The **branch** statement is also known as the **jump** statement. The branch statements in Java are the following:

(a) break statement
(b) continue statement
(c) goto statement
(d) return statement

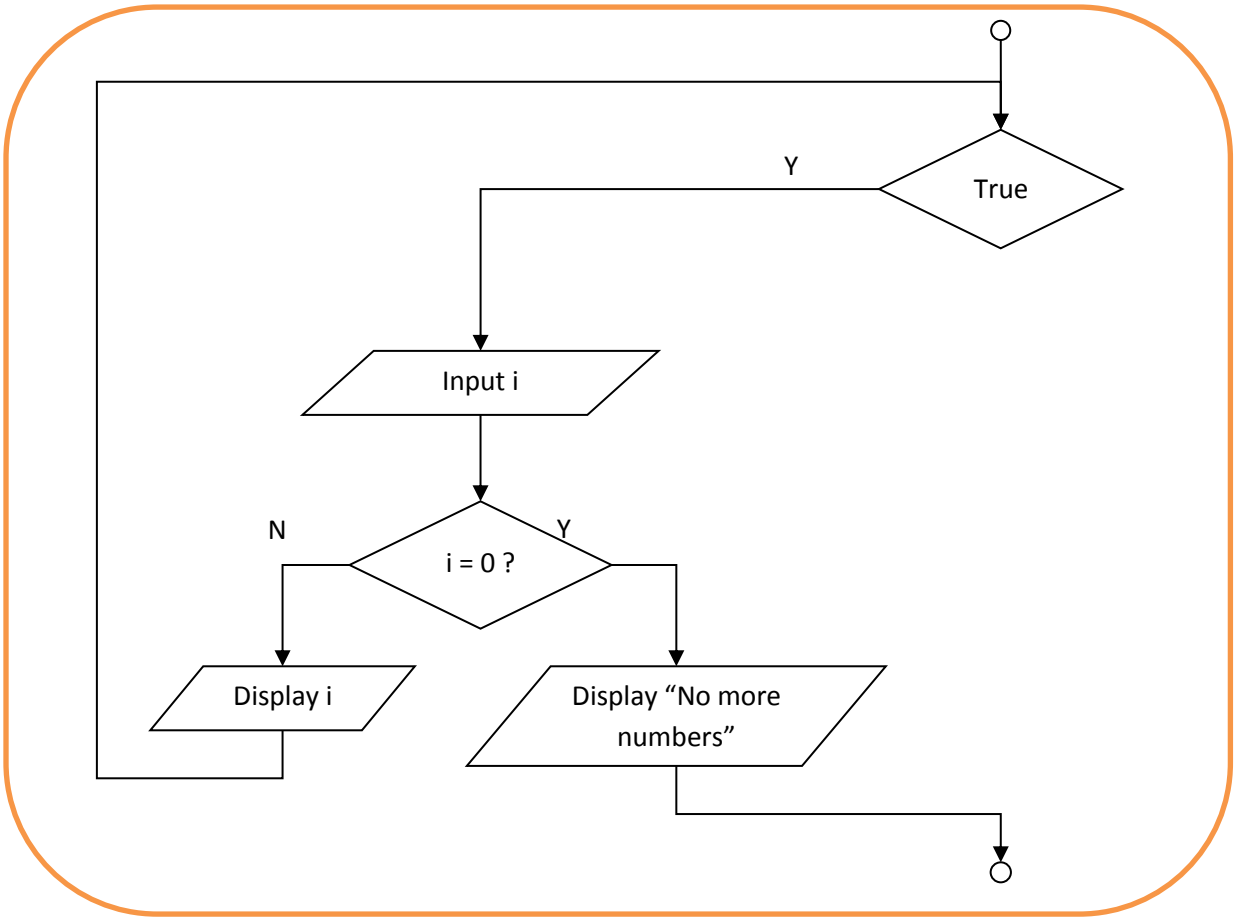
6.4.4.1 Break Statement

The **break** statement has two forms: **labelled** and **unlabelled**. The break statement is used to exit from a loop before its full completion. The diagram below shows a program snippet that makes use of an unlabelled break statement.

```
while (true)
        {
            i = scan.nextInt();
            if (i == 0)
                {
                    System.out.println ("No more
                        numbers");
                    break;
                }
            else System.out.println (i);
        }
```

Use of unlabelled break statement.

The flowchart of the snippet above is shown below.



Flowchart of unlabelled break statement.

The pseudocode of the snippet above is shown below.

```
while true do
    begin
        input integer i
        if i=0   then
                    begin
                        display (“No more numbers”)
                        break out of while loop
                    end
                else display (i)
    end
```

Pseudocode with unlabelled break statement.

An unlabelled ‘break’ statement terminates the innermost ‘switch’, ‘for’, ‘while’, or ‘do-while’ statement, but a labelled break can terminate an outer statement as shown below.

```
search:
        for (i = 0; i <= 2; i++)
        {
            for (j = 0; j <= 3; j++)
            {
                if (condition)
                {
                    foundIt = true;
                    break search;
                }
            }
        }
```

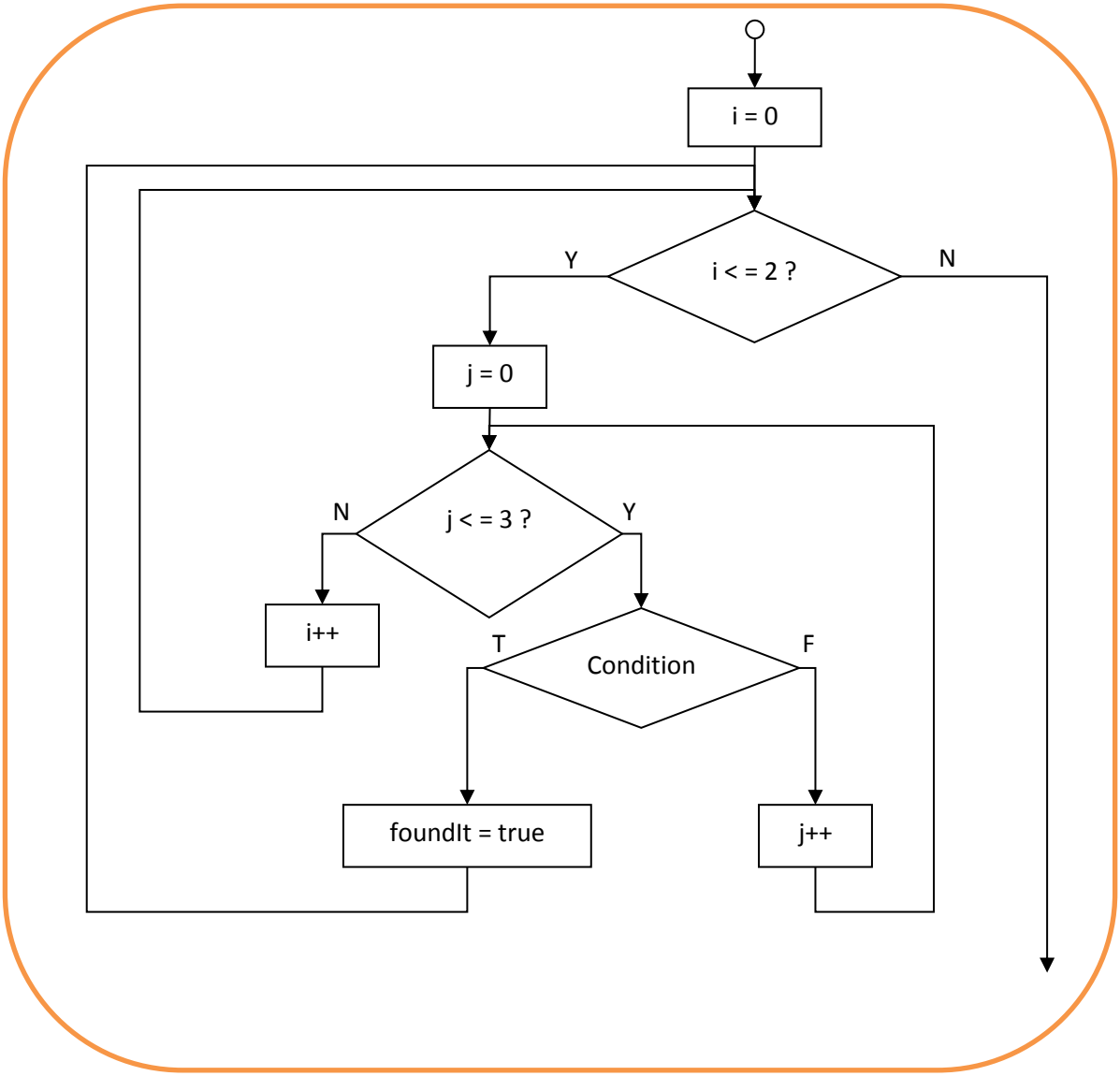
Program snippet with a labelled break statement.

A flowchart and pseudocode with labelled break statements are shown further down.

6.4.4.2 Continue Statement

The **continue** statement skips the current iteration of a ‘for’, ‘while’, or ‘do-while’ loop. The unlabelled form skips to the end of the innermost loop's body and evaluates the Boolean expression that controls the loop.

A **labelled** continue statement skips the current iteration of an outer loop marked with the given label.



Flowchart of a labelled break statement

```
for (int i = 0; i < lengthOfStrg; i++)
    {
        //interested only in p's
        if (Strg.charAt(i) != 'p')
            continue;

        // adding the number of Ps
        numPs++;
    }
```

An example of the use of the unlabelled Continue statement

```
OuterLoop: for (int i = 1; i <= limit; i++)
{
      factorial = 1;
      for (int j = 2; j <= i; j++)
      {
            if (i > 10 && i % 2 == 1)
            {
                  continue OuterLoop;
            }
            factorial *= j;
      }
      System.out.println(i + "! is " + factorial);
}
```

An example of the use of the labelled continue statement

6.4.4.3 Goto Statement

GOTO is a statement found in many computer programming languages. It is a combination of the English words 'go' and 'to'. It performs a one-way jump to another line of code. The jumped-to locations are usually identified using labels, though some languages use line numbers.

Many languages support the 'goto' statement, and many do not. In Java, 'goto' is a reserved word, but is unusable. However the 'break *label*' in Java can be considered as a kind of 'goto'. Yet the 'break *label*' cannot substitute any 'goto' use. You cannot use 'break' to transfer control to a block of code that does not enclose the 'break' statement.

The **structured program theorem** proves that the availability of the 'goto' statement is not necessary to write programs; some combination of the three programming constructs of (i) sequence, (ii) selection/choice, and (iii) repetition/iteration is sufficient to perform any computation.

6.4.4.4 Return Statement

The **return** statement exits from the current method (or procedure, function etc.) and control flow returns to where the method was invoked. The return statement has two forms: one that returns a value, and one that doesn't. In Java to return a value simply put the value (or an expression that calculates the value) after the 'return' keyword e.g. return ++count. The data type of the returned value must match the type of the method's declared return value. When a method is declared 'void', use the form of 'return' that doesn't return a value. This is simply expressed by writing 'return' followed simply by the end-of-statement (i.e. a semicolon).