SLIIT ACADEMY

Higher Diploma in Information Technology
Year 1, Semester 1

Introduction to Programming(C++)

Lecture 05 : Repetition Statements in C++

Intended Learning Outcomes

On the Completion of this lecture student will be able to learn ,

LO1 : Learn about repetition (looping) control structures

LO2: Explore how to construct and use count-controlled, sentinel-controlled, flag-controlled, and EOF-controlled repetition structures

LO3: Examine break and continue statements

LO4: Discover how to form and use nested control structures

Repetition Statements in C++

- Some programs require the same logic to be repeated for several sets of data.
- The most efficient way to deal with this situation is to establish a looping structure that will cause the processing logic to be repeated a number of times.

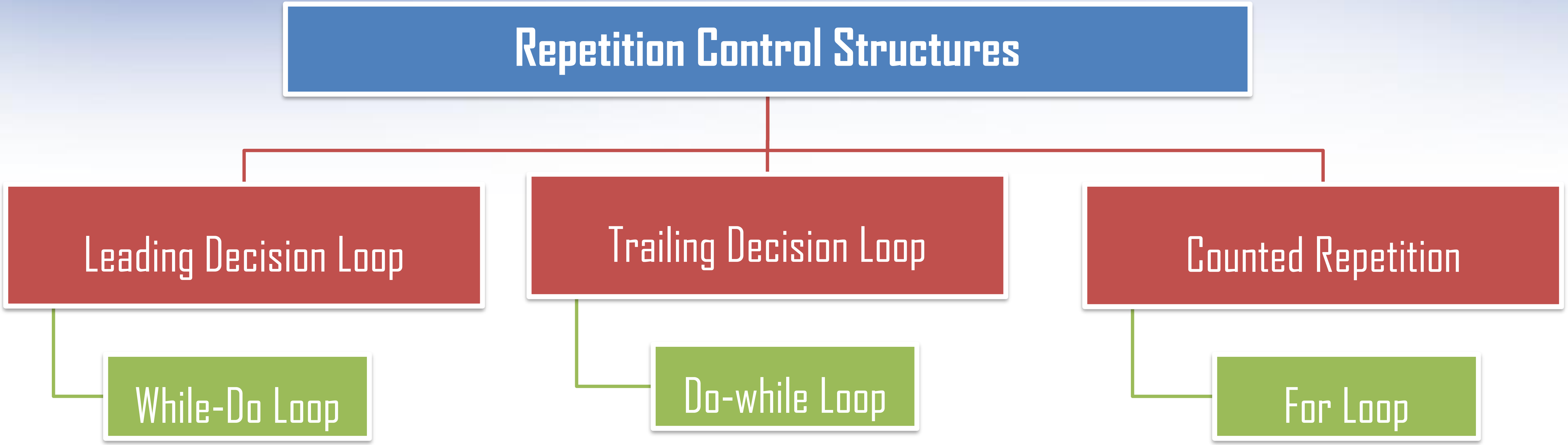
SLIIT ACADEMY

Repetition Statements in C++

- There are three different ways in which a set of instructions can be repeated, and each way is determined by where the decision to repeat is placed:
    - **At the beginning of the loop (leading decision loop)**
    - **At the end of the loop (trailing decision loop)**
    - **A counted number of times (counted loop).**

SLIIT ACADEMY

Repetition Control Structures in C++



Pre-test and Post-test loops

- Pre-test loops are entrance-controlled loops.
  - You execute the loop body after evaluating the test.
  - Loop body can be executed zero or more times.
- Post-test loops are exit controlled loops.
  - You test the loop after executing the entire loop body.
  - Loop body can be executed one or more times.

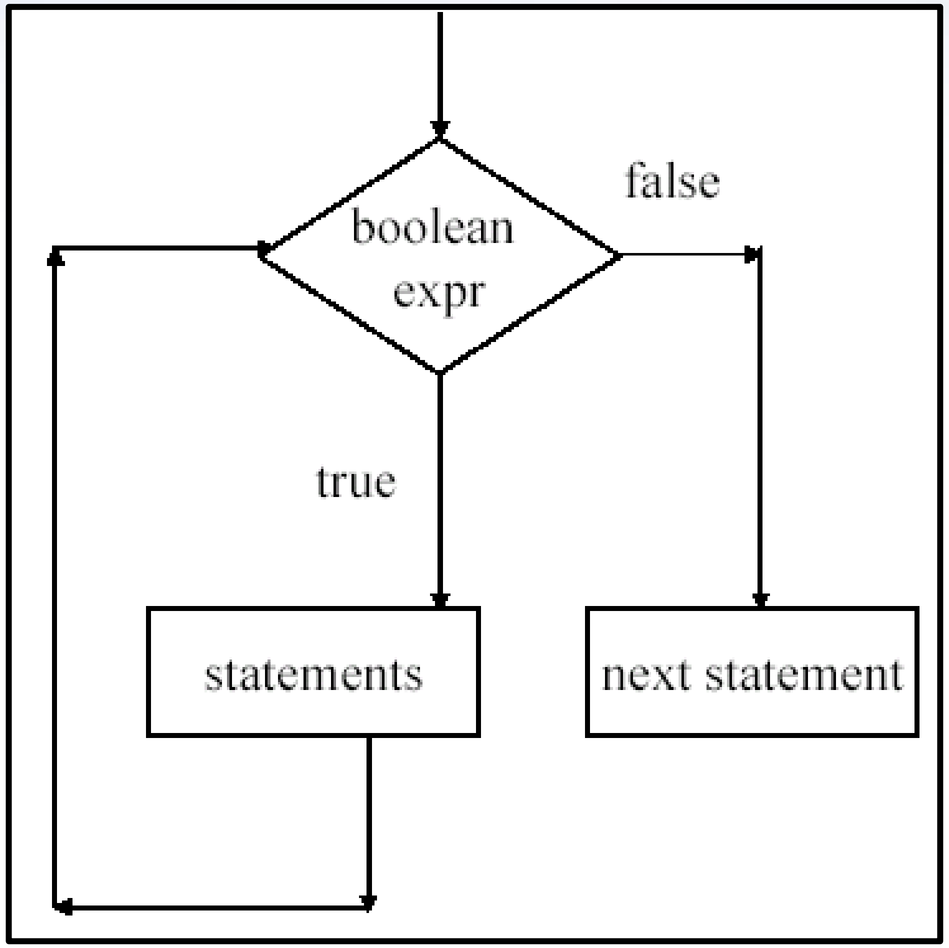
SLIIT ACADEMY

Pre-Test Loops : while structure

Syntax of While Loop

```
while (boolean expr)
     statement;
next statement;


while (boolean expr)
{
     statement 1;
     statement 2;
}
next statement;
```

Execution Flow Diagram



Notes on semantics of While Loop

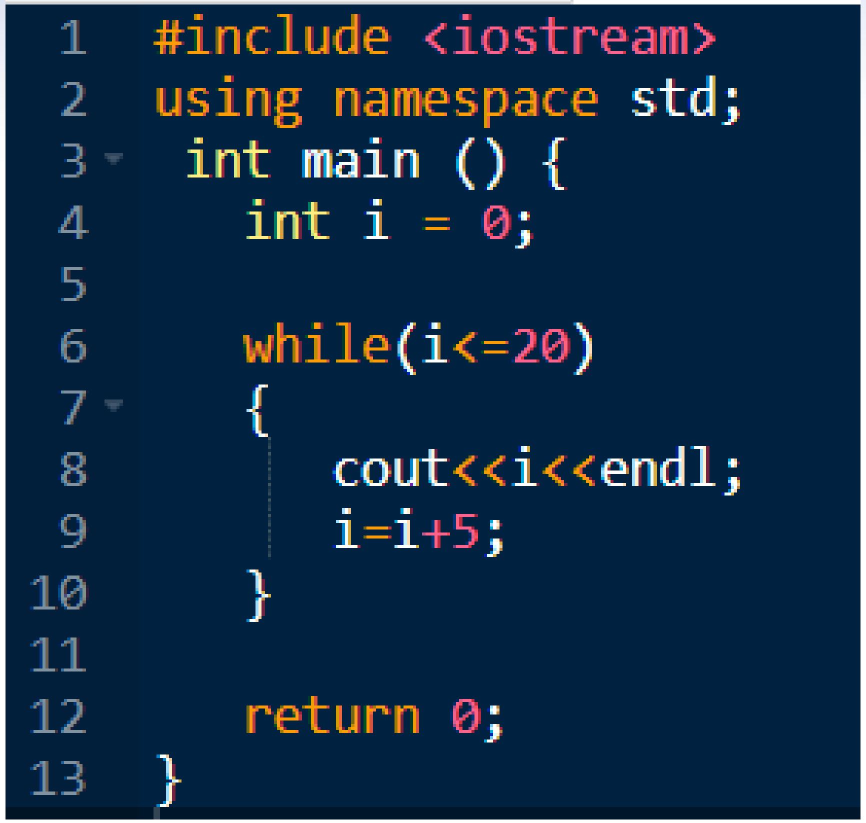
- All variables in the boolean expression must be initialized prior to the loop.
- At least one of the variables in the boolean expression must be assigned a new value inside the loop. In other words, the boolean expression must change its value inside the loop.
- The boolean expression is tested prior to entering the loop and before each repetition of the loop body.
- The entire loop body is executed if the boolean expression is true.
- It is possible not to execute the loop body, this occurs when the boolean expression is initially false.

Infinite Loops

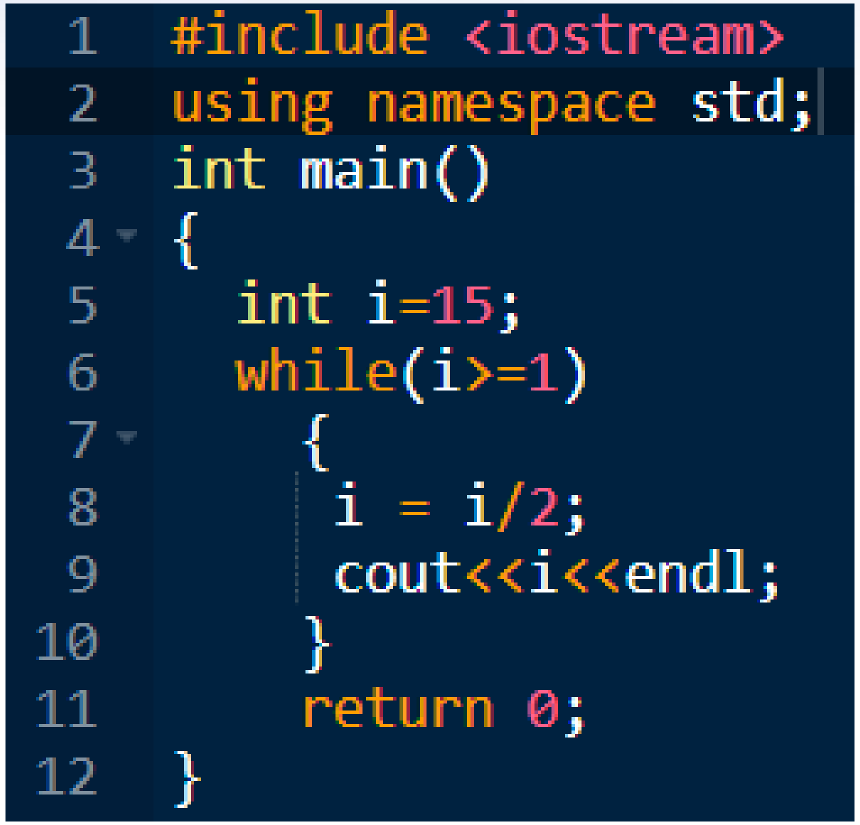
- An infinite loop is one in which the condition is initially satisfied, so the loop is entered, but the condition for exiting the loop is never met.
- Generally, an infinite loop is caused by failing to modify a variable involved in the condition within the loop body.
- To break out of a malfunctioning program press ctrl-C on Linux or ctrl-break, on a DOS or Windows machine.

Practice Question 01

What is the output of the following C++ program?



```
#include <iostream>
using namespace std;
 int main () {
    int i = 0;

    while(i<=20)
    {
       cout<<i<<endl;
       i=i+5;
    }

    return 0;
}
```



```
#include <iostream>
using namespace std;
int main()
{
   int i=15;
   while(i>=1)
     {
      i = i/2;
      cout<<i<<endl;
     }
     return 0;
}
```

Counter-Controlled `while` Loops

- If you know exactly how many pieces of data need to be read, the `while` loop becomes a counter-controlled loop

```
counter = 0;          //initialize the loop control variable

while (counter < N)  //test the loop control variable
{
      .
      .
      .
      counter++;      //update the loop control variable
      .
      .
      .
}
```

Practice Question 02

Write a C++ program to takes a number from the user and print the pattern as follows:

```
Sample Output:

Enter number: 3

3 2 1
```

Sentinel-Controlled `while` Loops

- Sentinel variable is tested in the condition and loop ends when sentinel is encountered.

```
cin >> variable;               //initialize the loop control variable

while (variable != sentinel)   //test the loop control variable
{
     .
     .
     .
     cin >> variable;          //update the loop control variable
     .
     .
     .
}
```

Interactive I/O and Looping

- The sentinel or trailer technique uses a special end-of data value to indicate the end of meaningful data.
- Using the while loop, we place an input statement before the loop to read the first value and an input statement at the bottom of the loop to read the next value.
- The loop condition tests that the value is not equal to the sentinel value.

Practice Question 03

Write a C++ program to prompt the user to enter a set of numbers (only positive numbers), one at a time. User enters a zero to indicate that he has completed entering numbers. Then, the program should display

- the count of odd numbers entered.
- the count of even numbers entered.

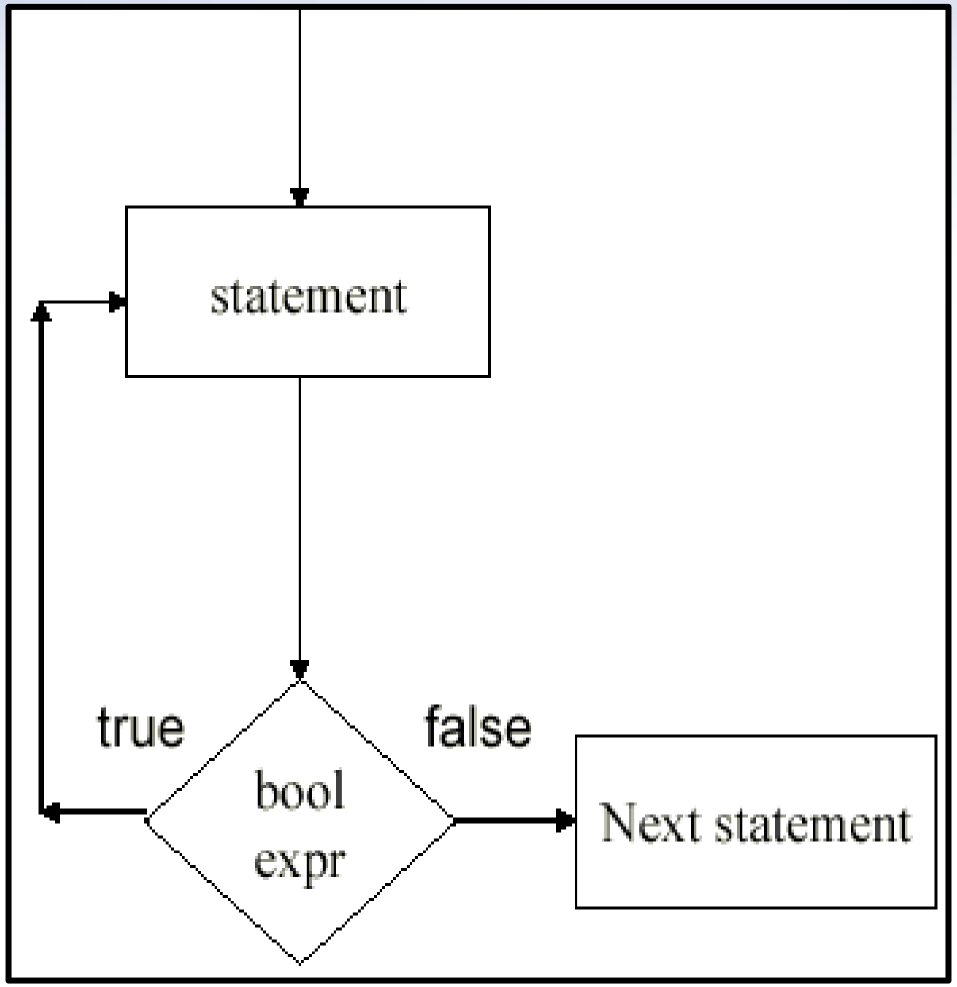
Flag-Controlled while Loops

- A flag-controlled while loop uses a bool variable to control the loop.
- The flag-controlled while loop takes the form:

```
found = false;          //initialize the loop control variable

while (!found)          //test the loop control variable
{
      .
      .
      .
      if (expression)
          found = true; //update the loop control variable
      .
      .
      .
}
```

Post-test loops : do while loop

```
do
  statement;
while (bool expr);


do
{
  statement 1;
  statement 2;
}while (bool expr);
```

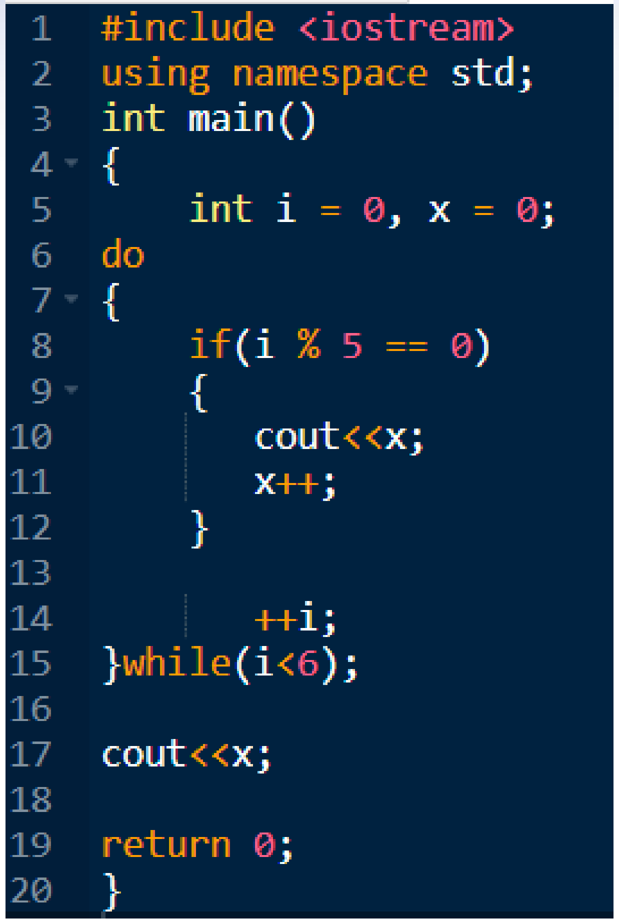


Notes on semantics of Do While Loop

- At least one of the variables in the boolean expression must be assigned a new value inside the loop. In other words, the boolean expression must change value inside the loop.
- The boolean expression is tested at the end of the loop body after each execution of the loop body.
- The entire loop body is executed if the boolean expression is true.
- The loop body is always executed at least once.

Practice Question 04

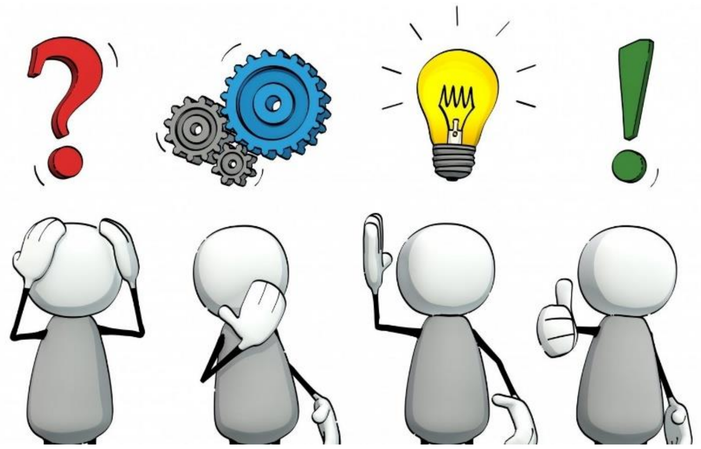
What is the output of the following C++ program?



```cpp
#include <iostream>
using namespace std;
int main()
{
        int i = 0, x = 0;
do
{
        if(i % 5 == 0)
        {
            cout<<x;
            x++;
        }

            ++i;
}while(i<6);

cout<<x;

return 0;
}
```

Determining The Loop To Use

- If the statements of the loop may not be executed at all, use a while loop.
- If the statements of the loop must be executed at least once either loop may be used but the do- while loop is preferable.



SLIIT ACADEMY

Counter –Controlled Loops

- A counter controlled loop is a looping control structure in which a loop variable manages the repetition by counting.
- The syntax for a counter-controlled loop- for loop in C and C++ is:

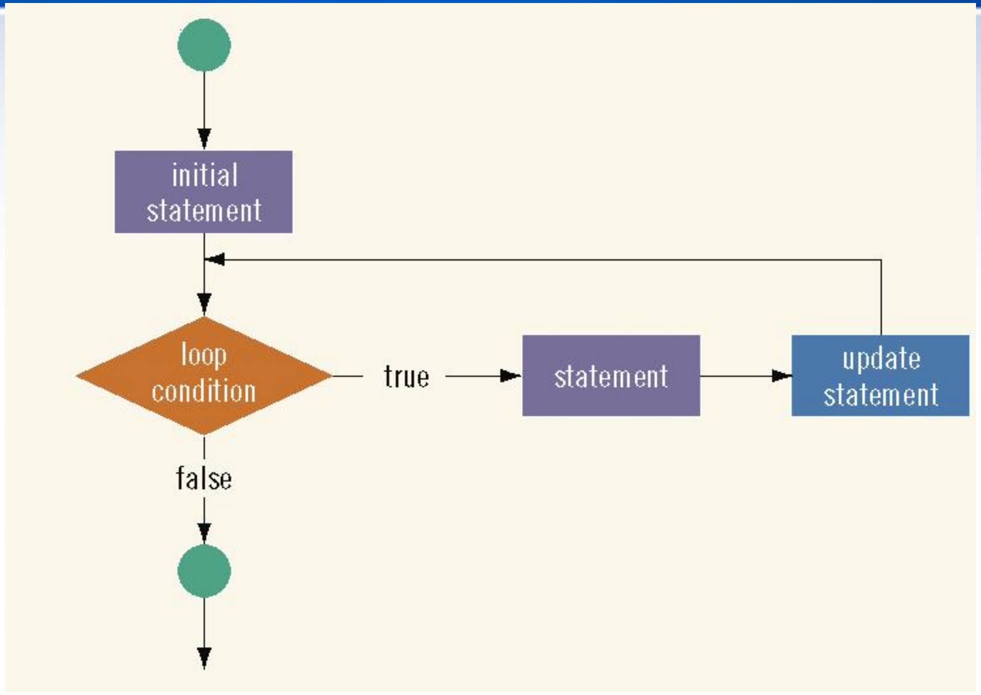
```
for(initial statement; loop condition; update statement)
     statement;
```

- The initial statement, loop condition, and update statement are called for loop control statements.
- A loop body of more than one statement, must be enclosed in curly braces.

Counter –Controlled Loops

The for loop executes as follows:

1. The initial statement executes.
2. The loop condition is evaluated. If the loop condition evaluates to true
   I. Execute the for loop statement.
   II. Execute the update statement (the third expression in the parentheses).
3. Repeat Step 2 until the loop condition evaluates to false



Comments on the for loop

- If the loop condition is initially false, the loop body does not execute.
- The update expression, when executed, changes the value of the loop control variable (initialized by the initial expression), which eventually sets the value of the loop condition to false. The for loop body executes indefinitely if the loop condition is always true.
- C++ allows you to use fractional values for loop control variables of the double type (or any real data type).

Comments on the for loop

- All three expressions that are part of the for loop are optional. The semicolons are not optional.
- In the `for` statement, if the loop condition is omitted, it is assumed to be `true.` Then the loop becomes an infinite loop.
- Use the for loop when you know exactly how many times the loop body is to be executed, either as a specific value or as an expression.

All parts of the for loop are optional

```
for(i=0; i< 5; i++)
```

```
for(i= 0; i< 5;){
   i++;
   cout << i << endl;
}
```

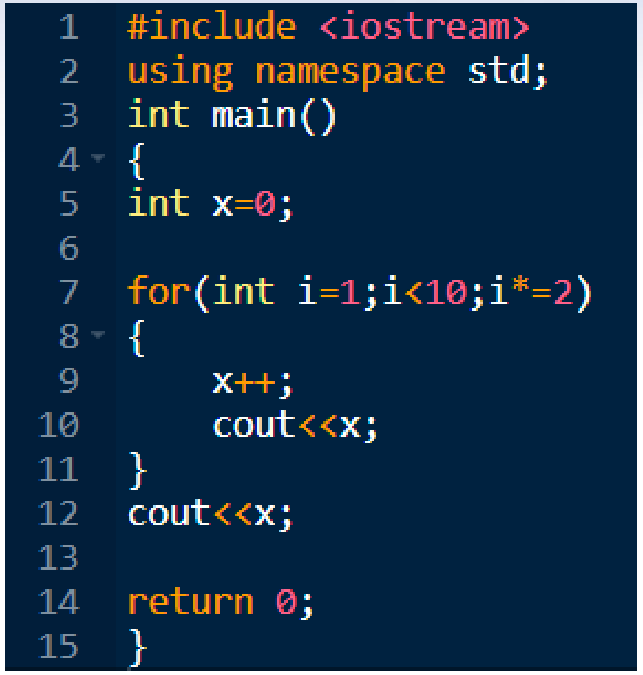
```
i = 0;
for(; i< 5; i++)
   cout << i << endl;
```

```
for( ; ; )
```

In a for statement, you can omit all three statements—initial statement, loop condition, and update statement.

Practice Question 05

What is the output of the following C++ program?



```cpp
#include <iostream>
using namespace std;
int main()
{
int x=0;

for(int i=1;i<10;i*=2)
{
    x++;
    cout<<x;
}
cout<<x;

return 0;
}
```

Practice Question 06

Write a C++ program to read a number and print a pattern as follows:

Use the while, do-while and for loops to write the answer.

```
Sample Output:

Enter number: 3

Output: @@@
```

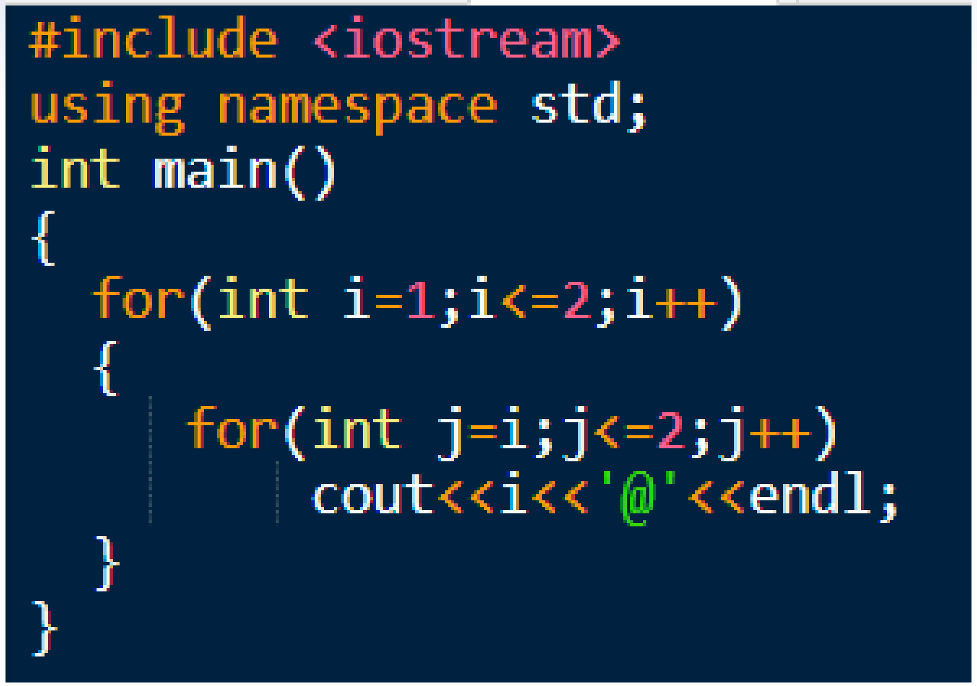
Nested Control Structures

- A nested control statement is a control statement that is contained within another control statement.
- Example : Suppose we want to create the following pattern

```
*
* *
* * *
* * * *
* * * * *
```

Practice Question 07

What is the output of the following C++ program?



```
#include <iostream>
using namespace std;
int main()
{
  for(int i=1;i<=2;i++)
  {
     for(int j=i;j<=2;j++)
         cout<<i<<'@'<<endl;
  }
}
```

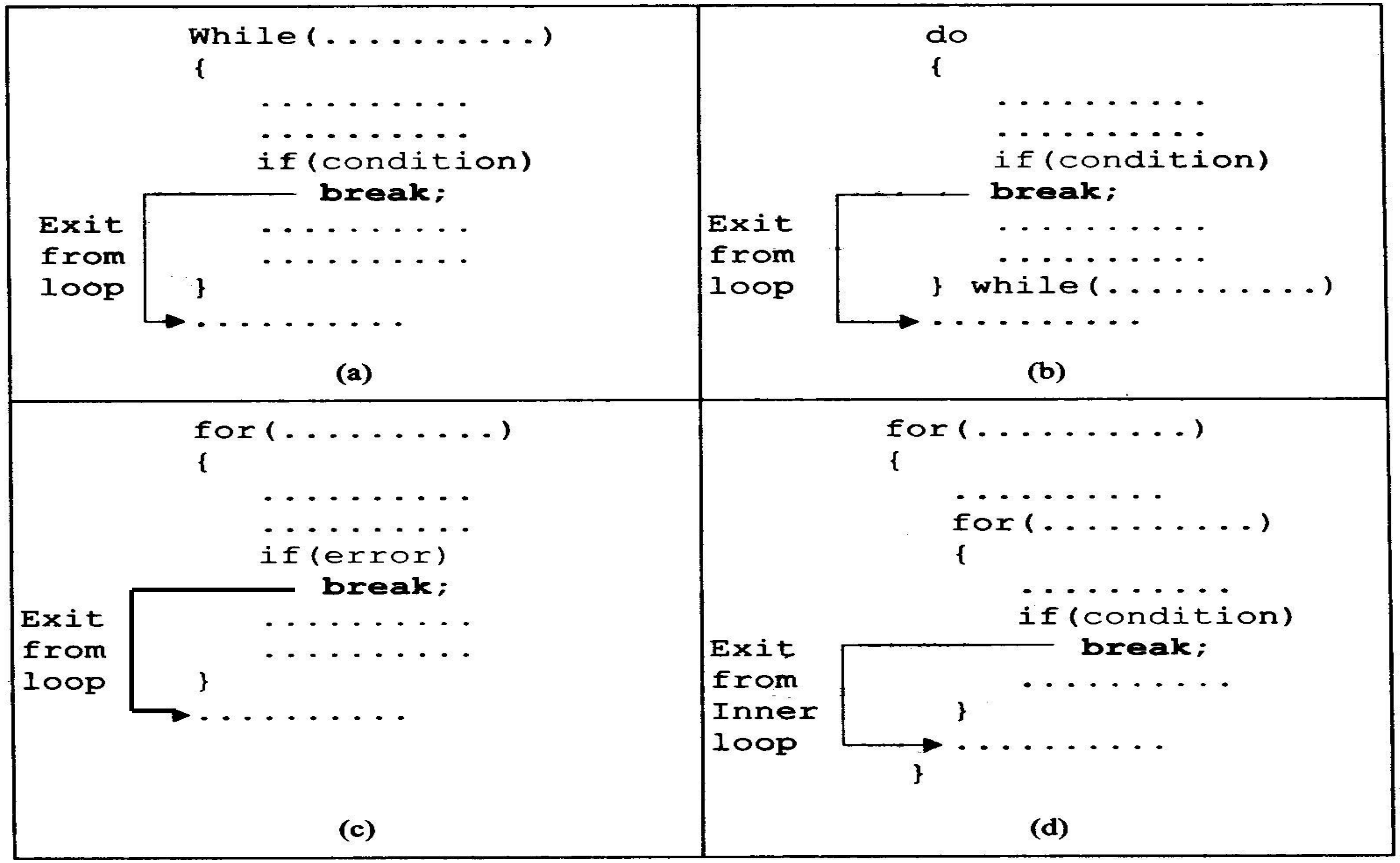
Loop Control Statements

- In C++, you have loop control statements that can be **used to alter or control the flow of loop execution based on specified conditions.**
- Loop control statements change execution from its normal sequence.
- C++ supports for the following control statements.
    - **break statement**
    - **continue statement**
    - **goto statement**

C++ break Statement

- When the break statement executes in a repetition structure, it break or terminate the execution of the loop. But the break statement, in a switch structure, provides an immediate exit.
- The break statement can be used in while, for, and do...while loops.
- The break statement is used for two purposes:
  - To exit early from a loop
  - To skip the remainder of the switch structure

Working of C++ break Statement



```
While(..........)
{
    ..........
    ..........
    if(condition)
        break;
    ..........
    ..........
}
..........
```

(a)

```
do
{
    ..........
    ..........
    if(condition)
    break;
    ..........
    ..........
} while(..........)
..........
```

(b)

```
for(..........)
{
    ..........
    ..........
    if(error)
        break;
    ..........
    ..........
}
..........
```

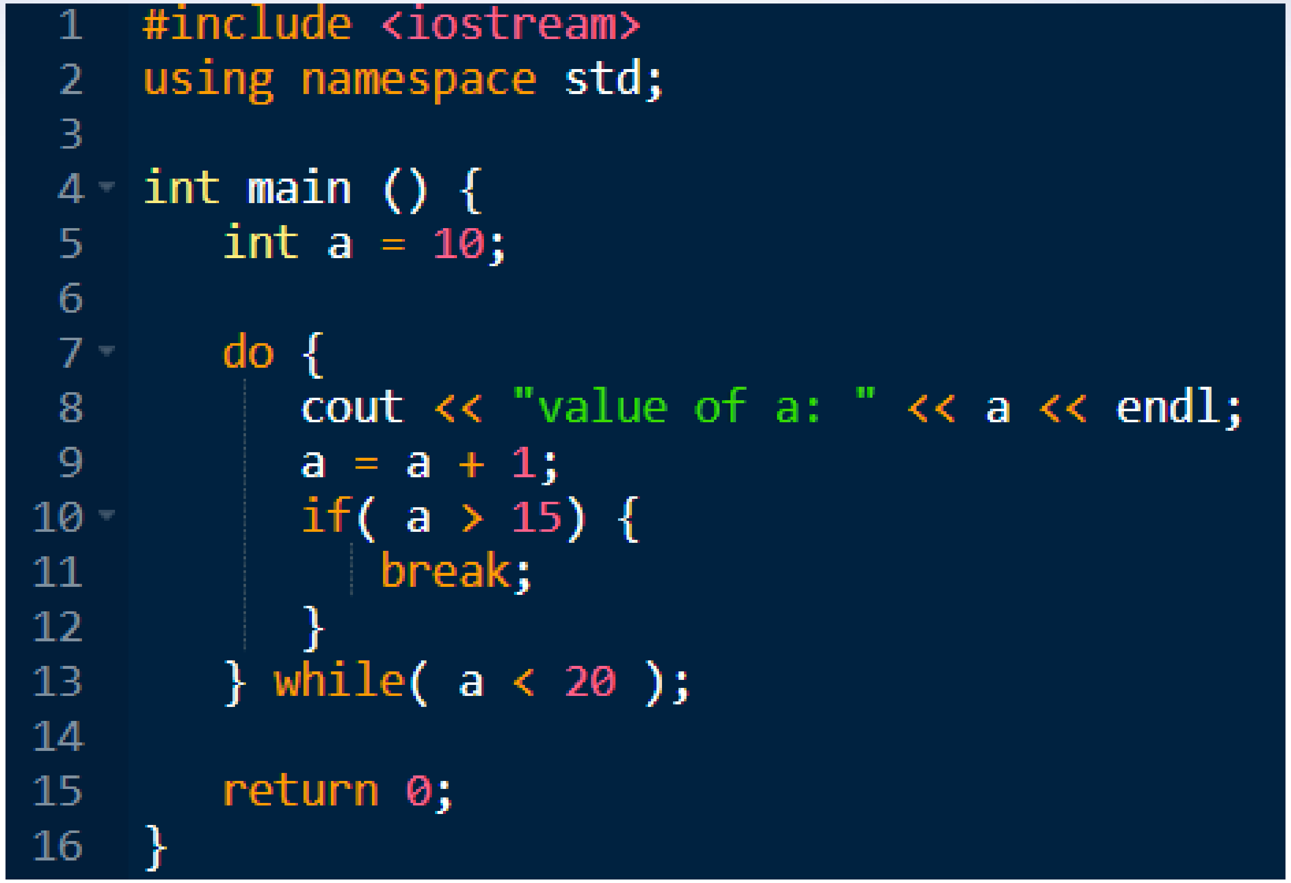
(c)

```
for(..........)
{
    ..........
    for(..........)
    {
        ..........
        if(condition)
            break;
        ..........
    }
    ..........
}
```

(d)

Practice Question 08

What is the output of the following C++ program?



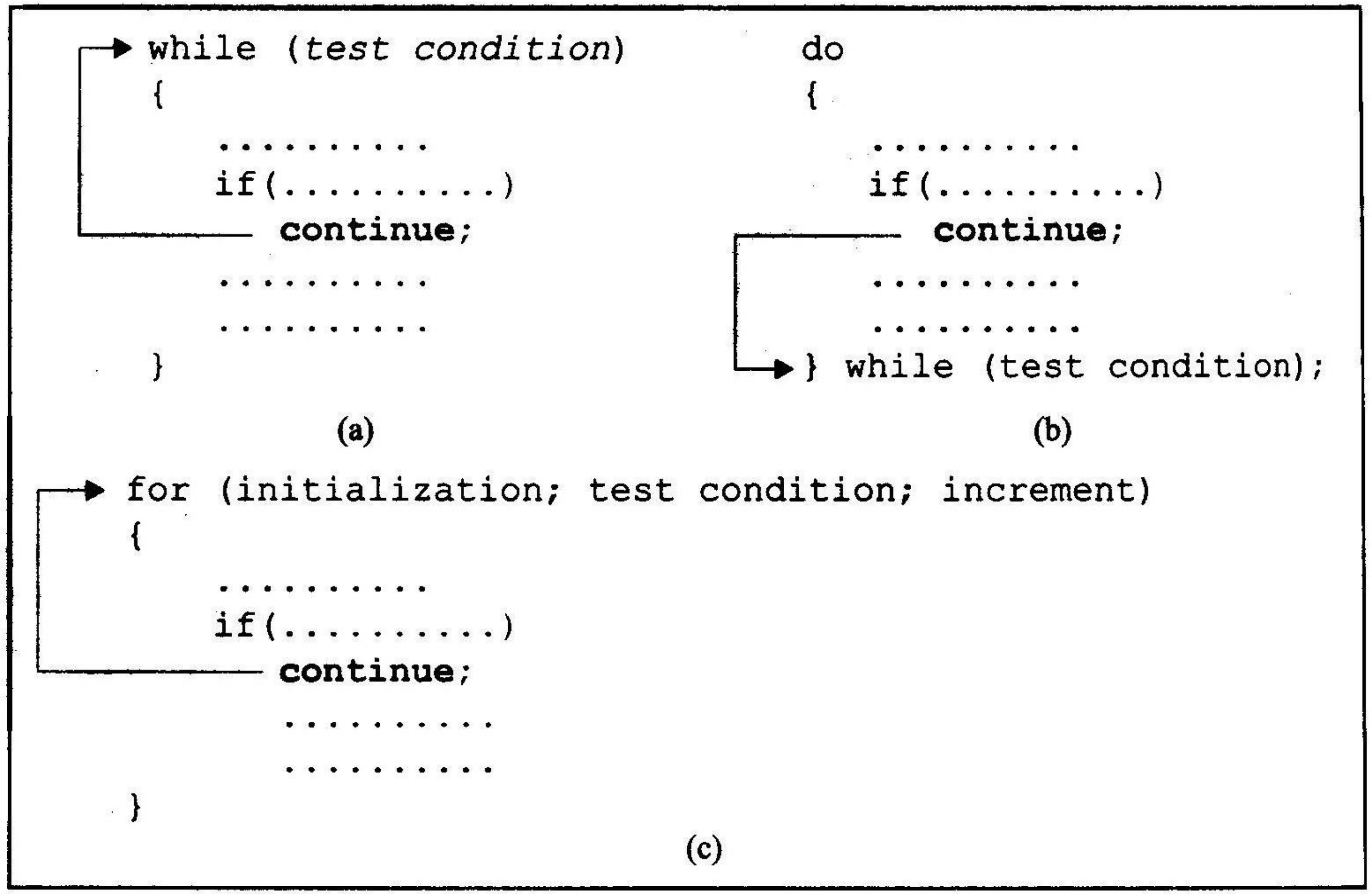
```cpp
#include <iostream>
using namespace std;

int main () {
    int a = 10;

    do {
        cout << "value of a: " << a << endl;
        a = a + 1;
        if( a > 15) {
            break;
        }
    } while( a < 20 );

    return 0;
}
```

C++ continue Statement

- In C++, the continue statement gives you a way to skip over the current iteration of any loop.
- When a continue statement is encountered in the loop, **the rest of statements in the loop body of current iteration will not be execute and returns the program execution to the very first statement in the loop body.**
- It does not terminate the loop rather continues with the next iteration.

Working of C++ continue Statement

- while and do...while structure: Expression (loop-continue test) is evaluated immediately after the continue statement.
- for structure, the update statement is executed after the continue statement ,Then the loop condition executes.



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

(a)

```
do
{
    ..........
    if(..........)
      continue;
    ..........
    ..........
} while (test condition);
```

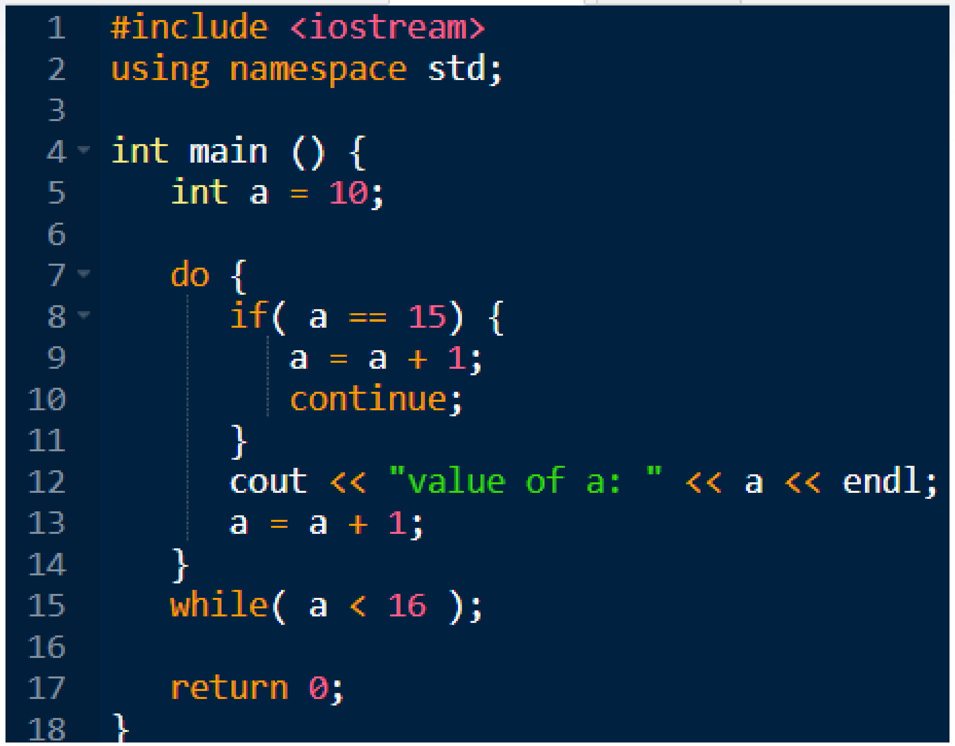
(b)

```
for (initialization; test condition; increment)
{
    ..........
    if(..........)
      continue;
      ..........
      ..........
}
```

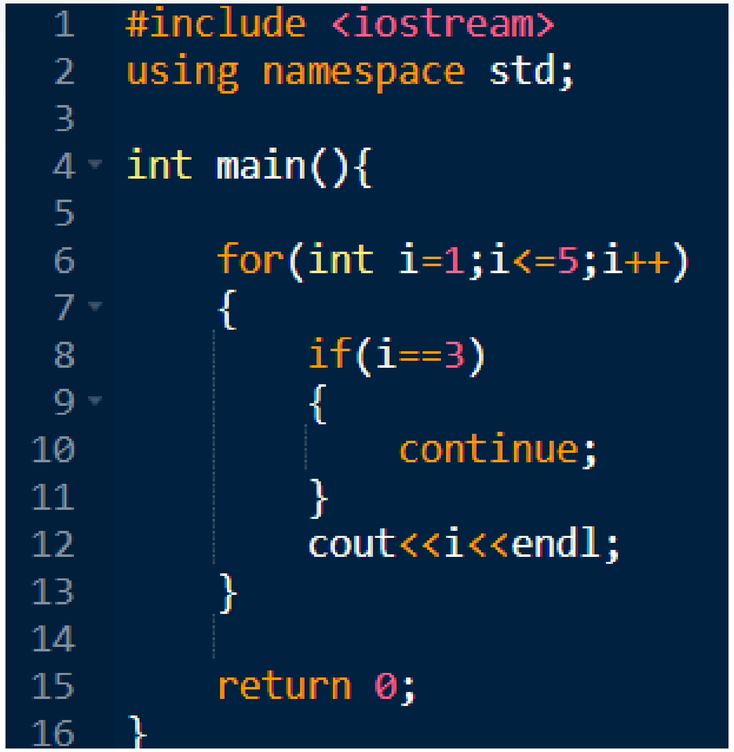
(c)

Practice Question 09

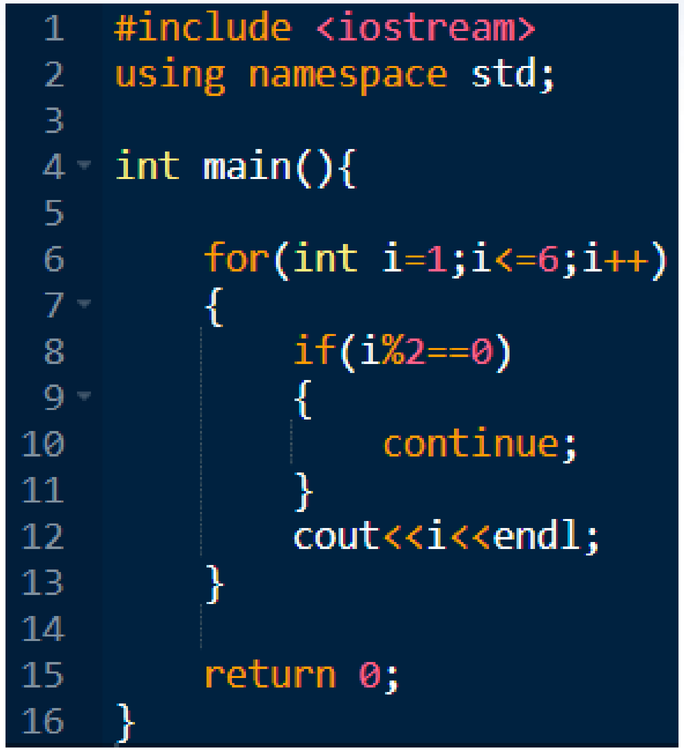
What is the output of the following C++ program?



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

int main () {
   int a = 10;

   do {
      if( a == 15) {
         a = a + 1;
         continue;
      }
      cout << "value of a: " << a << endl;
      a = a + 1;
   }
   while( a < 16 );

   return 0;
}
```



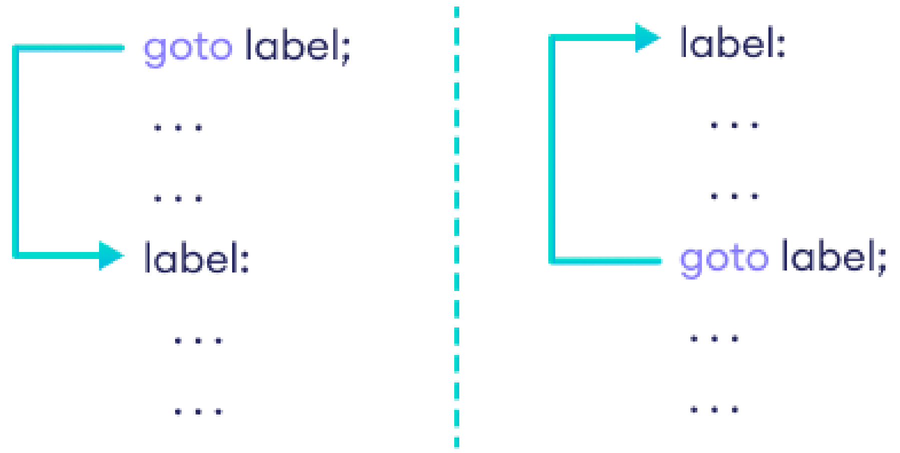
```cpp
#include <iostream>
using namespace std;

int main(){

    for(int i=1;i<=5;i++)
    {
        if(i==3)
        {
            continue;
        }
        cout<<i<<endl;
    }

    return 0;
}
```



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

int main(){

    for(int i=1;i<=6;i++)
    {
        if(i%2==0)
        {
            continue;
        }
        cout<<i<<endl;
    }

    return 0;
}
```

goto Statement

- In C++ programming, the goto statement is used for altering the normal sequence of program execution by transferring control to some other part of the program.



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 (:).

Reason to Avoid goto Statement

- The goto statement gives the power to jump to any part of a program but, makes the logic of the program complex and tangled.
- In modern programming, the goto statement is considered a harmful construct and a bad programming practice.
- In The goto statement can be replaced in most of C++ program with the use of break and continue statements.

Summary

- Introduction to repetition statements in C++
- While loop
- Do-while Loop
- For Loop
- Loop Control Statements