

# Looping

Victor Eijkhout, Susan Lindsey

Fall 2025

last formatted: September 2, 2025

# Reference material

The following slides are a high-level introduction;  
for details see: chapter Textbook, section 6 upto Textbook,  
section 6.4

## For loops

# 1. 'For' statement

Sometimes you need to repeat a statement a number of times. That's where the loop comes in. A loop has a counter, called a loop variable, which (usually) ranges from a lower bound to an upper bound.

Here is the syntax in the simplest case:

```
1 // loop/sumsquares.cpp
2 int sum_of_squares{0};
3 for (int var=low; var<upper; ++var) {
4     sum_of_squares += var*var;
5 }
6 cout << "The sum of squares from "
7     << low << " to " << upper
8     << " is " << sum_of_squares << '\n';
```

## 2. Loop variable

Old-style loop declaration:

```
1 int i;  
2 for ( i=0; i<N; ++i )  
3    // stuff
```

- Variable should be local to a loop (legal syntax since C99)
- (Other reasons too)

### 3. Loop syntax: variable

The loop variable is usually an integer:

```
1 for ( int index=0; index<max_index; index=index+1) {  
2     ...  
3 }
```

But other types are allowed too:

```
1 for ( float x=0.0; x<10.0; x+=delta ) {  
2     ...  
3 }
```

Beware the stopping test for non-integral variables!

## 4. Loop syntax: test

- If this boolean expression is true, do the next iteration.
- Done before the first iteration too!

```
for ( int i=0; i<N; ++i ) {...}
```

- Test can be empty. This means no test is applied.

```
for ( int i=0; ; ++i ) {...}
```

## 5. Loop syntax: increment

Increment performed after each iteration. Most common:

- `i++` or `++i` for a loop that counts forward;
- `i--` or `--i` for a loop that counts backward;

Others:

- `i+=2` to cover only odd or even numbers, depending on where you started;
- `i*=10` to cover powers of ten.

Even optional:

```
1 for (int i=0; i<N; ) {  
2     // stuff  
3     if ( something ) i+=1; else i+=2;  
4 }
```



# Quiz 1

For each of the following loop headers, how many times is the body executed? (You can assume that the body does not change the loop variable.)

```
for (int i=3; i<10; i++)
```

```
for (int i=3; i<=10; i++)
```

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

# Programming Project Exercise 1

Read an integer and set a boolean variable to determine whether it is prime by testing for the smaller numbers if they divide that number.

Print a final message

`Your number is prime`

or

`Your number is not prime: it is divisible by ....`

where you report just one found factor.

## 6. Nested loops

Traversing a matrix

(we will discuss actual matrix data structures later):

```
1 for (int row=0; row<m; row++)  
2   for (int col=0; col<n; col++)  
3     ...
```

These are called 'nested loops', with

*row*: outer loop

*col*: inner loop.

## 7. Indefinite looping

Sometimes you want to iterate some statements not a predetermined number of times, but until a certain condition is met. There are two ways to do this.

First of all, you can use a 'for' loop and leave the upper bound unspecified:

```
1 for (int var=low; ; var=var+1) { ... }
```

## Exercise 2

The integer sequence

$$u_{n+1} = \begin{cases} u_n/2 & \text{if } u_n \text{ is even} \\ 3u_n + 1 & \text{if } u_n \text{ is odd} \end{cases}$$

leads to the Collatz conjecture: no matter the starting guess  $u_1$ , the sequence  $n \mapsto u_n$  will always terminate at 1.

$$5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

$$7 \rightarrow 22 \rightarrow 11 \rightarrow 34 \rightarrow 17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \dots$$

(What happens if you keep iterating after reaching 1?)

Try all starting values  $u_1 = 1, \dots, 1000$  to find the values that lead to the longest sequence: every time you find a sequence that is longer than the previous maximum, print out the starting number.

## Breaking out of a loop

# Reference material

The following slides are a high-level introduction;  
for details see: section Textbook, section 6.3

## 8. Break out of a loop

This loop would run forever, so you need a different way to end it.  
For this, use the `break` statement:

```
1 for (int var=low; ; var=var+1) {  
2     statement;  
3     if (some_test) break;  
4     statement;  
5 }
```



## 9. Where did the break happen?

Suppose you want to know what the loop variable was when the break happened. You need the loop variable to be global:

```
1 int var;  
2 ... code that sets var ...  
3 for ( ; var<upper; var++) {  
4     ... statements ...  
5     if (some condition) break  
6     ... more statements ...  
7 }  
8 ... code that uses the breaking value of var ...
```

In other cases: define the loop variable in the loop header!

## 10. Test in the loop header

If the test comes at the start or end of an iteration, you can move it to the loop header:

```
1 bool need_to_stop{false};  
2 for (int var=low; !need_to_stop ; var++) {  
3     ... some code ...  
4     if ( some condition )  
5         need_to_stop = true;  
6 }
```

# 11. Skip iteration

```
1 for (int var=low; var<N; var++) {  
2     statement;  
3     if (some_test) {  
4         statement;  
5         statement;  
6     }  
7 }
```

Alternative:

```
1 for (int var=low; var<N; var++) {  
2     statement;  
3     if (!some_test) continue;  
4     statement;  
5     statement;  
6 }
```

The only difference is in layout.

## Nested loops

## Exercise 3

Write  $i, j$  nested loops that print out all pairs with

$$1 \leq i, j \leq 10, \quad j \leq i.$$

Output one line for each  $i$  value.

Now write an  $i, j$  loop that prints all pairs with

$$1 \leq i, j \leq 10, \quad |i - j| < 2,$$

again printing one line per  $i$  value. Food for thought: this exercise is definitely easiest with a conditional in the inner loop, but can you do it without?

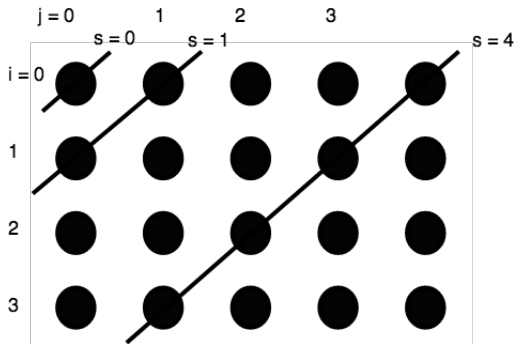
## Exercise 4

Write a double loop over  $0 \leq i, j < 10$  that prints the first pair where the product of indices satisfies  $i \cdot j > N$ , where  $N$  is a number your read in. A good test case is  $N = 40$ .

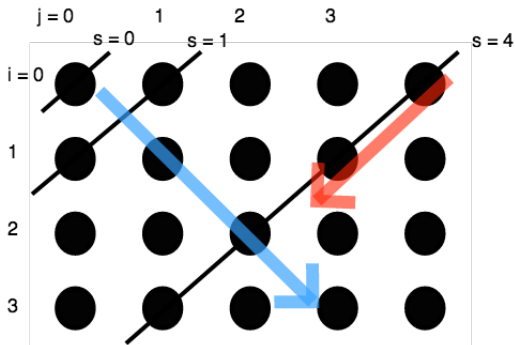
Secondly, find a pair with  $i \cdot j > N$ , but with the smallest value for  $i + j$ . (If there is more than one pair, report the one with lower  $i$  value.) Can you traverse the  $i, j$  indices such that they first enumerate all pairs  $i + j = 1$ , then  $i + j = 2$ , then  $i + j = 3$  et cetera? Hint: write a loop over the sum value  $1, 2, 3, \dots$ , then find  $i, j$ .

You program should print out both pairs, each on a separate line, with the numbers separated with a comma, for instance 8,5.

# Suggestive picture 1



## Suggestive picture 2





## While loops

# Reference material

The following slides are a high-level introduction;  
for details see: section Textbook, section 6.3.1

## 12. While loop

Syntax:

```
1 while ( condition ) {  
2   statements;  
3 }
```

or

```
1 do {  
2   statements;  
3 } while ( condition );
```

## 13. Pre-test while loop

```
1 float money = inheritance();  
2 while ( money < 1.e+6 )  
3     money += on_year_savings();
```

# 14. While syntax 1

Code:

```
1 // basic/whiledo.cpp
2 cout << "Enter a positive number: "
  ;
3 cin >> invar; cout << '\n';
4 cout << "You said: " << invar << '\n'
  ;
5 while (invar<=0) {
6   cout << "Enter a positive number: "
    ;
7   cin >> invar; cout << '\n';
8   cout << "You said: " << invar <<
    '\n';
9 }
10 cout << "Your positive number was "
11      << invar << '\n';
```

Output:

```
1 Enter a positive
   ↪number:
2 You said: -3
3 Enter a positive
   ↪number:
4 You said: 0
5 Enter a positive
   ↪number:
6 You said: 2
7 Your positive number
   ↪was 2
```

Problem: code duplication.

## 15. While syntax 2

Code:

```
1 // basic/dowhile.cpp
2 int invar;
3 do {
4     cout << "Enter a positive number:
      " ;
5     cin >> invar; cout << '\n';
6     cout << "You said: " << invar <<
      '\n';
7 } while (invar<=0);
8 cout << "Your positive number was: "
9     << invar << '\n';
```

Output:

```
1 Enter a positive
   ↪number:
2 You said: -3
3 Enter a positive
   ↪number:
4 You said: 0
5 Enter a positive
   ↪number:
6 You said: 2
7 Your positive number
   ↪was: 2
```

The post-test syntax leads to more elegant code.

## Optional exercise 5

A horse is tied to a post with a 1 meter elastic band. A spider that was sitting on the post starts walking to the horse over the band, at 1cm/sec. This startles the horse, which runs away at 1m/sec. Assuming that the elastic band is infinitely stretchable, will the spider ever reach the horse?