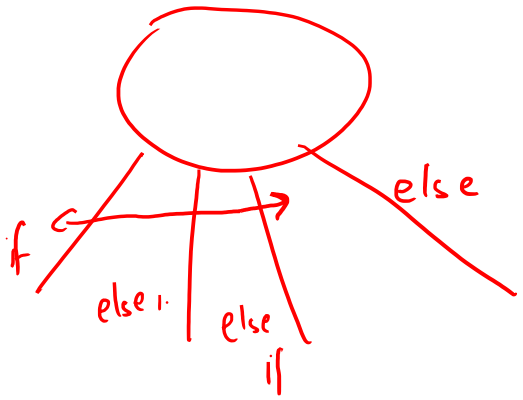
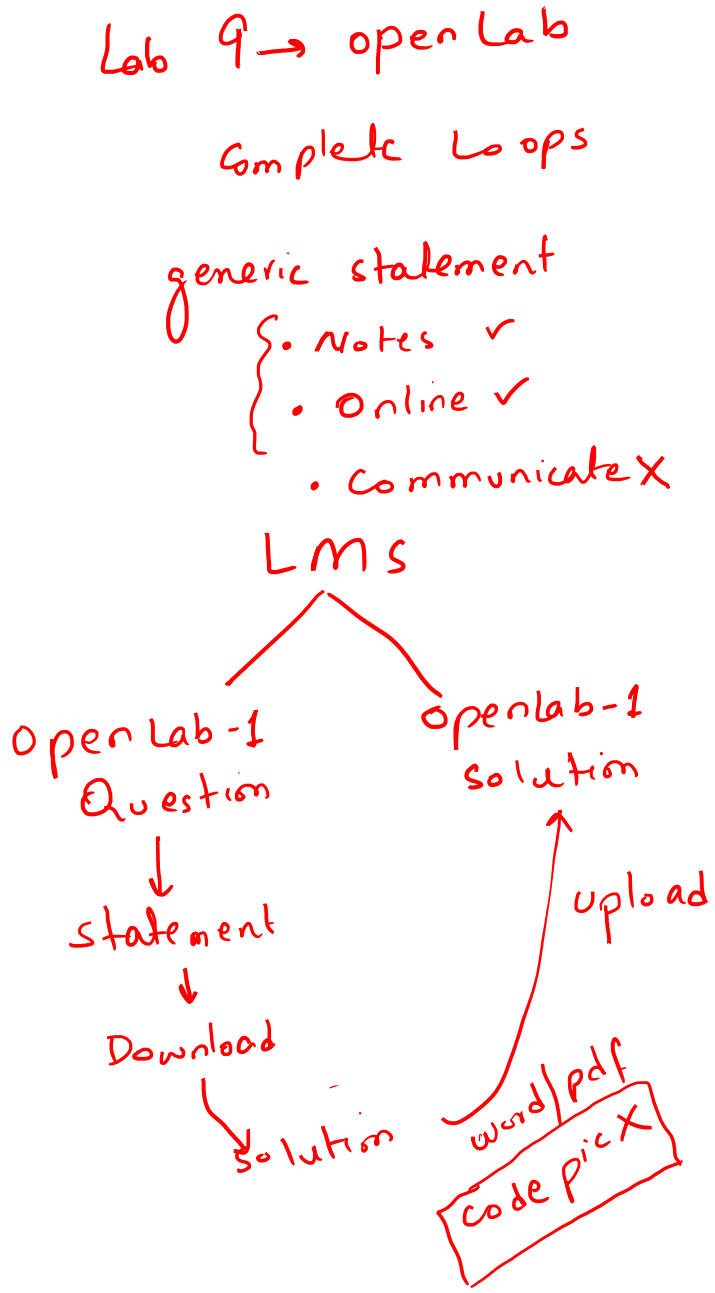
Fundamentals of Programming
CS-114



Lecture 6

Loops


Lab 9 → openLab
Complete Loops
generic statement
• Notes ✓
• Online ✓
• communicate X
LMS
openLab-1
Question
statement
Download
solution
openlab-1
solution
upload
word/pdf
codepic X

Why Is Repetition Needed?

Loop

- Repetition allows you to efficiently use variables
- Can input, add, and average multiple numbers using a limited number of variables
- For example, to add five numbers:
  - Declare a variable for each number, input the numbers and add the variables together
  - Create a loop that reads a number into a variable and adds it to a variable that contains the sum of the numbers

- while
- do while
- for

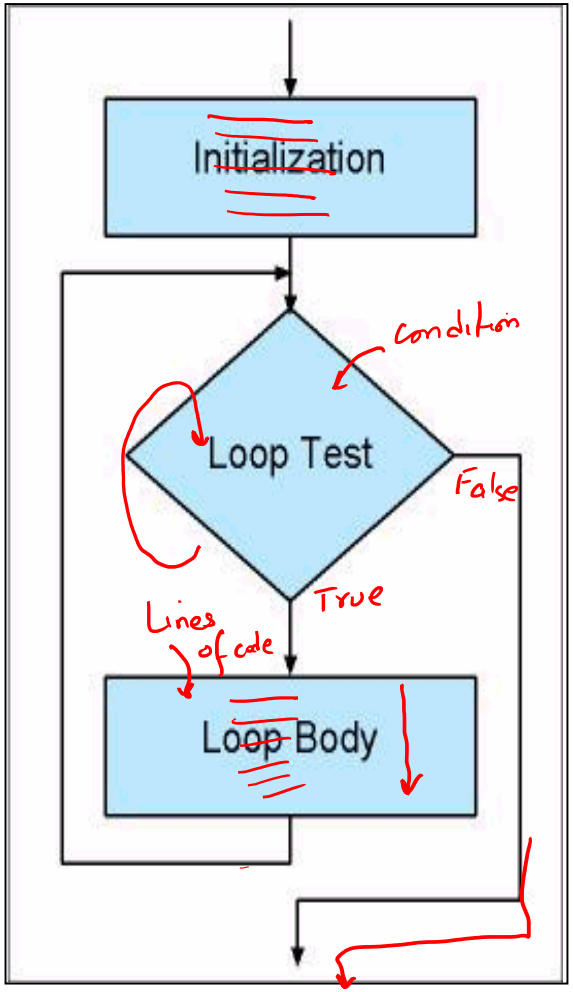
while Loop

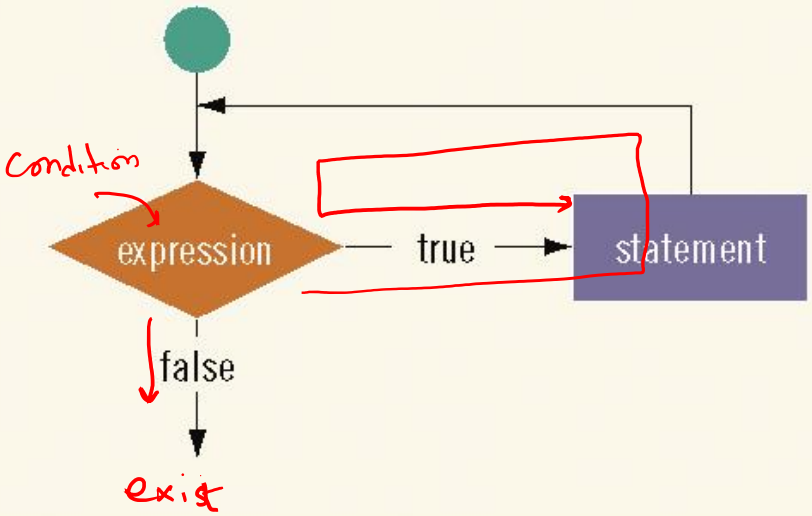
condition

- The for loop does something a fixed number of times.
- What happens if you don't know how many times you want to do something before you start the loop?
- In this case a different kind of loop may be used: the while loop.

while Loop

- **while loop** executes a set of statements as long as the condition specified at the beginning is **true**.
- The condition is evaluated at the beginning of the loop
  - If it is true, the loop will execute
  - If it is false, the loop will not execute even once




Condition
expression
true
statement
false
exit

while Loop

Syntax:

```
while(condition)
        {statement1;
         …
         statement n;
        }
```

Trace while Loop

```
int count = 0;
while (count < 2)
{
    cout<<"Welcome to C++!";
    count++;
}
```

Initialize count

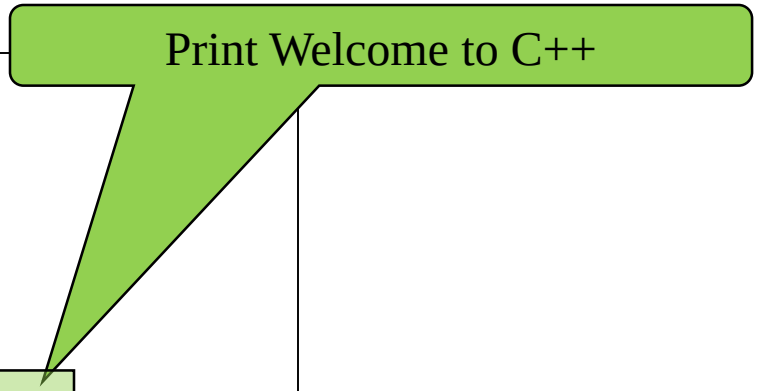
Trace while Loop, cont.

(count < 2) is true / false

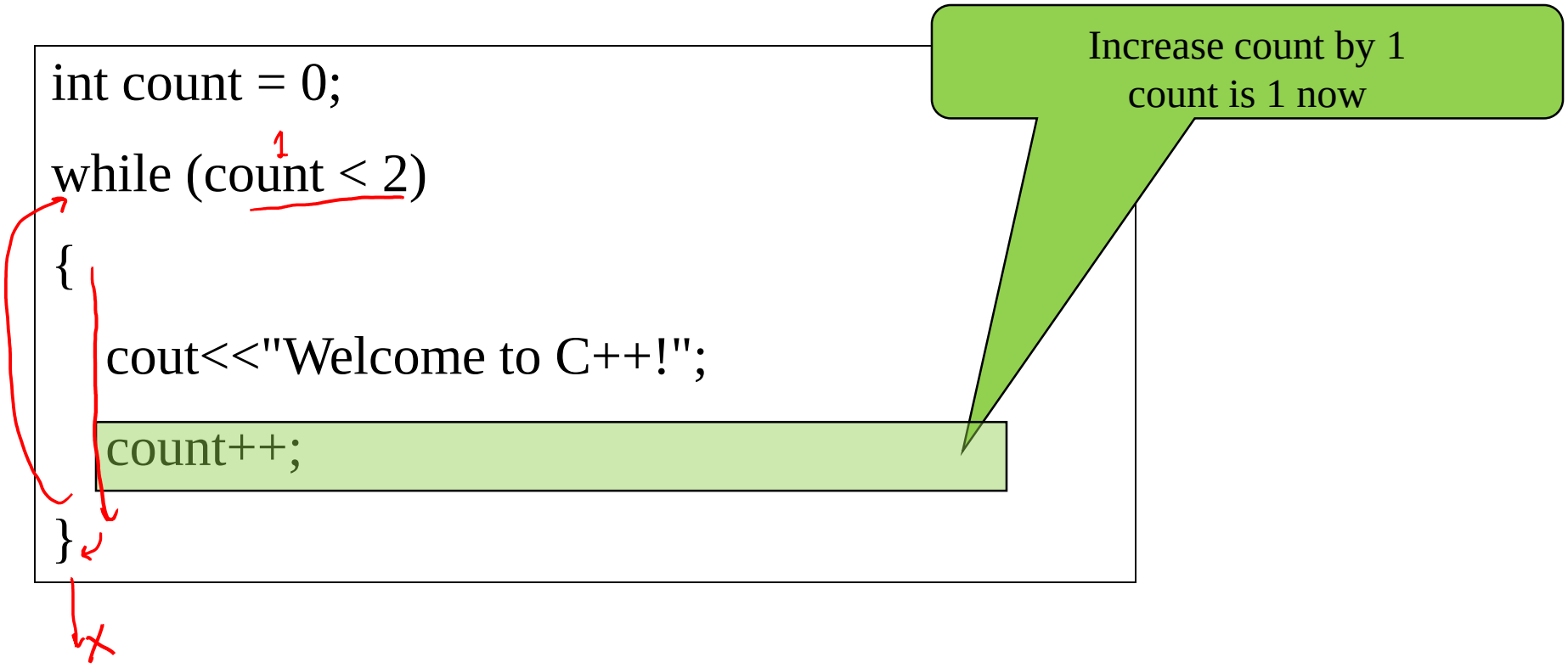
```
int count = 0;
while (count < 2)
{
    cout<<"Welcome to C++!";
    count++;
}
```

Trace while Loop, cont.

```
int count = 0;
while (count < 2)
{
  cout<<"Welcome to C++!";
  count++;
}
```



Trace while Loop, cont.



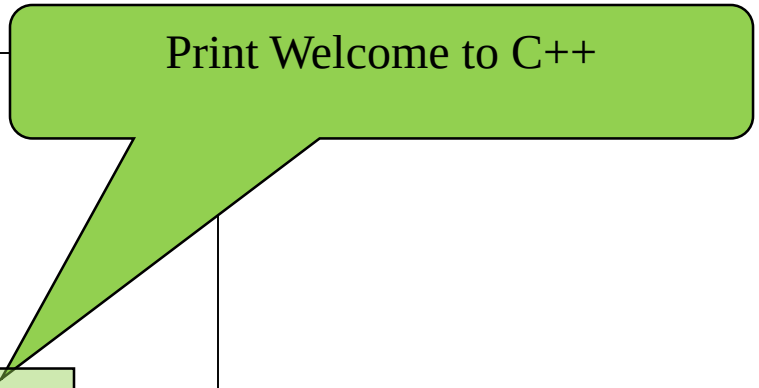
Trace while Loop, cont.

```
int count = 0;
while (count < 2)
{
    cout<<"Welcome to C++!";
    count++;
}
```

(count < 2) is still true since count is 1

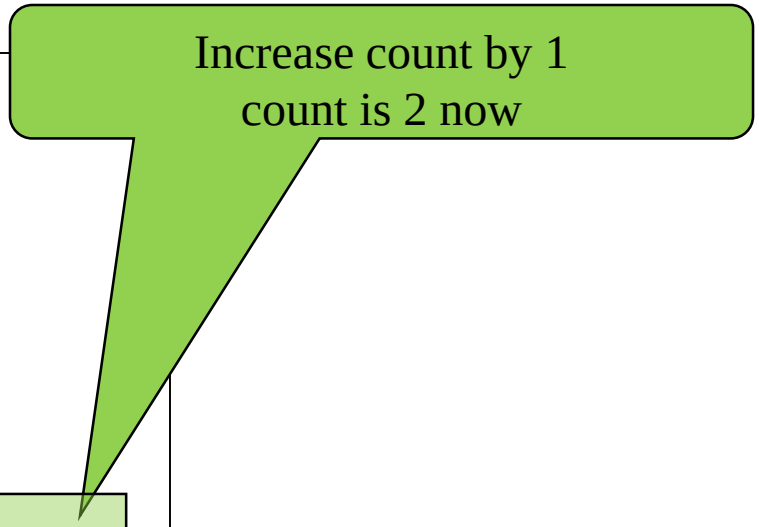
Trace while Loop, cont.

```
int count = 0;
while (count < 2)
{
   cout<<"Welcome to C++!";
   count++;
}
```



Trace while Loop, cont.

```
int count = 0;
while (count < 2)
{
    cout<<"Welcome to C++!";
    count++;
}
```

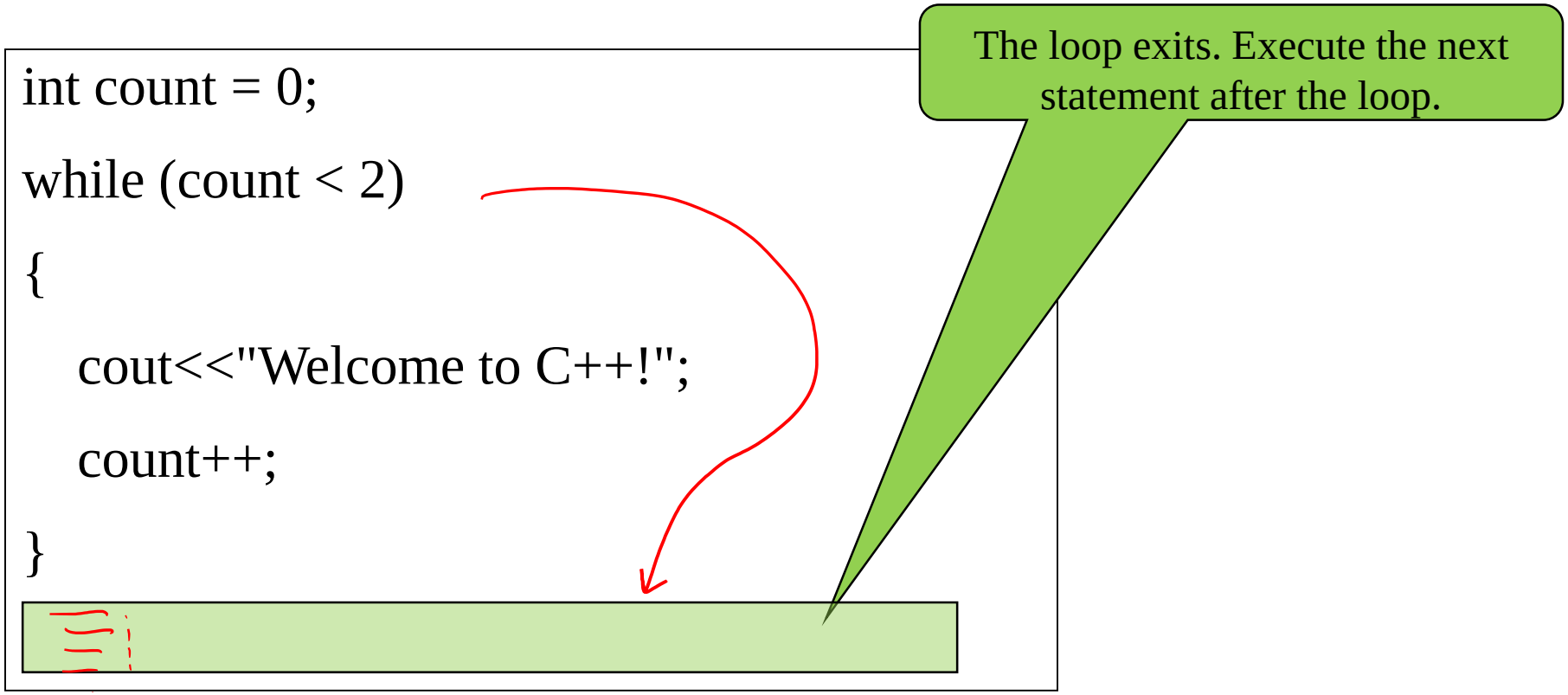


Trace while Loop, cont.

```
int count = 0;
while (count < 2)
{
    cout<<"Welcome to C++!";
    count++;
}
```

(count < 2) is false since count is 2 now

Trace while Loop



Example

```
#include <iostream>
using namespace std;
 int main()
{
        int n = 99; // make sure n isn't initialized to 0
        while( n != 0 ) // loop until n is 0
        {
            cin >> n; // read a number into n
        }
        cout << endl;
        return 0;
}
```

repeat

Sample output

- Here's some sample output. The user enters numbers, and the loop continues until 0 is entered, at which point the loop and the program terminate.

1

27

33

144

9

0

Infinite Loops

- Loops that never stop are infinite loops
- The loop body should contain a line that will eventually cause the boolean expression to become false
- Example: Print the odd numbers less than 12

```
x = 1;
while (x != 12)
{
    cout << x << endl;
    x = x + 2;
}
```

Never True

odd values

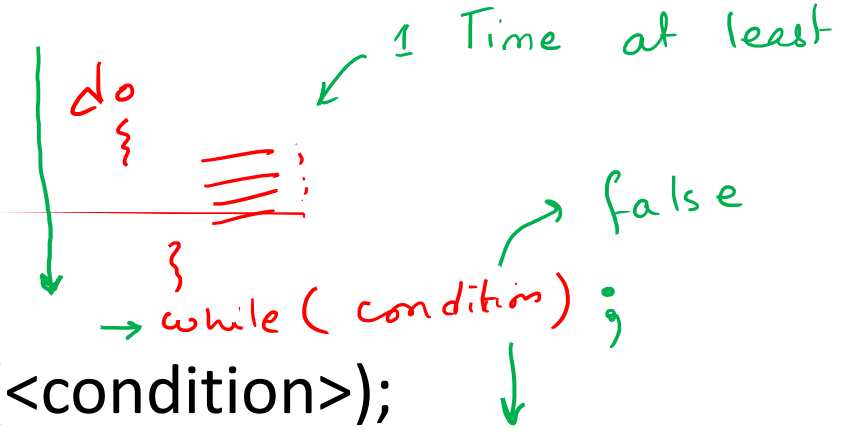
1
3
5
7
9
11
13
15
⋮ infinite

- Better to use this comparison: while (x < 12)

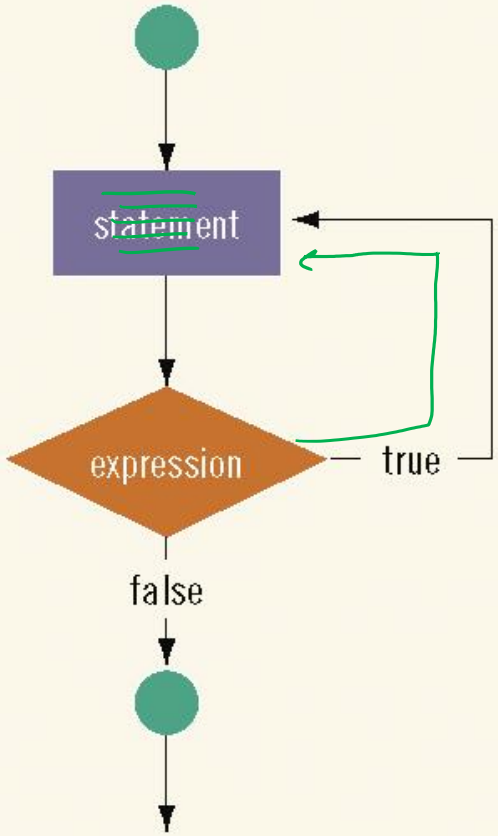
do-while

- Similar to while loop, except the condition is **at the end** of the loop.
  - The loop always executes at least once, regardless of whether the condition is true or not.
- **Syntax:**

```
do
    {statement 1;
    ...
    statement n; } while(<condition>);
```



Note: A semicolon at the end and curly braces are always required.

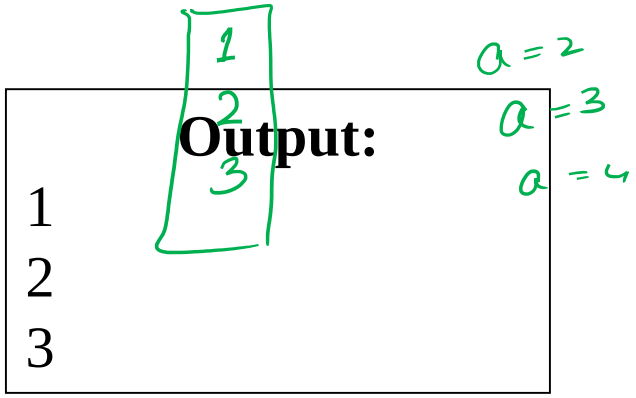

statement
expression
true
false


while (not edge) {
run();
}
do {
run();
} while (not edge);

do-while

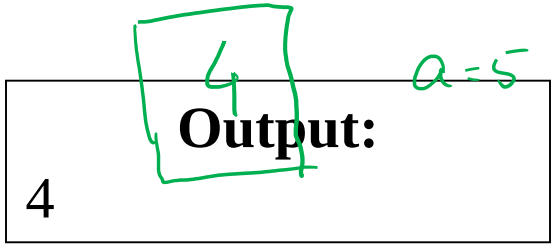
- **Example**

```
int a=1;
do {
    cout<<a<<endl;a++; }while(a<=3);
```



- However if,

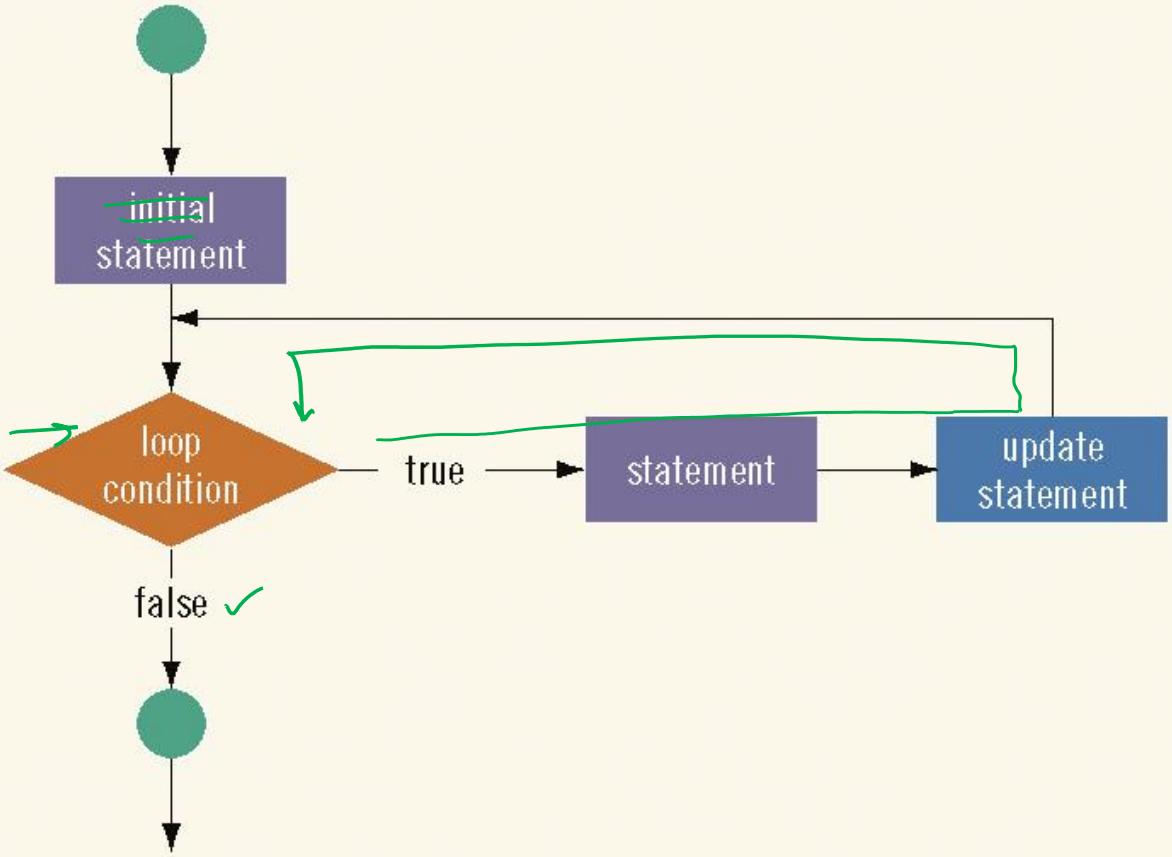
```
int a=4;
do {
    cout<<a<<endl;a++; }while(a<=3);
```



The for-Statement

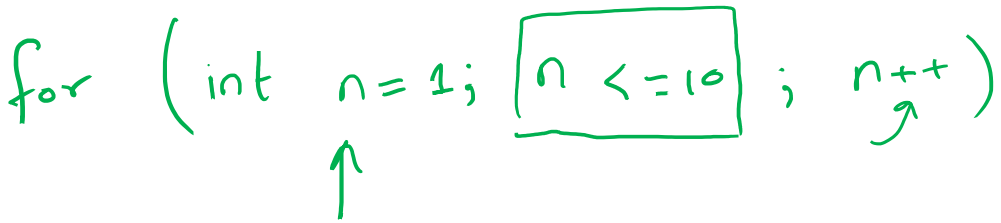
- A for-Statement (for-loop) is another loop mechanism in C++
  - Designed for common tasks such as adding numbers in a given range
  - Is sometimes more convenient to use than a while loop
  - Does not do anything a while loop cannot do

For Loop Dissection

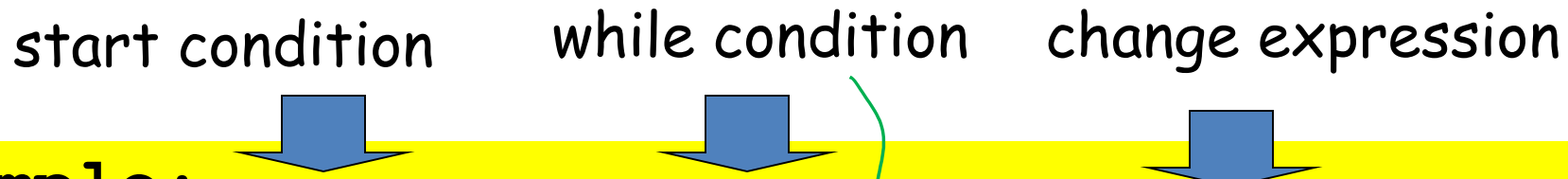


For Loop Dissection

- The for loop uses the same components as the while loop in a more compact form
  - `for (n = 1; n <= 10; n++)`



The for Statement Syntax




```
Example:
    for (count=1; count < 7; count++)
    {
        cout << count << endl;
    }
    //next C++ statements;
```

for Loop Alternative

- A for loop can also include a variable declaration in the initialization action
  - `for (int n = 1; n < = 10; n++)`
  - This line means
    - Create a variable, n, of type int and initialize it with 1
    - Continue to iterate the body as long as n <= 10
    - Increment n by one after each iteration
- For-loop syntax and while loop comparison are found in

The for-loop Body

- The body of a for-loop can be
  - A single statement
  - A compound statement enclosed in braces
    - Example:

```
for(int number = 1; number >= 0; number--)
{
    // loop body statements
}
```

2 times

```
int number = 1;
while ( number >= 0 )
{
    ≡
    number--;
}
```

for/while Loop Comparison

- ```
  int  sum = 0;
  int  n = 1;
  while(n <= 10)  // add the numbers 1 - 10
  {
     sum = sum + n;
     n++;
   }
  ```

- ```
  int  sum = 0;
  int  n = 0;
  for (n = 1; n <= 10; n++)  //add the numbers 1 - 10
  sum = sum + n;
  ```

The *for* Statement

for Statement

Syntax

```
for (Initialization_Action; Boolean_Expression; Update_Action)
    Body_Statement
```

Example

```
for (number = 100; number >= 0; number--)
{
    cout << number
         << " bottles of beer on the shelf.\n";
}
```

Equivalent *while* loop

Equivalent Syntax

```
Initialization_Action;
while (Boolean_Expression)
{
    Body_Statement
    Update_Action;
}
```

Equivalent Example

```
number = 100;
while (number >= 0)
{
    cout << number
         << " bottles of beer on the shelf.\n";
    number--;
}
```

Output

```
100 bottles of beer on the shelf.
99 bottles of beer on the shelf.
                .
                .
                .
0 bottles of beer on the shelf.
```

3, 9, 2, 1, 7 → ascending order

arrays

```
int a=3, b=9, c=2, d=1, e=7;
if( a >b && a>c && a>d && a>e )
    cout << a;
else if( b>c && b>d && b>e )
    cout << b;
if ( c > d && c>e )
    cout << c;
```

Loops

repeat

first letter

take avg of 5 subj marks

```
int i=0; float mark, sum = 0, avg;
while (i<5)
  {
    cout<<"input your grades"<< i+1;
    cin >> mark;
    sum += mark;          //sum = sum + mark;
      i++;
  }
  avg = sum/i;
  cout <<" your avg is " << avg;
```

Factorial

$6! = 6 \times 5 \times 4 \times 3 \times 2 \times 1$

```
int n;
cin >> n;
int product = 1;
for (int i=1; i<=n; i++)
{
    product = i * product;
}
cout << product;
```

# of digits

sum of all digits → 653 → 6+5+3 = 14

```
int n, sum=0, m;
cout << "Enter a number";
cin >> n;
while (n==0)
 {
    m = n%10;
    sum += m;   // 0+3+5+6
      n = n/10;
 }
cout << sum;
```

653 %10 → 3

653/10 → 65

65 %10 → 5

65/10 → 6

6 %10 → 6

6/10 → 0