# Introduction to Repetition Statements

## Syntaxes of repetition statements

Q: [What are three types of repetition statements in C++?](#three_repetition_statements)

Q: [What is syntax of while-statement?](#syntax_while_statement)

Q: [What is syntax of for-statement?](#syntax_for_statement)

Q: [What is syntax of do-while statement?](#syntax_do_while_statement)

## Repetition Statement Details

Q: [Imagine conditions in repetition statements as a door guard?](#door_guard)

Q: [What is minimum and maximum number of times of running loop-body of while- or for-statement?](#min_max_num_for_while_statement) Q: [What is minimum and maximum number of times of running loop-body of do-while statement?](#min_max_num_do_while_statement)

Q: [What to put in loop body?](#what_in_loop_body)

## Relationship among Repetition Statements?

Q: [When do we need to use repetition statement?](#when_to_use_repetition_statement)

Q: [When to use for-statement?](#when_to_use_for_statement)

Q: [When do we need to use do-while statement?](#when_to_use_do_while_statement)

Q: [Can while-statement rewritten as do-while statement?](#while_statement_to_do_while_statement)

Q: [Can do-while statement rewritten as while-statement?](#do_while_statement_to_while_statement)

## Common Mistakes of Repetition Statements

Q: [Can we add a semicolon right after while- or for-head?](#add_semicolon_right_after_while_head)

Q: [Can we omit ; in a do-while statement?](#add_semicolon_do_while_statement)

Q: [What if we forget to update loop-variable in loop?](#forget_to_update_loop_variable)

## Track Repetition Statement

Q: [Given a repetition statement, how to “spread out” to test its running?](#tabular_illustration_loop)

**Q:** **What are three types of repetition statements in C++?**

A: while-, for-, and do-while statements.

**Q:** **What is syntax of while-statement?**

A: while-statement has three parts: initialization, while-head, and while-body.

initialization loop variable;

while (condition)

{

statements-to-repeat;

}

1. Loop variable is used in condition and/or statements inside the loop to decide when to stop.
2. Initialization of loop variable is not formally a part of while-statement, but its importance cannot be omitted. The initial value of loop variable decides whether the loop is entered and/or how many rounds statements-to-repeat will run.
3. Keyword while followed by condition is called while-head. The condition must be enclosed in a pair of parentheses.
4. Statements-to-repeat is also called loop body. If loop body has only one statement, curly braces { and } are NOT needed; otherwise, use a pair of { and } to enclose loop-body (it is like to use a rubber band to tie statements together).
5. while-head and while-body (i.e., loop-body) together make a while-statement. Do not put semicolon between while-head and while-body unless you are sure that while-body is an empty statement, that is, do nothing in loop body.

**Q:** **What is syntax of for-statement?**

A: for-statement is another way of writing repetition statement. Unlike while-statement, for-statement formally introduce initialization of loop variable and update loop variable as part of for-heads. Here is its syntax.

for (initialize loop variable; condition; update loop variable)

{

statements-to-repeat;

}

The above for-statement can be rewritten as while-statement as follows; however, update of loop variable must run after statements-to-repeat. In while-statement, update of loop variable can be anywhere (beginning, middle, or bottom) of the loop.

initialize loop variable;

while (condition)

{

statements-to-repeat;

update loop variable;

}

**Q:** **What is syntax of do-while statement?**

A: The syntax of do-while statement is as follows.

do {

statements-to-repeat;

} while (condition);

Warning: semicolon (;) ends do-while statement and cannot be omitted.

The main difference between while- (also for-) versus do-while statement is, when entering loop **for the first time**, condition is needed or not.

* Condition in while-statement is located in entry point, only when condition is true can you enter the loop.
* Condition in do-while statement is located in exit-point, first time entry is free.

The common things among all repetition statements are:

* Once you enter the loop, you are not free to go: after finishing one round, you must check condition to decide whether to stay inside the loop or not.
* Once we exit the loop, the condition must be violated.

**Q: Imagine conditions in repetition statements as a door guard?**

A: Imagine condition as a door guard.

* In while- or for-statements, if condition is true, then enter the room, run codes in loop body. In do-while statement, first time entry is free.
* Once you enter the loop, exit is not free. After one round, come back to door guard again (i.e., re-evaluate condition) to see whether you can leave the room. If the condition is still true, then you stay inside the room to run one more round of codes in loop body, afterwards, come back to the door guard to check whether you can leave again …
* Hopefully, sooner or later, the condition becomes false. Then you can leave the room (i.e., exit the loop) and run the statement immediately following the loop.
  + Hence, loop body should include statement to change loop variable; otherwise, once in a loop, forever in the loop (i.e., endless loop).

**Q: What is minimum and maximum number of times of running loop-body of while- or for-statement?**

A: In while- or for-statements, the minimum number of time of running loop-body is 0, the maximum number of times of running loop-body is infinity.

**Q: What is minimum and maximum number of times of running loop-body of do-while statement?**

A: In do-while statement, the minimum number of time of running loop-body is 1, the maximum number of times of running loop-body is infinity.

**Q: What to put in loop body?**

A: Things to be repeated are put inside a loop.

**Q: When to use repetition statement?**

A: If we know that the statement **may** run zero, one, or multiple times. For example, to ensure the input is in a certain range. Then the user may take several tries to get the correct input.

**Q: When to use for-statement?**

A: for-statement is a concise way of writing while-statement. In general, it is used in situations where number of repetitions is known in advance. For example, do something for n times, where n can be a constant (a fixed number) or an int variable.

**Q: When to use do-while statement?**

A: If we know code in loop statement must run once, then we can use do-while statement. For example, enter an integer until it is in [0, 100] – we need to input at least once. Or find out the number of digits of an int, we know each number has at least a digit, so we can keep on throwing away the least significant digit until no digit is left to be thrown.

**Q: Can while-statement rewritten as do-while statement?**

A: No. The loop body of a while-statement may not run at all. But do-while statement needs to run loop statement at least once.

For example, in the following while-statement, loop body is not run at all. Hence, it cannot be rewritten as a do-while statement.

int num = 5;

    while (num < 0)

    {

        cout << num << endl;

        num++;

    }

**Q: Can do-while statement rewritten as while-statement?**

A: In general, yes. Take out one running of loop statement in do-while statement before the loop.

For example, the left side is do-while statement, and the right side is while-statement.

|  |  |
| --- | --- |
| do {  statements-to-repeat;  } while (condition); | statements-to-repeat;  while (condition)  {  statements-to-repeat;  } |

**Q:** **Can we add a semicolon right after while- or for-head?**

A: Normally we do notput asemicolon (;)right after while- or for-head unless we are absolutely sure that we do not need to do anything in loop-body.

**Q: Can we omit ; in a do-while statement?**

A: Absolutely not. A do-while statement is a statement and must be ended by a semicolon. It looks like that the semicolon is added right after the condition, however, unlike while- or for-head, the condition in do-while statement is at its exit point.

**Q:** **What if we forget to update loop-variable in loop?**

A: If we do not enter the loop (due to the initial value of loop variable fails to make condition true), it will be ok. However, once we enter the loop, there is no way to exit. In that situation, dead loop (also called endless loop) is encountered.

**Q:** **Given a repetition statement, how to “spread out” to test its running?**

A: For example, we have the following code, what is the output?

#include<iostream>

using namespace std;

int main()

{

    int n = 1729;

    int sum = 0;

    while (n > 0) {

       int digit = n % 10;

       sum = sum + digit;

       n = n / 10;

    }

    cout << sum << endl;

    return 0;

}

1. Initialize n to be 1729 and sum to be 0.
2. Is (n > 0)? Yes, Enter loop.

(1a) digit = n % 10; where % is remainder operator. Expression n % 10 finds out the remainder of n divided by 10. That is, divide 1729 pens among 10 students, each student gets the same number of pens (that is, 172), there is 9 pens left. So digit is 9, which is the least significant digit of number 1729.

(1b) sum = sum + digit; means to add digit to sum. Previously the value of sum is 0, after this statement, sum is changed to 9.

(1c) n = n / 10; That is, use the quotient of n divided by 10 to reset n. Since n is an int and 10 is also an int, divide n by 10 is integer division, that is, the result quotient is also an integer, this is done by omitting decimal part of n / 10 and keep only the whole number. Since n is 1729, the result of n / 10 is 172 (not 172.9, why?). After assignment statement n = n / 10; the value of n is 172.

(1d) Return to while-head.

1. Is n larger than 0? Yes. Enter the loop.

(2a) digit = n % 10; since n is 172, digit is set to be the least significant digit of n, which is 2.

(2b) sum = sum + digit; Before update, the value of sum is 9, after the update, the value of sum is 9 + 2 = 11.

(2c) n = n / 10; The value of n before update is 172, after the update, n is 17.

(2d) Return to while-head.

1. Is n larger than 0? Yes. Enter the loop.

(3a) digit = n % 10; since n is 17, digit is 17 % 10, which is 7.

(3b) sum = sum + digit; After this statement, sum is 11 + 7, which is 18.

(3c) n = n / 10; After this statement, n is 1.

(3d) Return to while-head.

1. Is n larger than 0? Yes. Enter the loop.

(4a) digit = n % 10; since n is 1, digit is 1 % 10, which is 1.

(4b) sum = sum + digit; After this statement, sum is 18 + 1 = 19.

(4c) n = n / 10; If we have one pen, divided among 10 students, each student get 0 pen. So, after this statement, n is 0.

(4d) Return to while-head.

1. Is n larger than 0? No. Stop the loop.
2. Go to the next statement following the loop, which is to print the current value of sum, which is 19, to the screen.

First initialize variables before the loop. These statements run only once. Variable n is used in condition, so n is loop variable.

sum = 0;

n = 1729;

The following is a tabular format to illustrate running of loop statement. Each row in the table represent a round of loop.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Loop variable n | Condition (n > 0) | Loop body | | |
| digit = n % 10 | sum += digit; | n /= 10; |
| 1729 | Yes | 9 | 0 + 9 = 9 | 172 |
| 172 | Yes | 2 | 9 + 2 = 11 | 17 |
| 17 | Yes | 7 | 11 + 7 = 18 | 1 |
| 1 | Yes | 1 | 18 + 1 = 19 | 0 |
| 0 | NO | Stop the loop. | | |

After the loop, print out the value of sum, which is 19, to the screen.

## Examples of repetition statements

Q: Starting with $10,000, how many years until we have at least $20,000, at 5% interest?

Q: How to enter salaries (-1 to stop), then calculate their average?

Q: Enter a series of integers from console, calculate their total and average.

Q: How to ensure a user enter an integer in [0, 100]?

Q: Given a string, how to construct a string that contains only even-index letters?

Q: Count number of characters in a string.

Q: Counting words in a user input sequence.

Q: Find the location in a string of first space char.

Q: Find the min and max values of user input list (related to Lab 3B).

Q: Remove all dashes or spaces in a string. For example, if you enter your phone number as “123-456-7890”, the output is “1234567890”.

Q: Print out a table of power as follows.

