GLS UNIVERSITY

CONDITIONAL STATEMENTS, LOOPING
UNIT – II

Simple IF

In C++ if statement is used to check the truth of the expression.

Condition written inside If statement consists of a Boolean expression.

If the condition written inside if is true then if block gets executed.

Syntax :

If statement is having following syntax –

```
if(boolean_expression)
{
   // statement(s) will execute if the
         // boolean expression is true
}
```

Example : C++ If Statement

Note #1 : Curly braces

Complete if block is written inside the pair of curly braces. If we have single statement inside the if block then we can skip the pair of curly braces –

```
if( num > 4 )
    cout << "num is greater than 4" << endl;
```

Example : C++ If else Statement

If block sometimes followed with the optional else block. When the if condition becomes false then else block gets executed.

Syntax of C++ If-else :

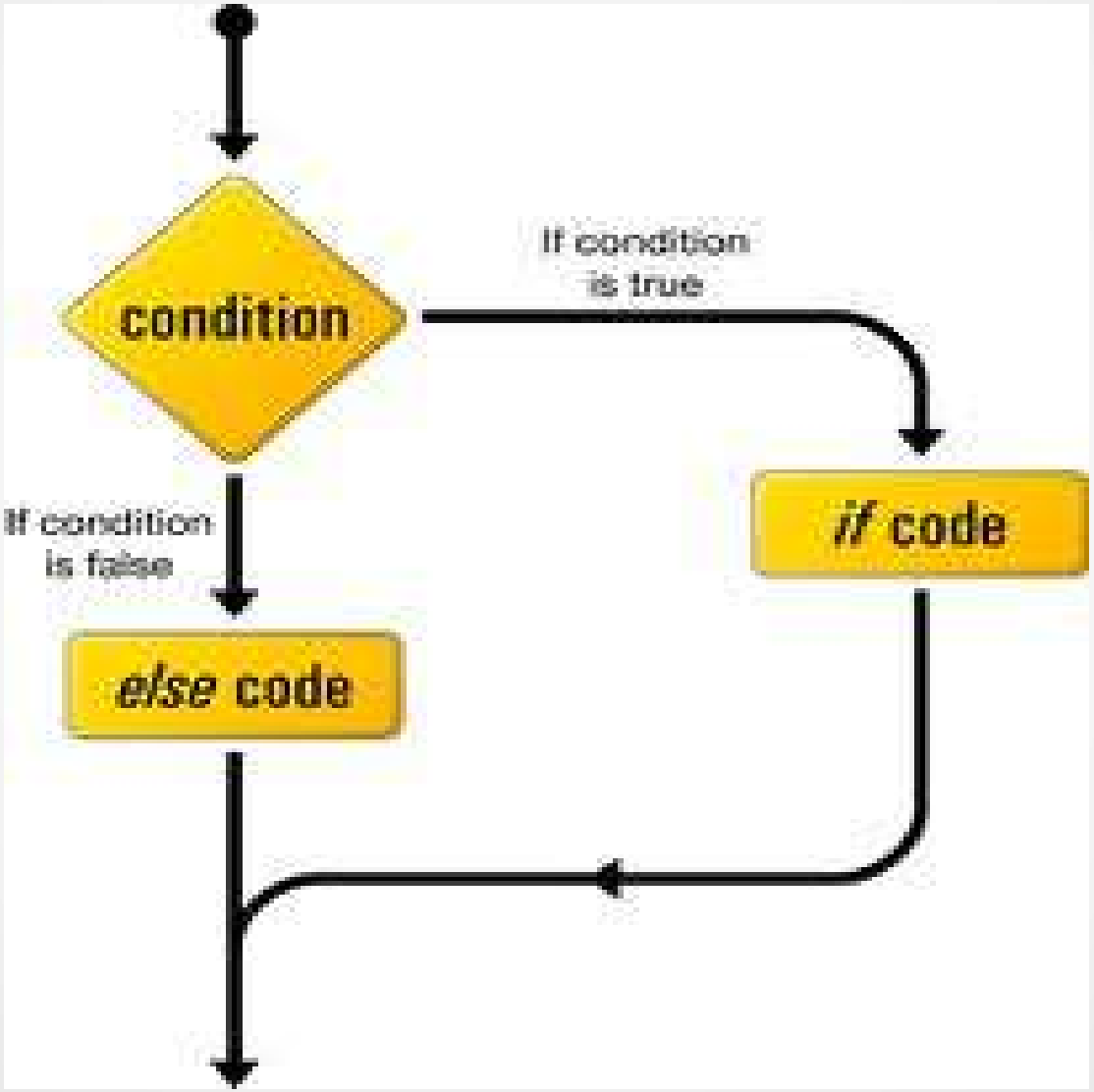
```
if(boolean_expression)
{
   // True Block
}
else
{
   // False Block
}
```

Example : C++ If Statement

```
#include <iostream>
using namespace std;
int main ()
{
   // Declaring local variable
   int num = 5;// check the boolean condition
   if( num > 4 )
   {
      cout << "num is greater than 4" << endl;
   }
   cout << "Given number is : " << num << endl;
   return 0;
}
```

Example : C++ If else Statement

If the Boolean_expression becomes false then then else block will be executed. If the Boolean expression becomes true then by default true block will be executed.


condition
If condition is true
if code
If condition is false
else code

```
#include <iostream>
using namespace std;
 int main ()
{
   // declare local variable
   int marks = 30;// check the
boolean condition
   if( marks > 40 )
   {
       // if condition is true
  cout << "You are pass !!" << endl;
   }
   else
   {
       // if condition is false
 cout << "You are fail !!" << endl;
   }
 return 0;
}
```

Example : C++ If else Ladder Statement

When we need to list multiple else if…else statement blocks. We can check the various conditions using the following syntax –

Syntax : C++ Else if ladder

```
if(boolean_expression 1)
{
   // When expression 1 is true
}
else if( boolean_expression 2)
{
   // When expression 2 is true
}
else if( boolean_expression 3)
{
   // When expression 3 is true
}
else
{
   // When none of expression is true
}
```

Example : C++ If else Ladder Statement

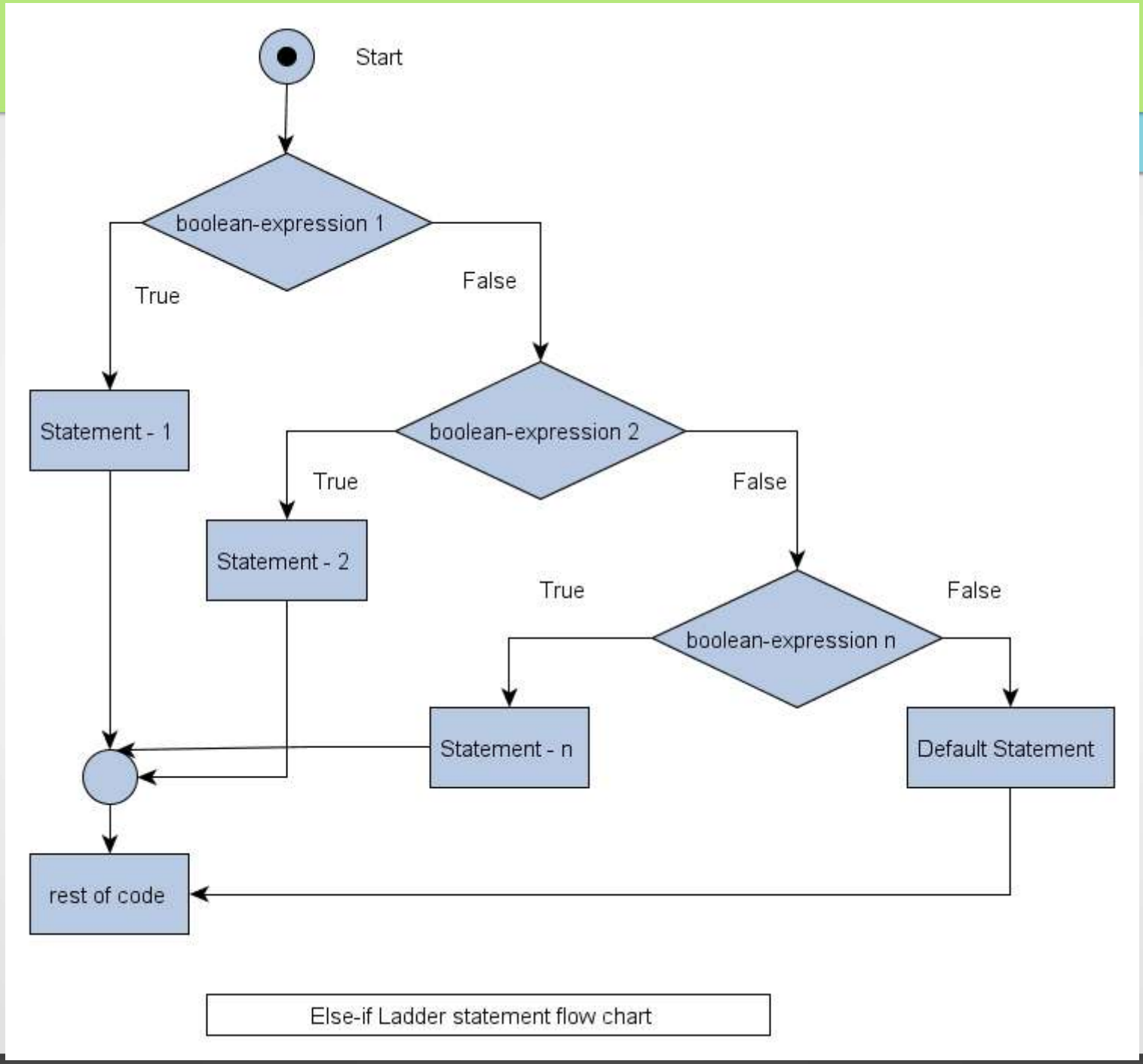
When the first Boolean condition becomes true then first block gets executed.

When first condition becomes false then second block is checked against the condition. If second condition becomes true then second block gets executed.

Else IF

When we need to list multiple else if…else statement blocks. We can check the various conditions using the following syntax –

```
#include <iostream>
using namespace std;
int main ()
{
  // declare local variable
  int marks = 55;
  // check the boolean condition
  if( marks >= 80 )
  {
     // if 1st condition is true
     cout << "U are 1st class !!" << endl;
  }
  else if( marks >= 60 && marks < 80)
  {
     // if 2nd condition is false
     cout << "U are 2nd class !!" << endl;
  }
  else if( marks >= 40 && marks < 60)
  {
     // if 3rd condition is false
     cout << "U are 3rd class !!" << endl;
  }
  else
  {
     // none of condition is true
     cout << "U are fail !!" << endl;
  }
  return 0;
}
```



Else-if Ladder statement flow chart

Some rules of using else if ladder in C++ :

Keep some important things in mind –

if block can have zero or one else block and it must come after else-if blocks.

if block can have zero to many else-if blocks and they must come before the else.

After the execution of any else-if block none of the remaining else-if block condition is checked

Nested If

It is always legal to nest if-else statements, which means you can use one if or else if statement inside another if or else if statement(s).

Syntax:

The syntax for a nested if statement is as follows:

```
if( boolean_expression 1)
{
   // Executes when the boolean expression
1 is true
   if(boolean_expression 2)
   {
      // Executes when the boolean
expression 2 is true
   }
}
```

You can nest else if...else in the similar way as you have nested if statement.

Example:

```
#include <iostream>
using namespace std;

int main ()
{
   // local variable declaration:
   int a = 100;
   int b = 200;

   // check the boolean condition
   if( a == 100 )
   {
       // if condition is true then check the following
       if( b == 200 )
       {
          // if condition is true then print the following
          cout << "Value of a is 100 and b is 200" << endl;
       }
   }
   cout << "Exact value of a is : " << a << endl;
   cout << "Exact value of b is : " << b << endl;

   return 0;
}
```

goto statement

- A goto statement provides an unconditional jump from the goto to a labeled statement in the same function.

- **Syntax**:

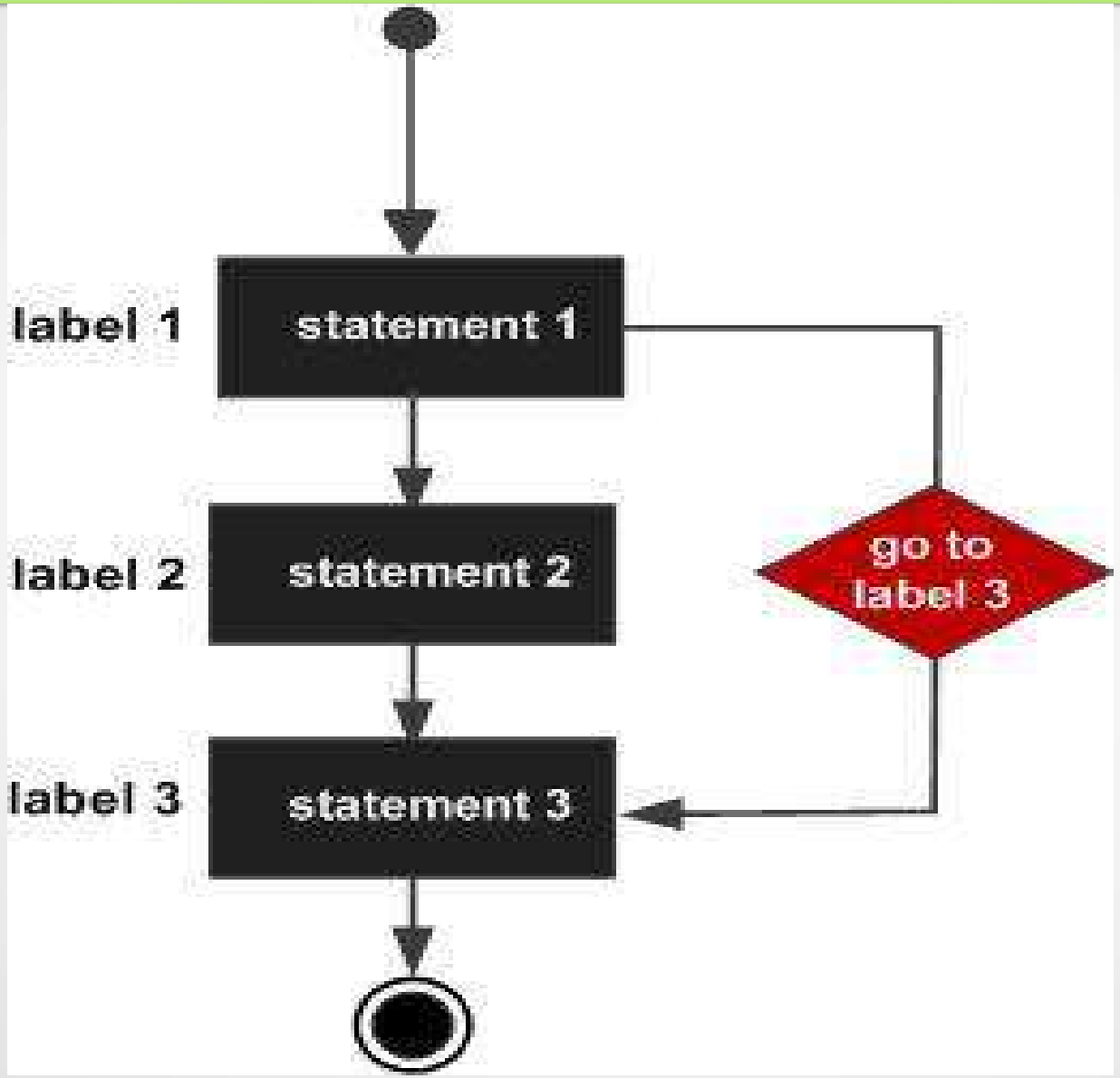
goto label;

..

.

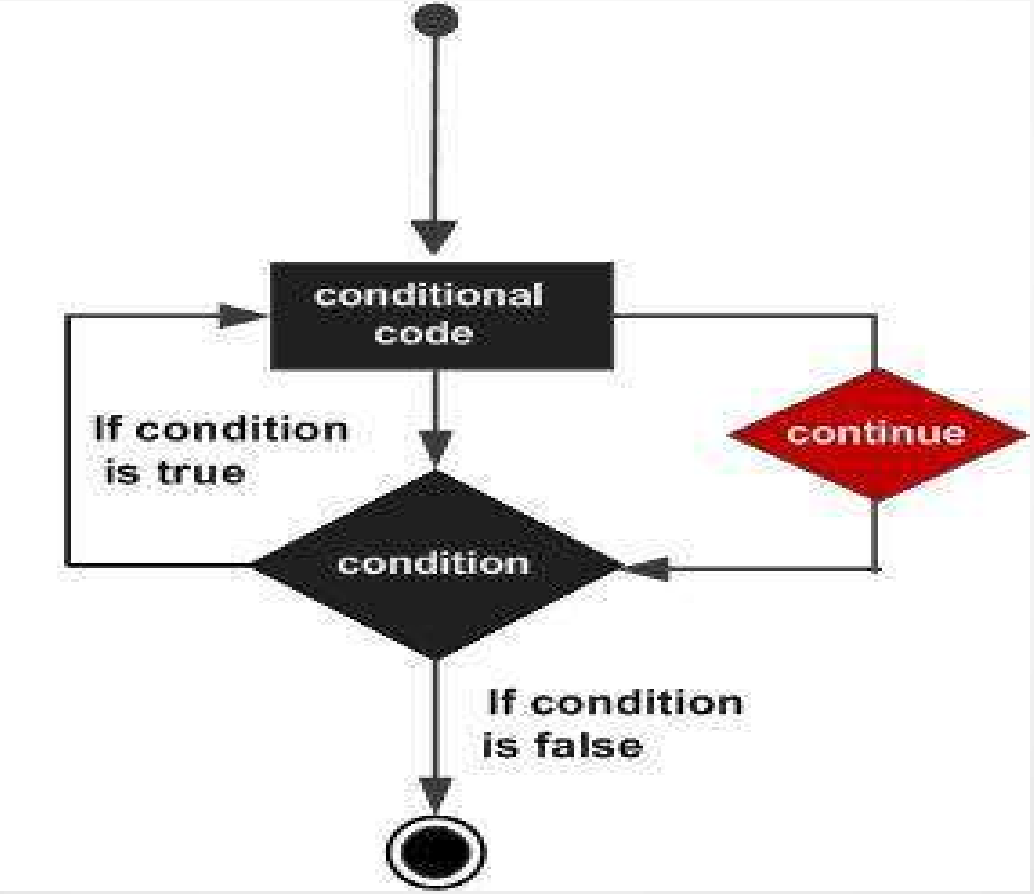
label: statement;

- Where label is an identifier that identifies a labeled statement. A labeled statement is any statement that is preceded by an identifier followed by a colon (:).


label 1
statement 1
label 2
statement 2
go to
label 3
label 3
statement 3

continue statement

- The continue statement works somewhat like the break statement. Instead of forcing termination, however, continue forces the next iteration of the loop to take place, skipping any code in between.

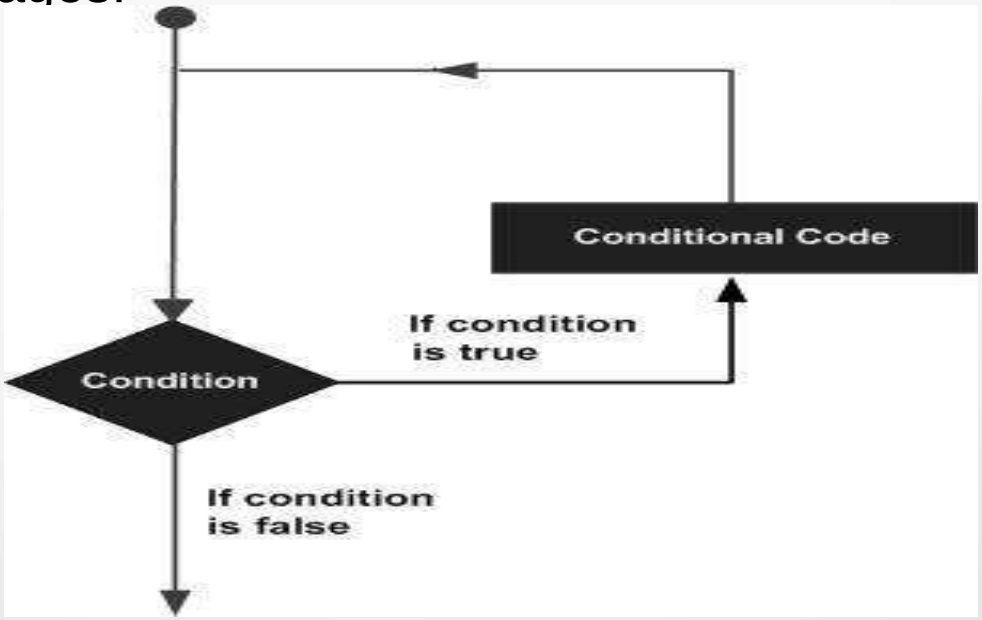


Looping

- There may be a situation, when you need to execute a block of code several number of times. In general statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

- Programming languages provide various control structures that allow for more complicated execution paths.

- A loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in most of the programming languages:

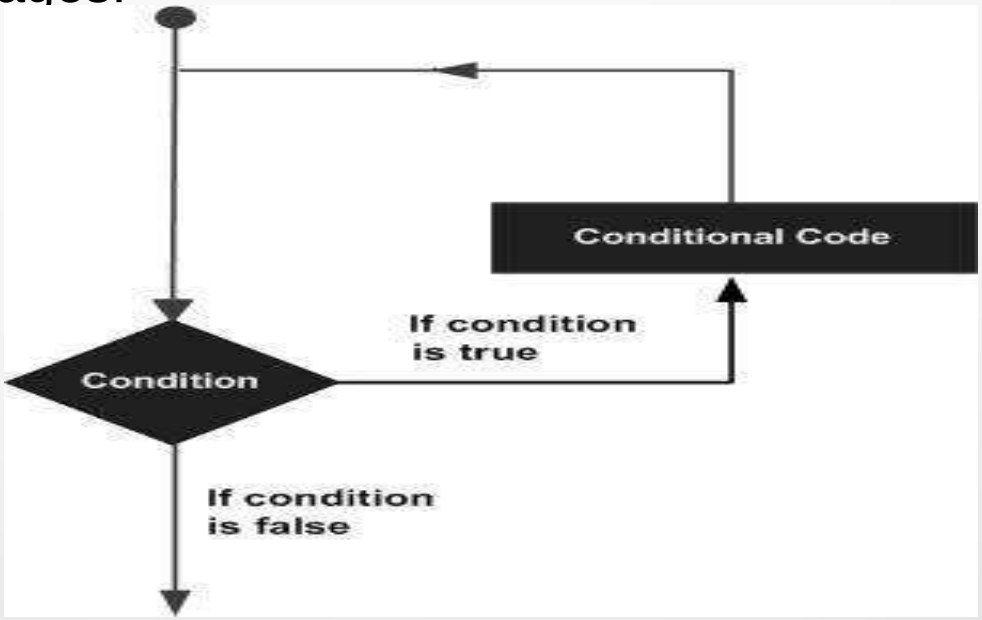


Looping

- There may be a situation, when you need to execute a block of code several number of times. In general statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on.

- Programming languages provide various control structures that allow for more complicated execution paths.

- A loop statement allows us to execute a statement or group of statements multiple times and following is the general from of a loop statement in most of the programming languages:



Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop	Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop	Like a while statement, except that it tests the condition at the end of the loop body
nested loops	You can use one or more loop inside any another while, for or do..while loop.

while loop

- A while loop statement repeatedly executes a target statement as long as a given condition is true.

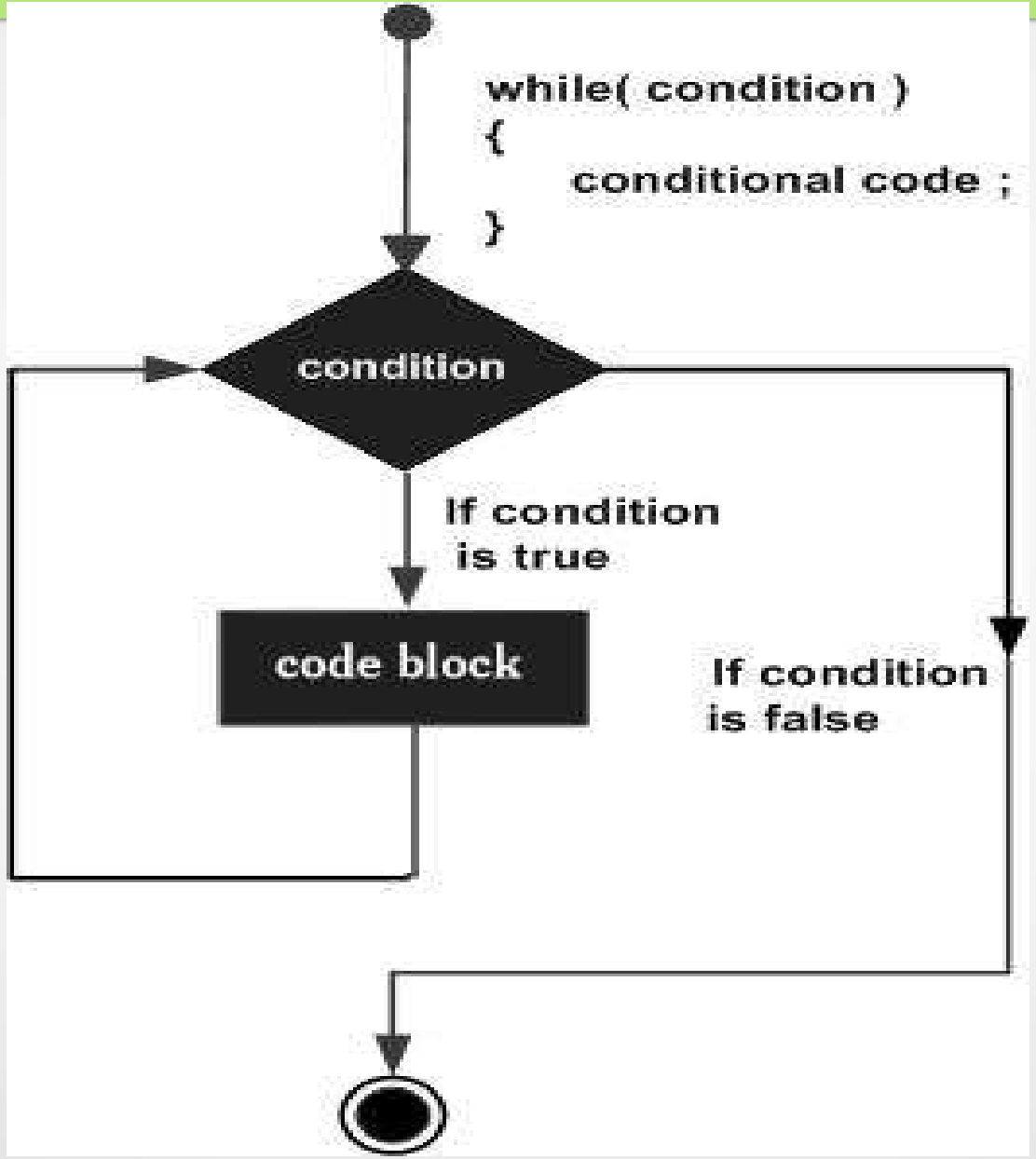
- Syntax:

```
while(condition)

{

   statement(s);

}
```

- statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

- When the condition becomes false, program control passes to the line immediately following the loop.

- When the condition is tested and the result is false, the loop body will be skipped and the first statement after the while loop will be executed.


while(condition)
{
conditional code ;
}
condition
If condition
is true
code block
If condition
is false

do..while loop

- Unlike for and while loops, which test the loop condition at the top of the loop, the do...while loop checks its condition at the bottom of the loop.

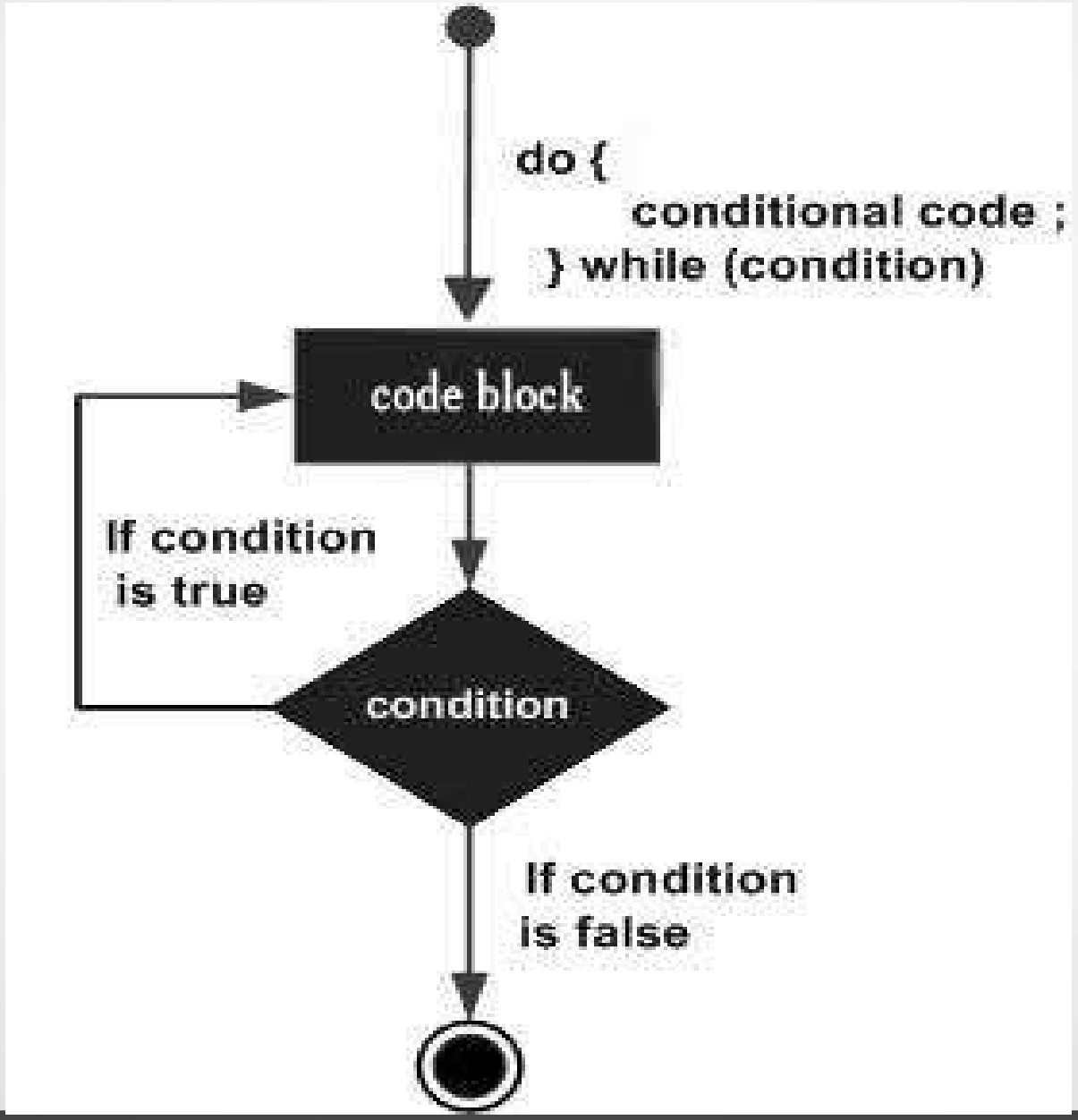
- A do...while loop is similar to a while loop, except that a do...while loop is guaranteed to execute at least one time.

- Syntax:

```
do

{

   statement(s);

}while( condition );
```

- Notice that the conditional expression appears at the end of the loop, so the statement(s) in the loop execute once before the condition is tested.

- If the condition is true, the flow of control jumps back up to do, and the statement(s) in the loop execute again. This process repeats until the given condition becomes false.


do {
conditional code ;
} while (condition)
code block
If condition
is true
condition
If condition
is false

for loop

- A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.
- Syntax:

```
for ( init; condition; increment )
{
   statement(s);
}
```

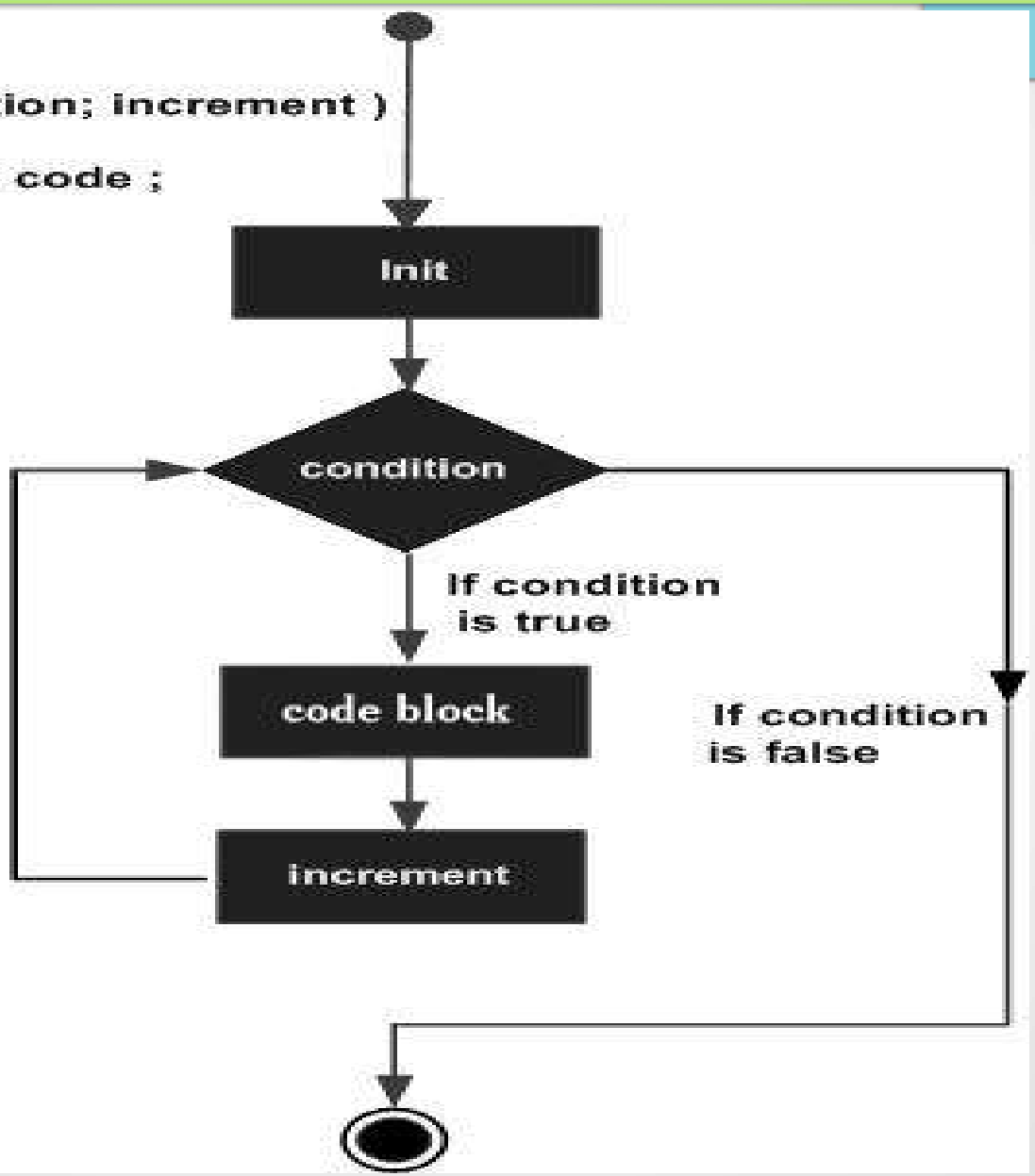
The init step is executed first, and only once. This step allows you to declare and initialize any loop control variables. You are not required to put a statement here, as long as a semicolon appears.

Next, the condition is evaluated. If it is true, the body of the loop is executed. If it is false, the body of the loop does not execute and flow of control jumps to the next statement just after the for loop.

After the body of the for loop executes, the flow of control jumps back up to the increment statement. This statement allows you to update any loop control variables. This statement can be left blank, as long as a semicolon appears after the condition.

The condition is now evaluated again. If it is true, the loop executes and the process repeats itself (body of loop, then increment step, and then again condition). After the condition becomes false, the for loop terminates.

```
for( init; condition; increment )
{
    conditional code ;
}
```



nested loop

- A loop can be nested inside of another loop.

The syntax for a nested for loop statement in C++

```
for ( init; condition; increment )
{
   for ( init; condition; increment )
   {
      statement(s);
   }
   statement(s); // you can put more statements.
}
```

- The syntax for a nested while loop statement:

```
while(condition)
{
   while(condition)
   {
      statement(s);
   }
   statement(s); // you can put more statements.
}
```

- The syntax for a nested do...while loop statement

```
do
{
   statement(s); // you can put more statements.
   do
   {
      statement(s);
   }while( condition );

}while( condition );
```