

Loops: Repeating One or More Statements

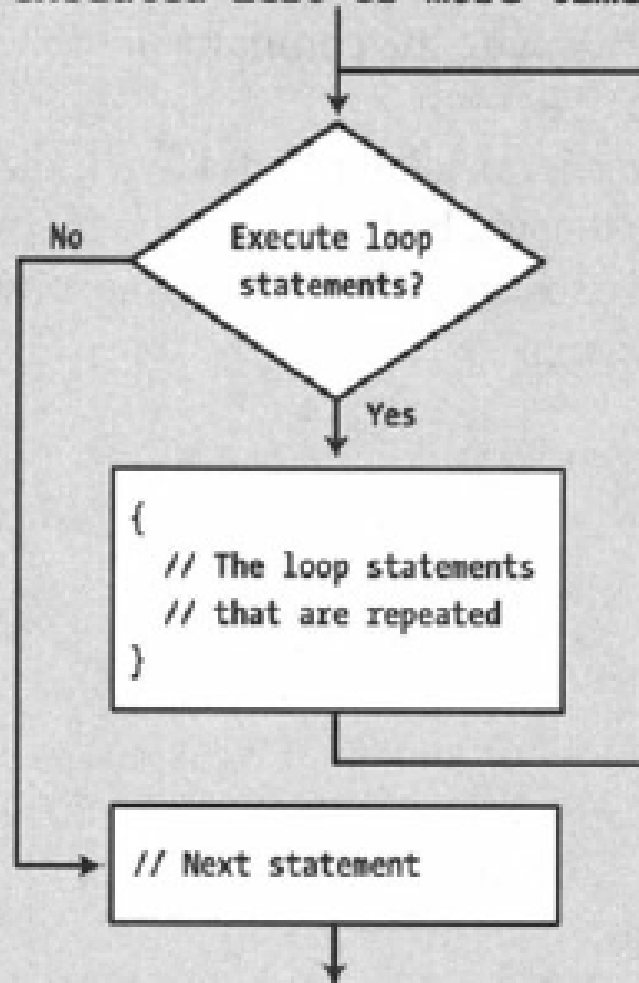
- for loop
- while loop
- do-while loop
- break statement in a loop
- continue statement in a loop
- Nested loops

Loops

- Iteration statements
 - `while (conditions) { statements; }`
 - `do {statements;} while (conditions);`
 - `for (initializing_expression; conditions; iteration_expression) { statements; }`

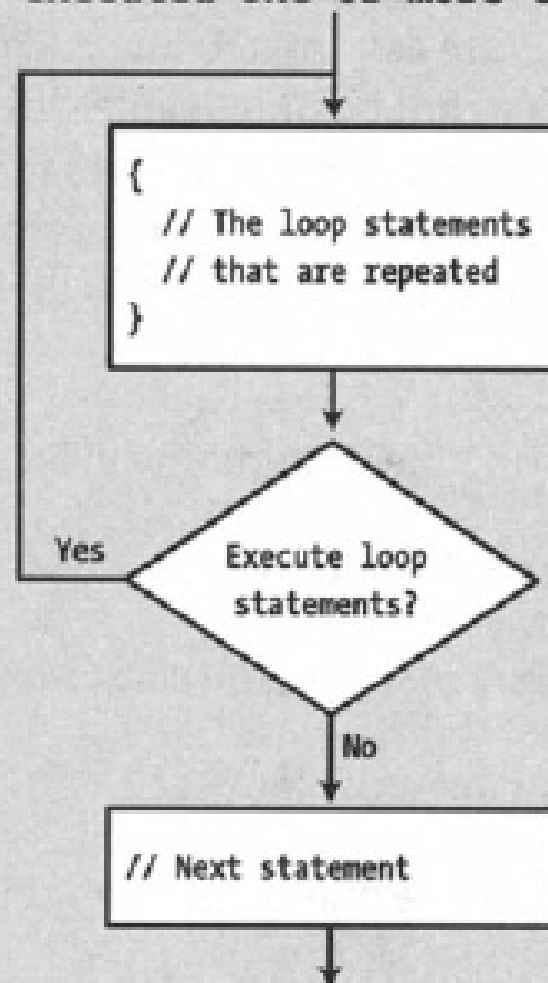
while loop

Loop statements
executed zero or more times



do-while loop

Loop statements
executed one or more times



The while Loop

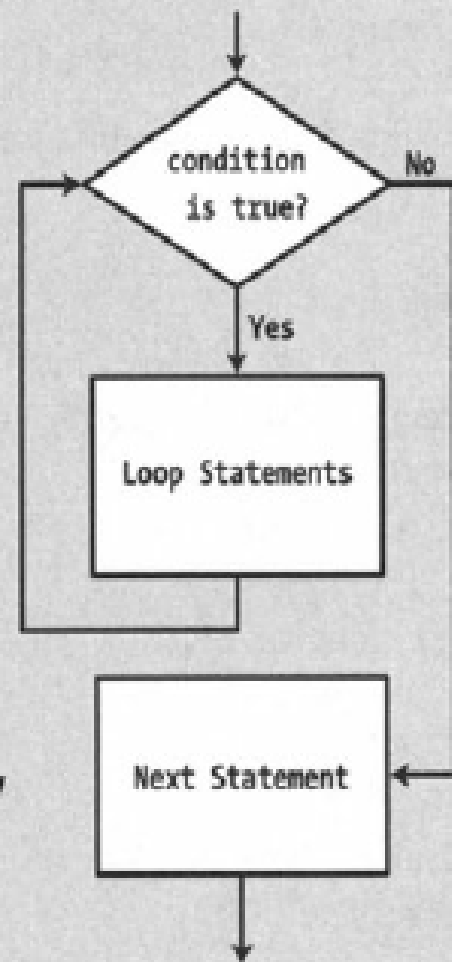
This expression is evaluated at the beginning of each loop iteration. If it is true, the loop continues, and if it is false, execution continues with the statement after the loop.

```
while( condition ) {
```

```
    // Loop statements to be executed repeatedly
```

```
}
```

```
// Next statement
```



while Loop

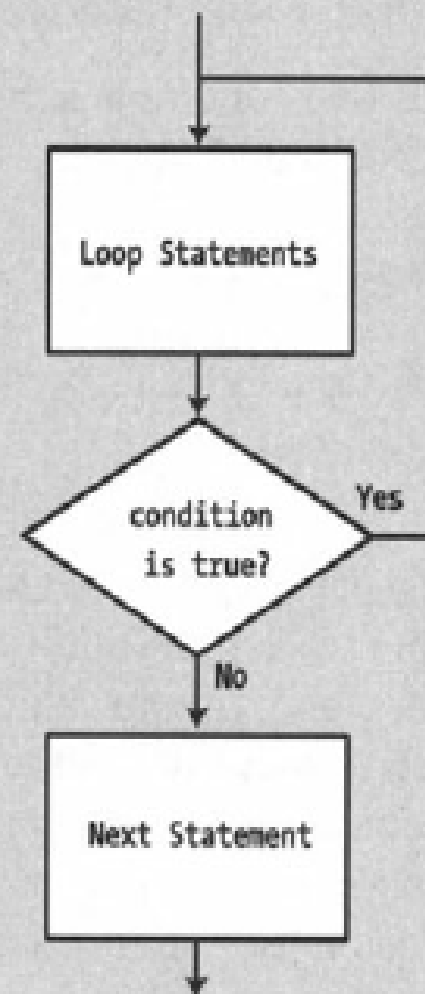
```
while (condition)
{
    statements;
}
```

- Program 5.1

The do-while Loop

```
do {  
    // Loop statements to be executed repeatedly  
}while( condition );  
// Next Statement
```

This expression is evaluated at the end of each loop iteration. If it is true, the loop continues, and if it is false, execution continues with the statement after the loop. The loop statements are always executed at least once.



The for Loop

```
for( initializing_expression ; condition ; iteration_expression )  
{  
    // Loop statements  
}  
// Next statement
```

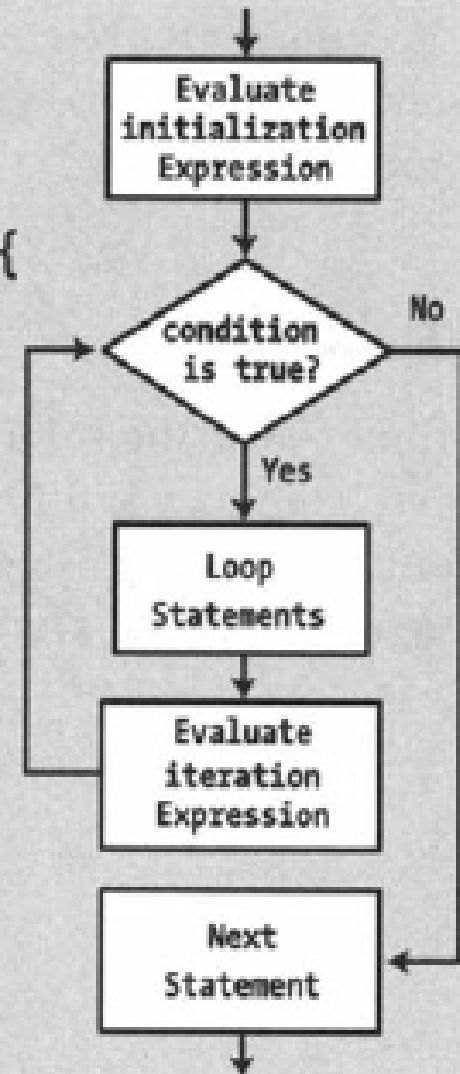
The diagram illustrates the syntax of a C++ for loop. The loop header consists of three parts enclosed in boxes: `initializing_expression`, `condition`, and `iteration_expression`, separated by semicolons. Arrows point from above to each of these boxes. Below the loop body, two arrows point upwards to the semicolons separating the three expressions. Accompanying text states: "The semicolons must always be present. The expressions need not be."

for(initializing_expression ; condition ; iteration_expression)
{
 // Loop statements
}
// Next statement

The semicolons must always be present.
The expressions need not be.

for Flowchart

```
for(initialization ; condition ; iteration) {  
    // Loop statements  
}  
// Next statement
```



for Loop

```
for (initialization; conditions; iteration)
{
    statements;
}
```

- Program 5.3

Loops and Variable Scope

```
int count = 10;
```

```
// Calculate the sum of integers 1 to count
```

```
int count = 10;
```

```
long sum = 0;
```

```
for(int i = 1 ; i <= count ; i++)
```

```
    sum += i;
```

```
// Calculate the product of integers 1 to count
```

```
long product = 1;
```

```
for(int i = 2 ; i <= count ; i++)
```

```
    product *= i;
```

Controlling a for Loop with Floating-points Values

```
const double pi = 3.14159265;  
for (double radius=2.5; radius<=20.0;  
    radius+=2.5)  
    cout << "radius = " << setw(12) << radius  
        << " area =" << setw(12)  
        << pi*radius *radius << endl;
```

- Program 5.4, 5.5

Multiple Initializations in A Loop Expression

- for (initialized expressions; conditions; expressions)
- E.g.
 - for (int j=0, k=2, product=1; k<count; j++, k++)
- Program 5.6, 5.7


The comma operator

```
int i = 1;  
int value1 = 1;  
int value2 = 1;  
int value3 = 1;  
value1 += ++i, value2 += ++i, value3 += ++i;
```

Nested Loop

- Using a nested loop to generate multiplication tables
- Program 5.8

continue Statement



```
while (cin.get(ch))  
{  
    statement1;  
    if (ch == '\n')  
        continue;  
    statement2;  
}
```

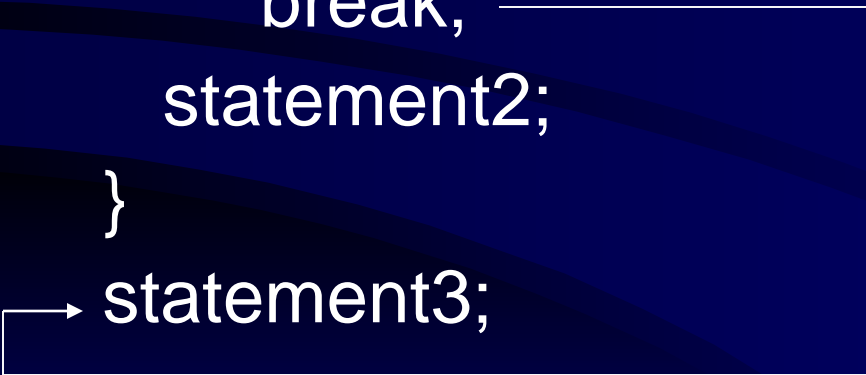
- Program 5.9

Indefinite Loops

- `for (;;) { statements; }`
- `while (true) { statements; }`
- Using the `break` statement to terminate the indefinite loop

break Statement

```
while (cin.get(ch))  
{  
    statement1;  
    if (ch == '\n')  
        break;  
    statement2;  
}  
→ statement3;
```



- Program 5.10, 5.11