Presentation Outline

1. Flow Diagram - Basic Idea
2. Control Statements

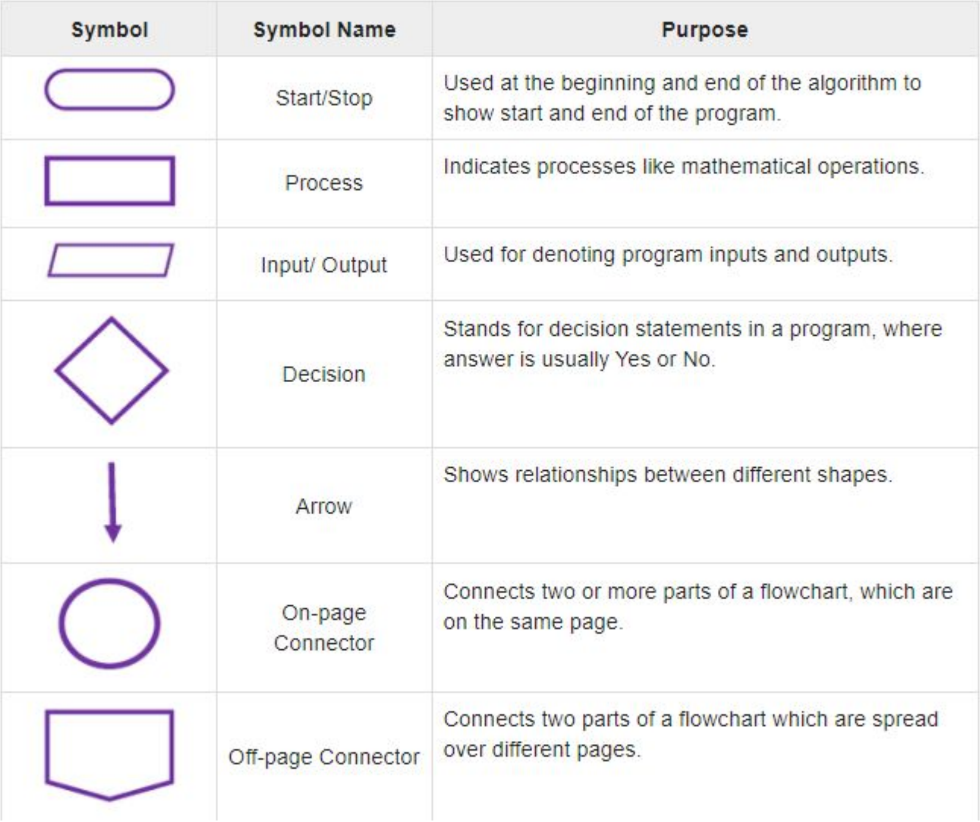
   - Decision Control (Branch- If, If-Else, If-Else If, Switch-Case)
   - Iterative Control (Loop- While, do-While, For)
   - Jump (Break, Continue, GoTo)

1. Flow Diagram - Basic Idea

- Only one start - one stop symbol
- On-page connectors:- **Numbers**
- Off-page connectors:- **Alphabets**
- General flow:-

  Top-Bottom, Left-Right

- Arrows should not cross each other



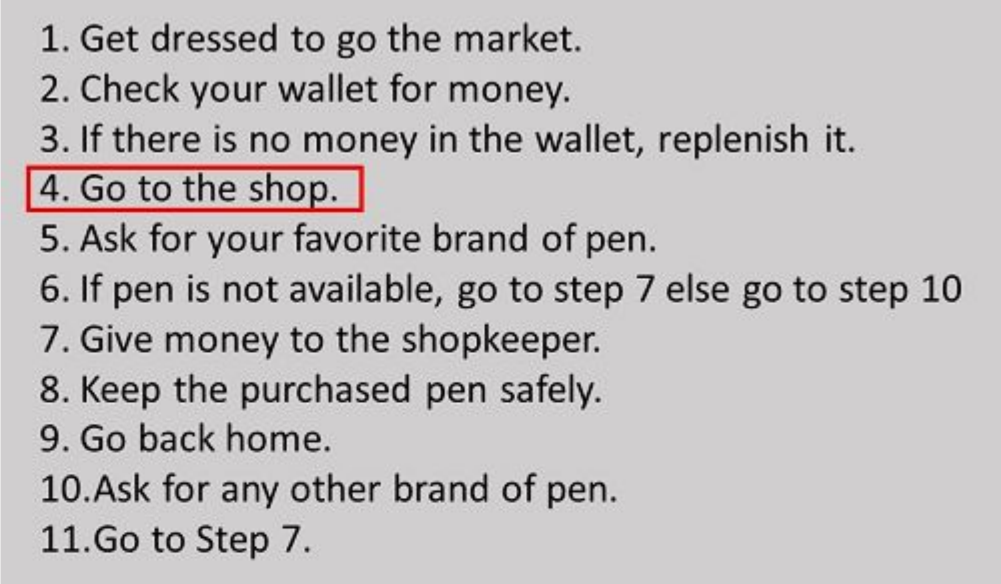
Symbol	Symbol Name	Purpose
⬭	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program.
▭	Process	Indicates processes like mathematical operations.
▱	Input/ Output	Used for denoting program inputs and outputs.
◇	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
↓	Arrow	Shows relationships between different shapes.
○	On-page Connector	Connects two or more parts of a flowchart, which are on the same page.
⬠	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

1. Flow Diagram - Basic Idea

If you want to purchase a pen by going to the market then write the necessary logical flow of steps. Also, draw the flowchart of the same process.

1. Flow Diagram - Basic Idea

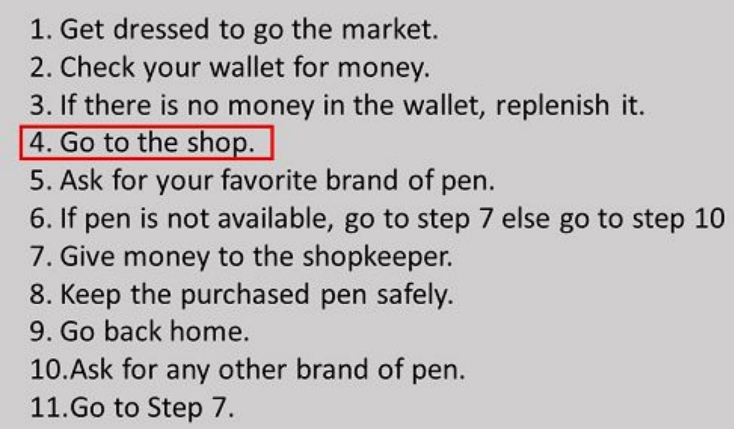
If you want to purchase a pen by going to the market then write the necessary logical flow of steps. Also, draw the flowchart of the same process.



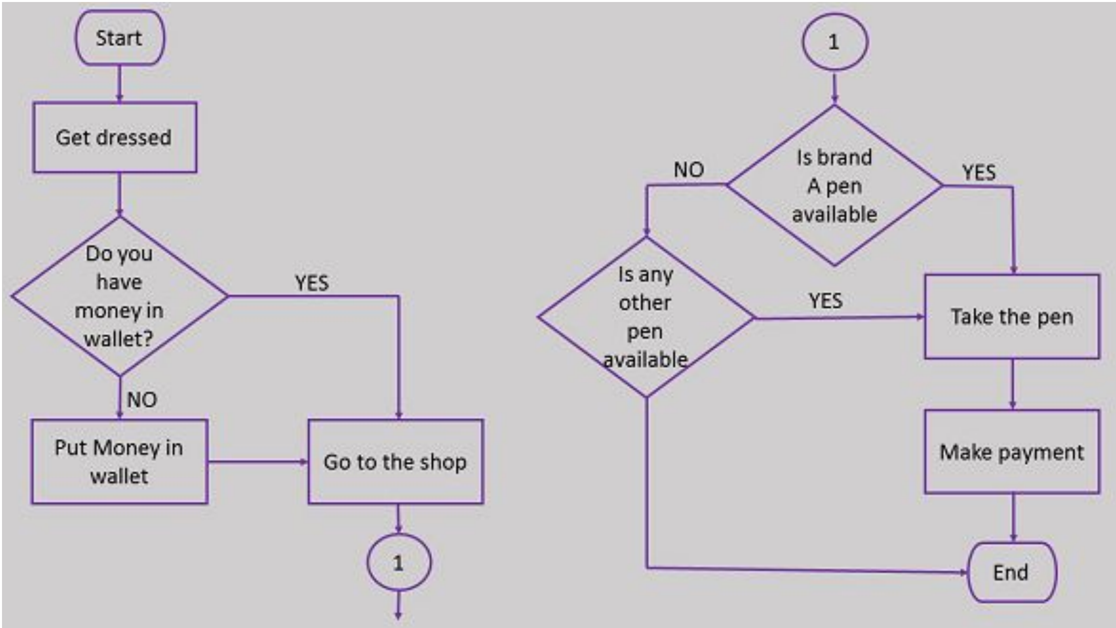
1. Get dressed to go the market.
2. Check your wallet for money.
3. If there is no money in the wallet, replenish it.
4. Go to the shop.
5. Ask for your favorite brand of pen.
6. If pen is not available, go to step 7 else go to step 10
7. Give money to the shopkeeper.
8. Keep the purchased pen safely.
9. Go back home.
10. Ask for any other brand of pen.
11. Go to Step 7.

1. Flow Diagram - Basic Idea

If you want to purchase a pen by going to the market then write the necessary logical flow of steps. Also, draw the flowchart of the same process.



1. Get dressed to go the market.
2. Check your wallet for money.
3. If there is no money in the wallet, replenish it.
4. Go to the shop.
5. Ask for your favorite brand of pen.
6. If pen is not available, go to step 7 else go to step 10
7. Give money to the shopkeeper.
8. Keep the purchased pen safely.
9. Go back home.
10. Ask for any other brand of pen.
11. Go to Step 7.

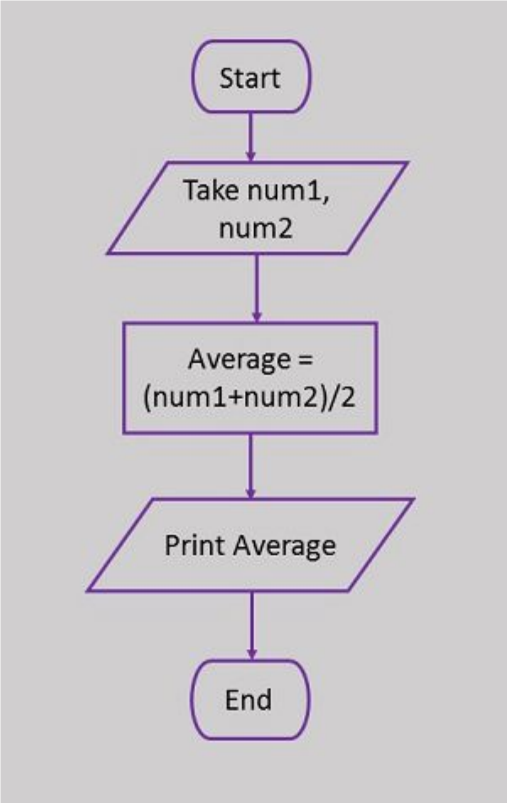


1. Flow Diagram - Problem 1

If you want to calculate the average of two numbers then write down the steps and draw the flow diagram. Write a C code for the same.

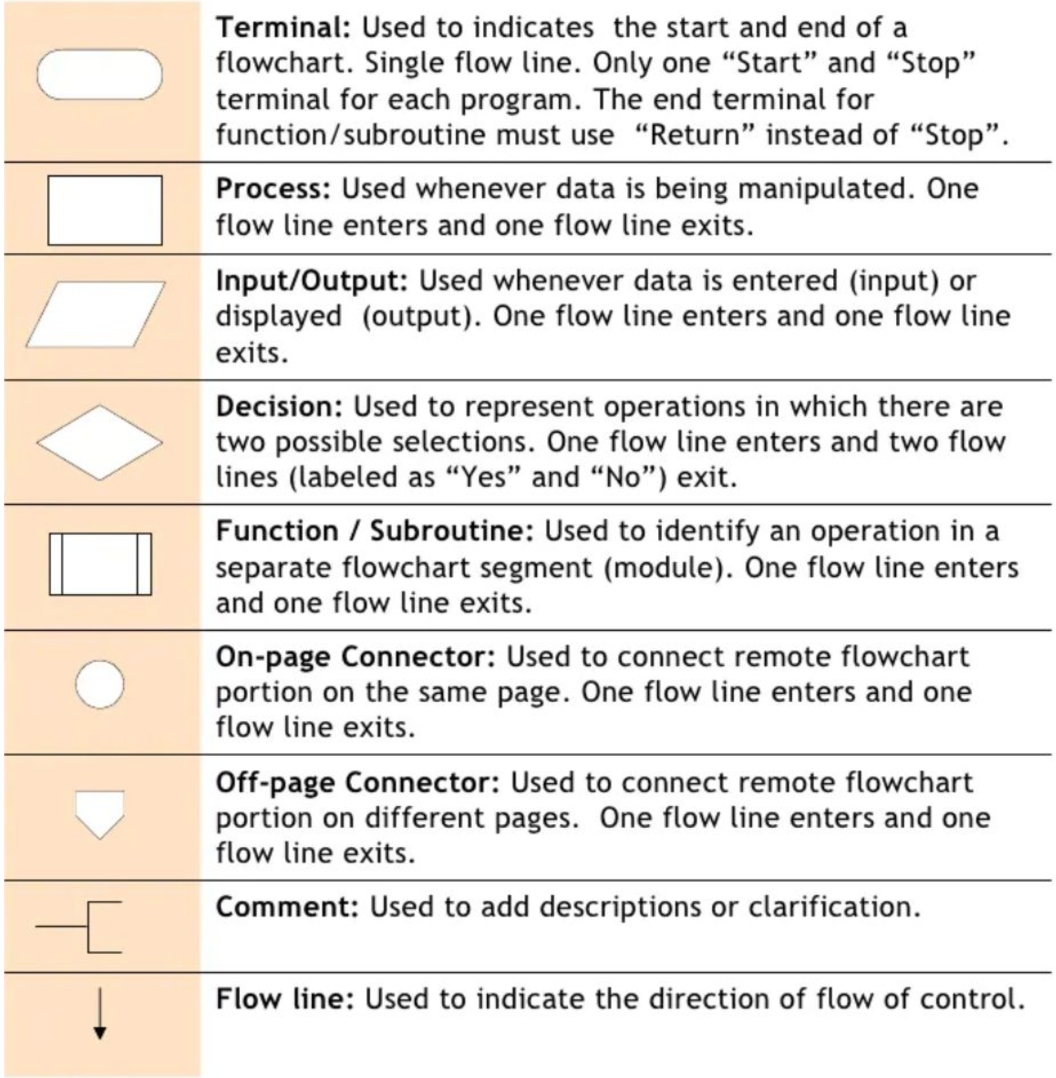
1. Flow Diagram - Problem 1

If you want to calculate the average of two numbers then write down the steps and draw the flow diagram. Write a C code for the same.



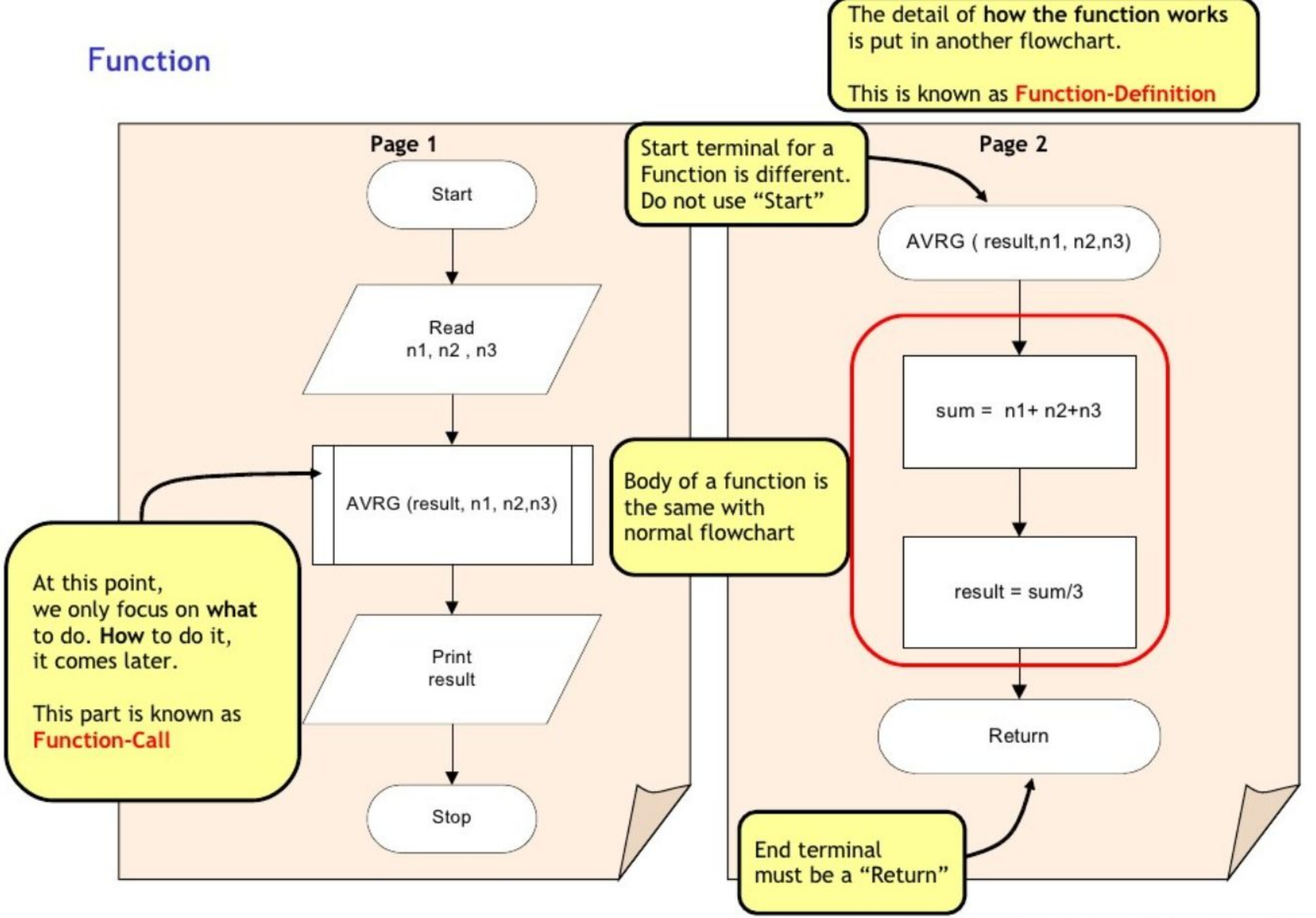
1. Flow Diagram - Problem 2

Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of five marks. Write a C++ code for the same.



⬭	**Terminal:** Used to indicates the start and end of a flowchart. Single flow line. Only one “Start” and “Stop” terminal for each program. The end terminal for function/subroutine must use “Return” instead of “Stop”.
▭	**Process:** Used whenever data is being manipulated. One flow line enters and one flow line exits.
▱	**Input/Output:** Used whenever data is entered (input) or displayed (output). One flow line enters and one flow line exits.
◇	**Decision:** Used to represent operations in which there are two possible selections. One flow line enters and two flow lines (labeled as “Yes” and “No”) exit.
◫	**Function / Subroutine:** Used to identify an operation in a separate flowchart segment (module). One flow line enters and one flow line exits.
○	**On-page Connector:** Used to connect remote flowchart portion on the same page. One flow line enters and one flow line exits.
⬠	**Off-page Connector:** Used to connect remote flowchart portion on different pages. One flow line enters and one flow line exits.
—[	**Comment:** Used to add descriptions or clarification.
↓	**Flow line:** Used to indicate the direction of flow of control.

Function



The detail of **how the function works** is put in another flowchart.

This is known as Function-Definition

Start terminal for a Function is different. Do not use “Start”

Body of a function is the same with normal flowchart

At this point, we only focus on **what** to do. **How** to do it, it comes later.

This part is known as Function-Call

End terminal must be a “Return”

This flowchart calculates the average of three numbers

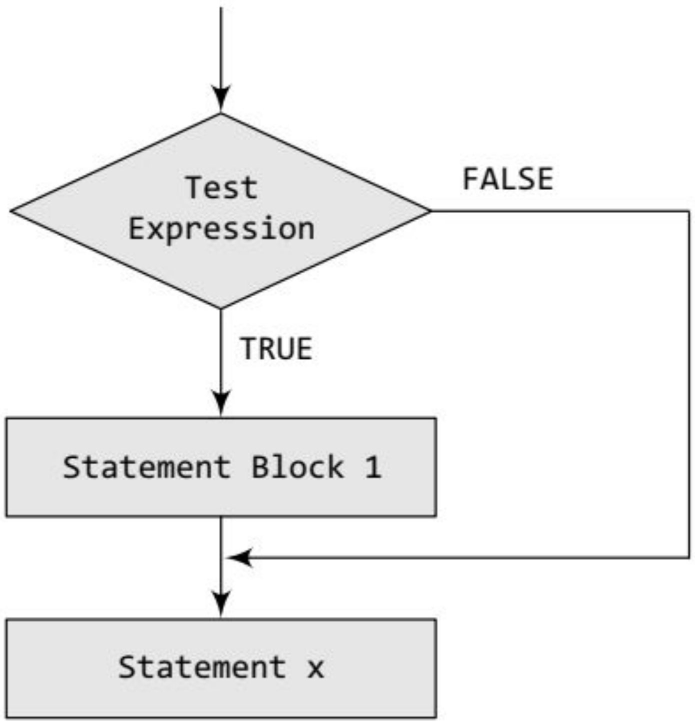
Reference:-

https://www.tutorialspoint.com/programming_methodologies/programming_methodologies_flowchart_elements.htm

2. Control Statement - If

Syntax of if Statement

```
if (test expression)
{
    statement 1;
    ............
    statement n;
}
statement x;
```



2. Control Statement - If

Write a C++ code for testing whether a number 10 is greater than 0.

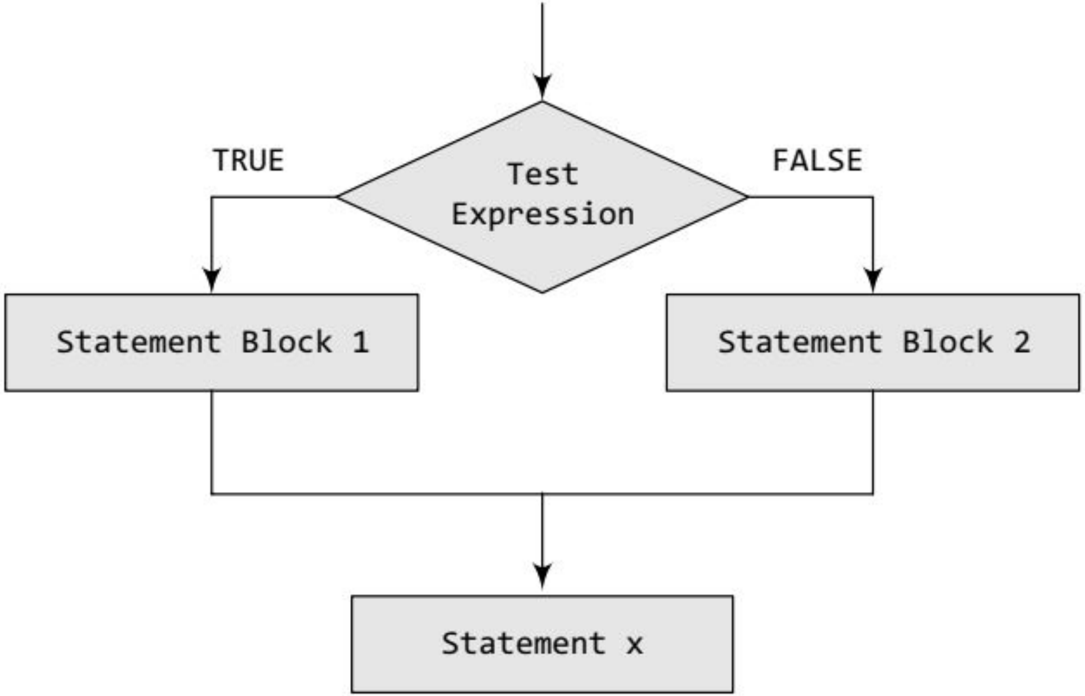
2. Control Statement - If

Write a C++ code for testing whether a number 10 is greater than 0.

2. Control Statement - If-Else

Syntax of if-else Statement

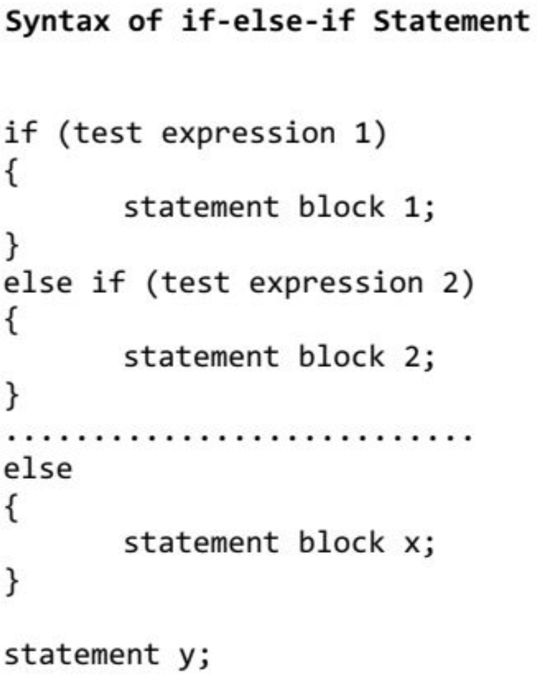
```
if (test expression)
{
    statement block 1;
}
else
{
    statement block 2;
}
statement x;
```



2. Control Statement - If-Else

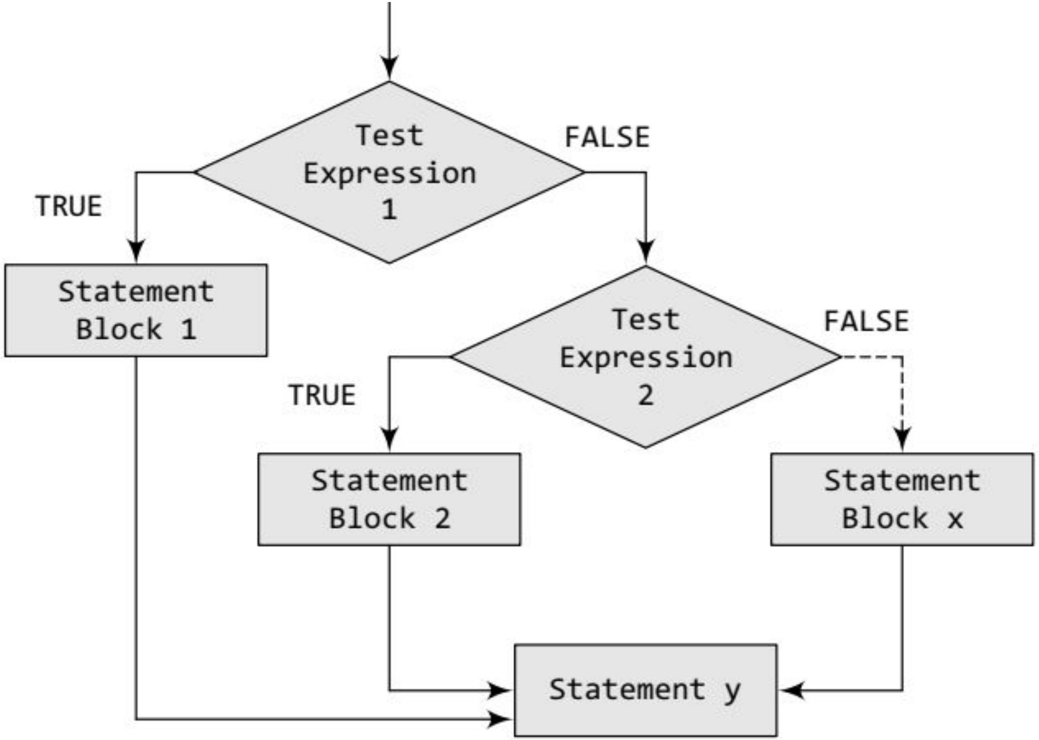
Write a C++ code for testing whether a number is odd or even.

2. Control Statement - If-Else If



Syntax of if-else-if Statement

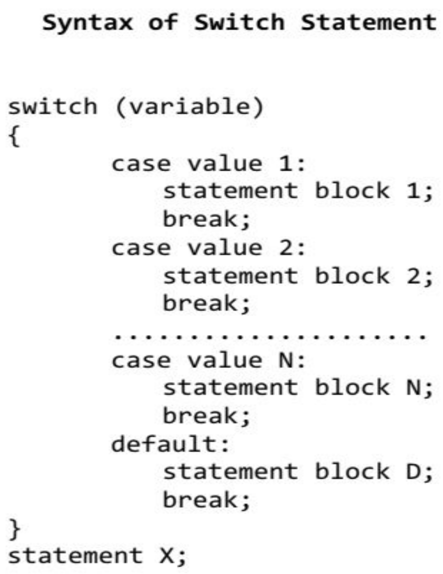
```
if (test expression 1)
{
        statement block 1;
}
else if (test expression 2)
{
        statement block 2;
}
..........................
else
{
        statement block x;
}

statement y;
```



2. Control Statement - If-Else If

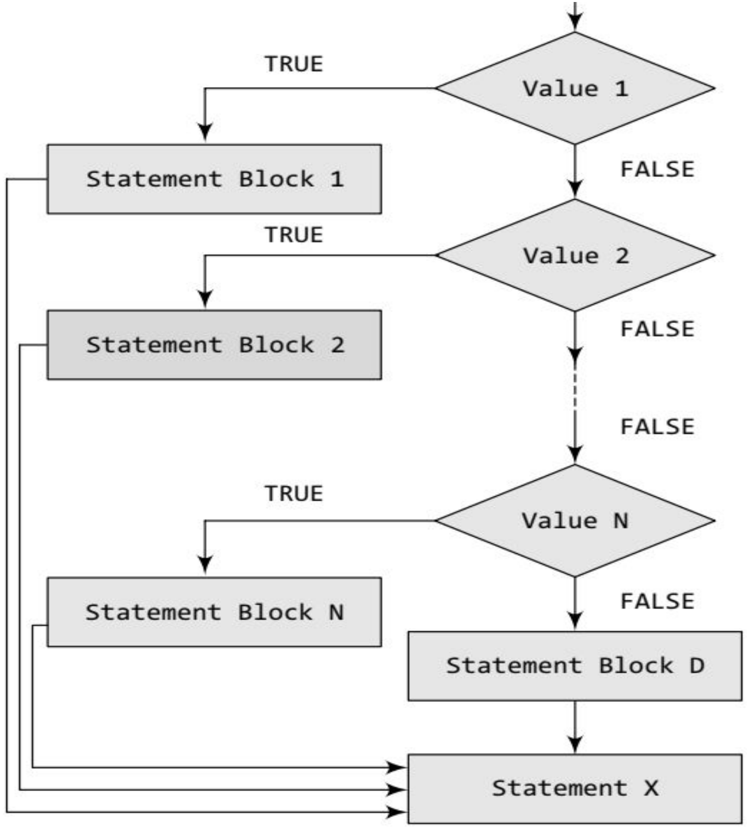
Write a C++ code for testing whether a number is positive, negative or zero

2. Control Statement - Switch-Case



Syntax of Switch Statement

```
switch (variable)
{
       case value 1:
          statement block 1;
          break;
       case value 2:
          statement block 2;
          break;
       .....................
       case value N:
          statement block N;
          break;
       default:
          statement block D;
          break;
}
statement X;
```

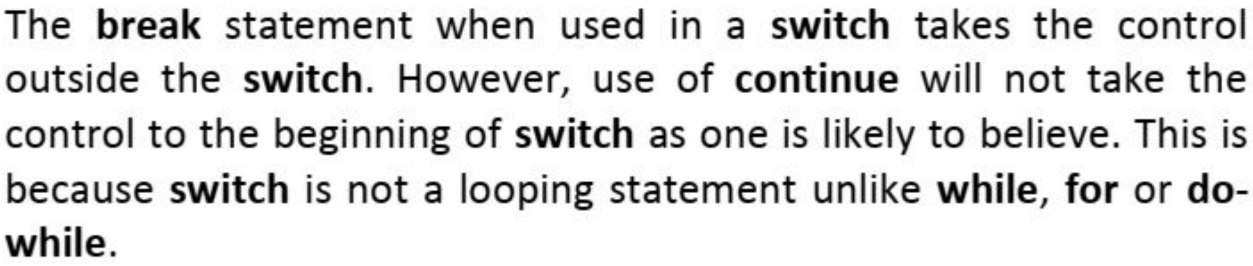


2. Control Statement - Switch-Case

- Write a C++ code for testing whether a number is one, two or three using switch-case.

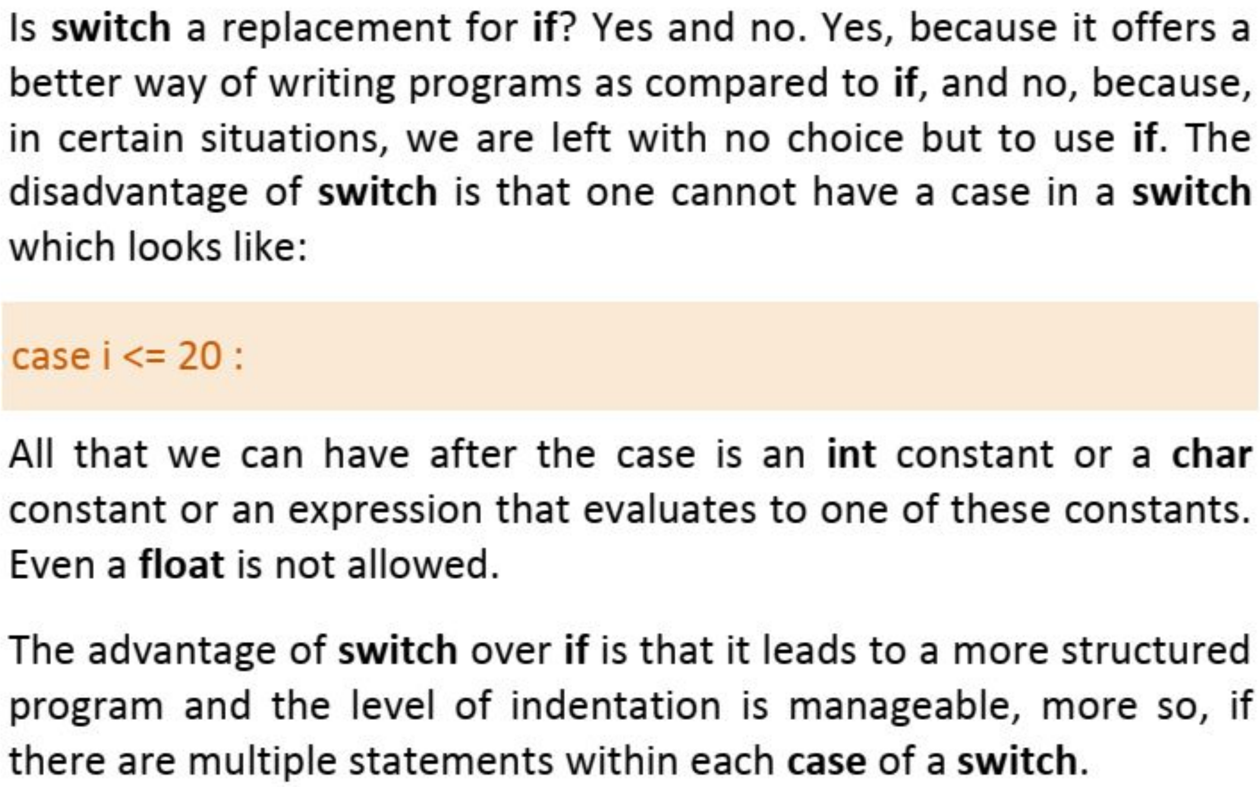
- Write a program to determine whether an entered character is a vowel or not.

2. Control Statement - Switch-Case



The **break** statement when used in a **switch** takes the control outside the **switch**. However, use of **continue** will not take the control to the beginning of **switch** as one is likely to believe. This is because **switch** is not a looping statement unlike **while**, **for** or **do-while**.

2. Control Statement -Switch-Case & If-Else If



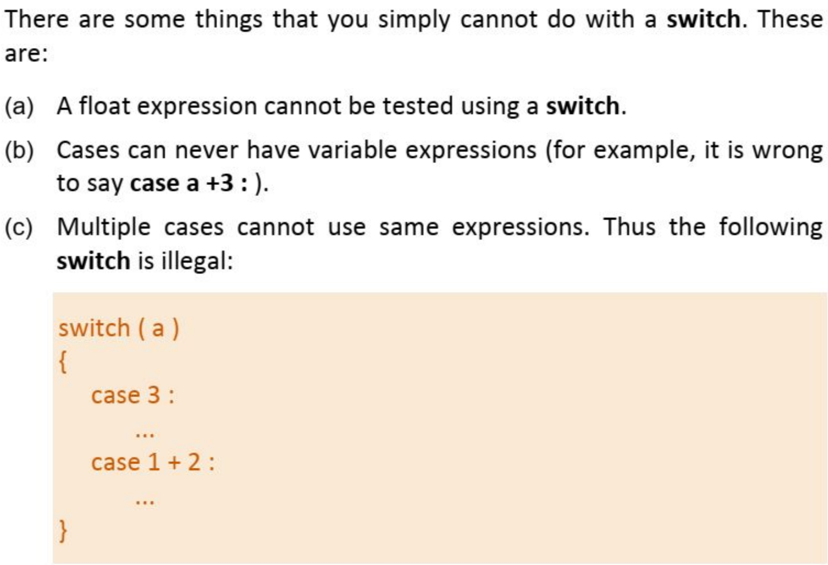
Is **switch** a replacement for **if**? Yes and no. Yes, because it offers a better way of writing programs as compared to **if**, and no, because, in certain situations, we are left with no choice but to use **if**. The disadvantage of **switch** is that one cannot have a case in a **switch** which looks like:

```
case i <= 20 :
```

All that we can have after the case is an **int** constant or a **char** constant or an expression that evaluates to one of these constants. Even a **float** is not allowed.

The advantage of **switch** over **if** is that it leads to a more structured program and the level of indentation is manageable, more so, if there are multiple statements within each **case** of a **switch**.

2. Control Statement -Switch-Case & If-Else If



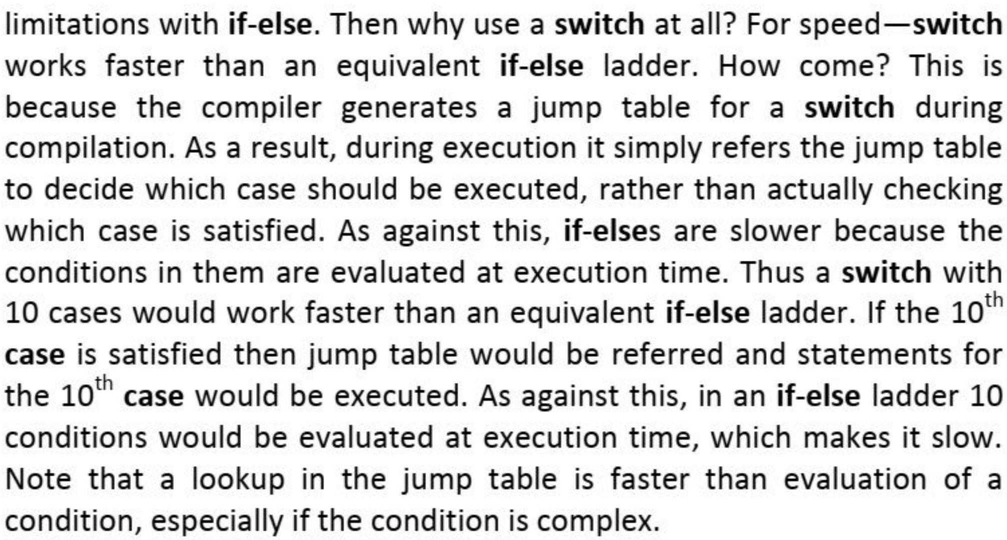
There are some things that you simply cannot do with a **switch**. These are:

(a) A float expression cannot be tested using a **switch**.

(b) Cases can never have variable expressions (for example, it is wrong to say **case a +3 :**).

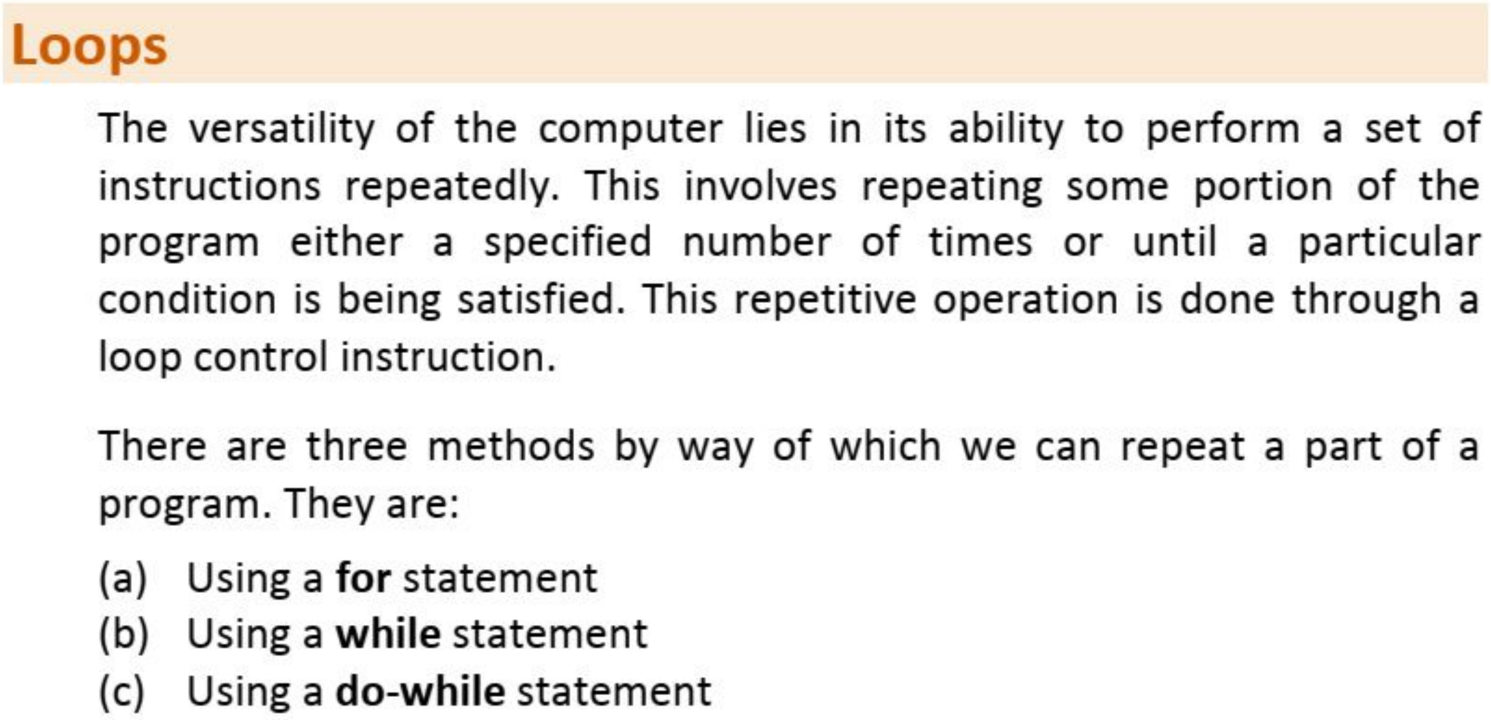
(c) Multiple cases cannot use same expressions. Thus the following **switch** is illegal:

```
switch ( a )
{
    case 3 :
        ...
    case 1 + 2 :
        ...
}
```



limitations with **if-else**. Then why use a **switch** at all? For speed—**switch** works faster than an equivalent **if-else** ladder. How come? This is because the compiler generates a jump table for a **switch** during compilation. As a result, during execution it simply refers the jump table to decide which case should be executed, rather than actually checking which case is satisfied. As against this, **if-else**s are slower because the conditions in them are evaluated at execution time. Thus a **switch** with 10 cases would work faster than an equivalent **if-else** ladder. If the 10th **case** is satisfied then jump table would be referred and statements for the 10th **case** would be executed. As against this, in an **if-else** ladder 10 conditions would be evaluated at execution time, which makes it slow. Note that a lookup in the jump table is faster than evaluation of a condition, especially if the condition is complex.

2. Control Statement - LooP



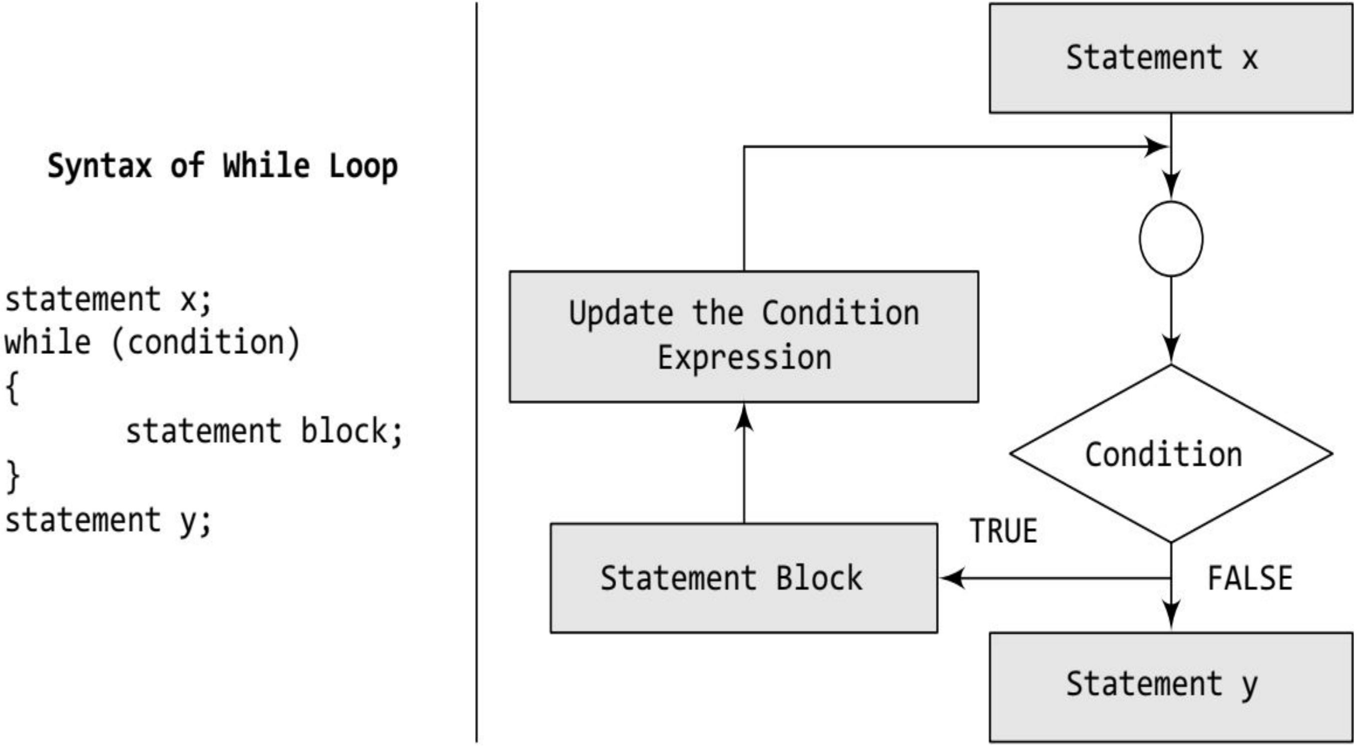
Loops

The versatility of the computer lies in its ability to perform a set of instructions repeatedly. This involves repeating some portion of the program either a specified number of times or until a particular condition is being satisfied. This repetitive operation is done through a loop control instruction.

There are three methods by way of which we can repeat a part of a program. They are:

(a) Using a **for** statement
(b) Using a **while** statement
(c) Using a **do-while** statement

2. Control Statement - While



Syntax of While Loop

```
statement x;
while (condition)
{
        statement block;
}
statement y;
```

2. Control Statement - While

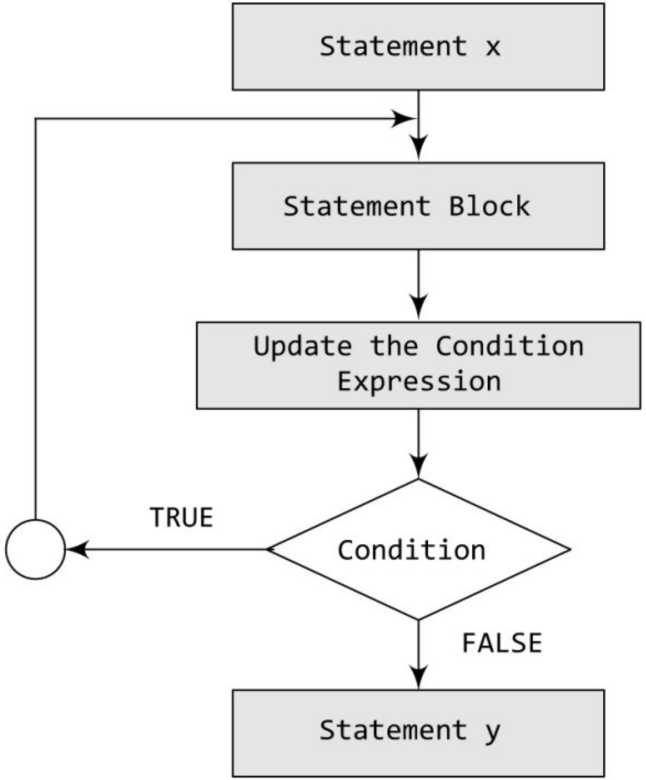
- Write a C++ code that can print numbers from 1 to 10.

- Write a program which can take the number from user and check the value entered is -1 or not and also you need to check the given number is >=or< than 0 or not using conditional statement. Do the same program using switch-case.

2. Control Statement - Do-While

Syntax of do-While Loop

```
statement x;
do
{
        statement block;
}       while (condition);

statement y;
```



2. Control Statement - Do-While

- Write a C++ code that can print numbers from 1 to 10.
- Write a C++ program that can calculate square and cube of the first n natural numbers.

2. Control Statement - While & Do-While

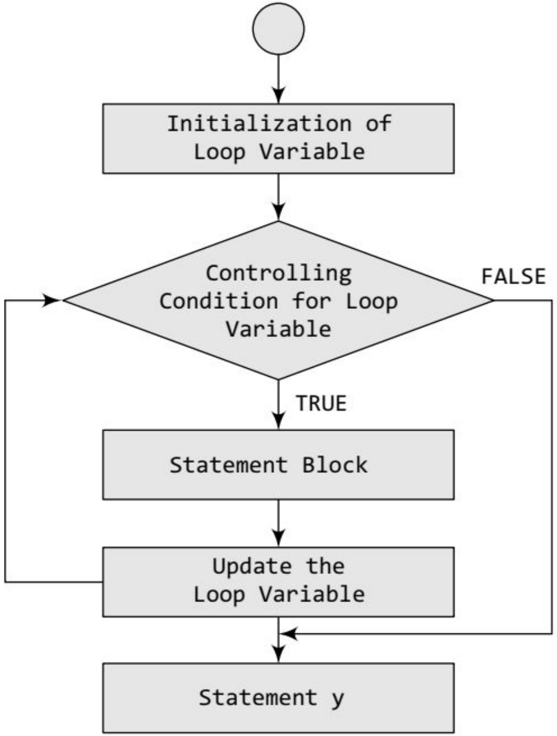
Write a C++ code that can print numbers from 1 to 10.

2. Control Statement - For

Write a C++ code that can print first n numbers.

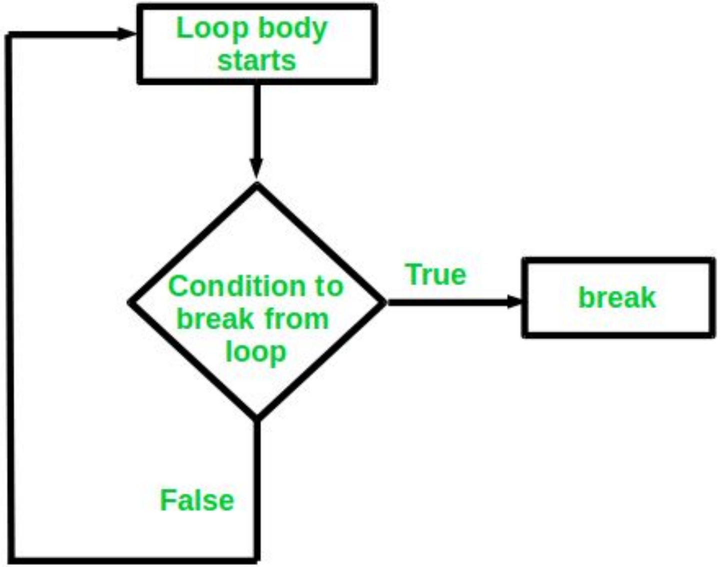
Syntax of for Loop

```
for (initialization; condition;
increment/decrement/update)
{
     statement block;
}
statement y;
```



Initialization of Loop Variable
Controlling Condition for Loop Variable
FALSE
TRUE
Statement Block
Update the Loop Variable
Statement y

2. Control Statement - Break

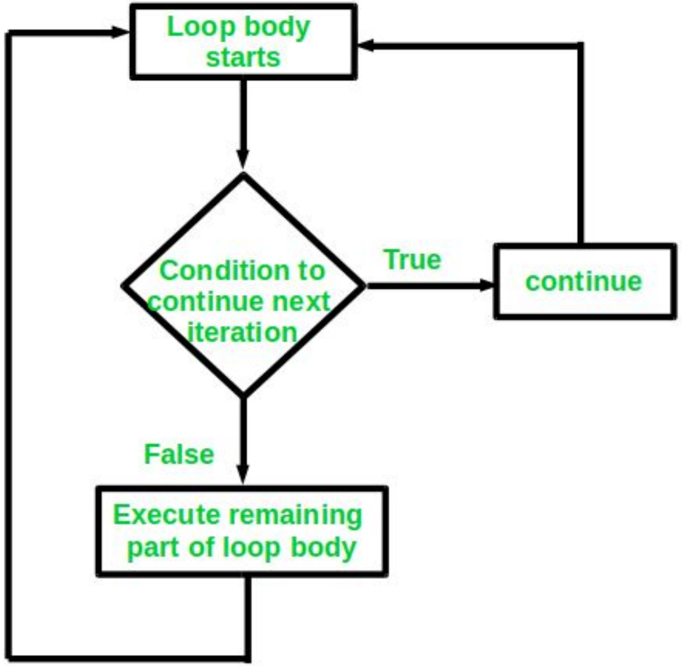


Break statements are used in situations when we are not sure about the actual number of iterations for the loop or we want to terminate the loop based on some condition. We will see here the usage of break statements with three different types of loops:

1. Simple loops
2. Nested loops
3. Infinite loops

Ref:- https://www.geeksforgeeks.org/cpp-break-statement/

2. Control Statement - Continue



Loop body starts

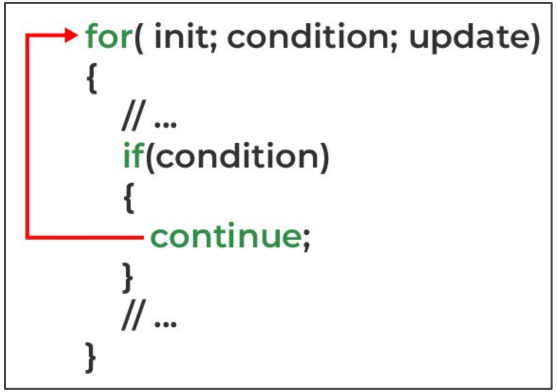
Condition to continue next iteration

True

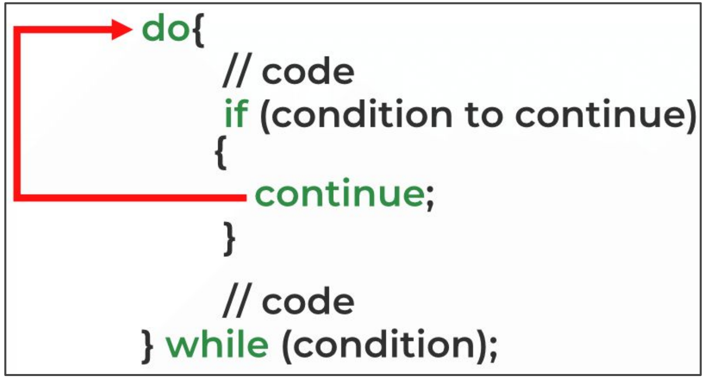
continue

False

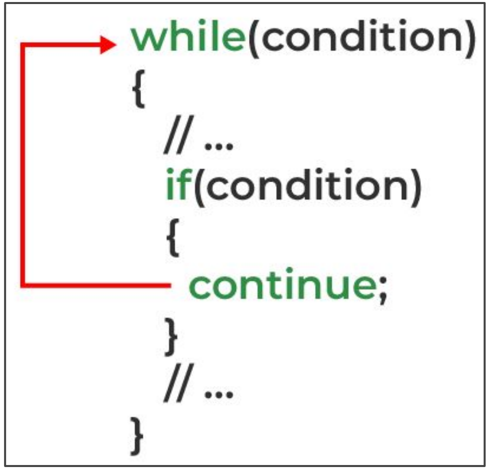
Execute remaining part of loop body



```
for( init; condition; update)
{
    // ...
    if(condition)
    {
     continue;
    }
    // ...
}
```



```
do{
        // code
        if (condition to continue)
        {
         continue;
        }

        // code
} while (condition);
```



```
while(condition)
{
    // ...
    if(condition)
    {
     continue;
    }
    // ...
}
```

Ref:- https://www.geeksforgeeks.org/cpp-break-statement/

Problem:-

Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of five marks. Write a C++ code for the same.

Using if-else, switch-case

Problem:-

- Write a C++ program to enter a number and then calculate the sum of the digits.

- Write a C++ program that can print the reverse of a number.

- Write a C++ program that can calculate the sum of cubes of numbers from 1-n.

- Write a C++ program that can calculate the sum of squares of odd numbers.