

# Looping

Victor Eijkhout, Susan Lindsey

COE 322 Fall 2021

## 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:

```
int sum_of_squares{0};
for (int var=low; var<upper; var++) {
    sum_of_squares += var*var;
}
cout << "The sum of squares from "
      << low << " to " << upper
      << " is " << sum_of_squares << endl;
```

## 2. Loop syntax: variable

The loop variable is usually an integer:

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

But other types are allowed too:

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

Beware the stopping test for non-integral variables!

### 3. Loop syntax: test

- If this test is true, do the next iteration.
- Done before the first iteration too!
- Test can be empty. What does this mean?

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

## 4. Loop syntax: increment

Increment performed after each iteration. Most common:

- `i++` for a loop that counts forward;
- `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:

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

# Review 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=0; i<7; i++)
```

```
/poll "for (int i=0; i<7; i++)" "6 iterations" "7" "8"
```

```
for (int i=0; i<=7; i++)
```

```
/poll "for (int i=0; i<=7; i++)" "6 iterations" "7" "8"
```

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

```
/poll "for (int i=0; i<0; i++)" "0 iterations" "1" "inf"
```

# Review quiz 2

What is the last iteration executed?

```
for (int i=1; i<=2; i=i+2)
```

```
/poll "for (int i=1; i<=2; i=i+2) last iteration" "i=1" "i=2" "i=3" "i=4"
```

```
for (int i=1; i<=5; i*=2)
```

```
/poll "for (int i=1; i<=5; i*=2) last iteration" "4" "5" "8"
```

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

```
/poll "for (int i=0; i<0; i--) last iteration" "none" "0" "-1" "-inf"
```

```
for (int i=5; i>=0; i--)
```

```
/poll "for (int i=5; i>=0; i--) last iteration" "0" "1" "-1" "4"
```

```
for (int i=5; i>0; i--)
```



# Exercise 1

Take this code:

```
int sum_of_squares{0};
for (int var=low; var<upper; var++) {
    sum_of_squares += var*var;
}
cout << "The sum of squares from "
      << low << " to " << upper
      << " is " << sum_of_squares << endl;
```

and modify it to sum only the squares of every other number, starting at *low*.

Can you find a way to sum the squares of the even numbers  $\geq low$ ?

## Project Exercise 2

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.

## 5. Nested loops

Traversing a matrix

(we will discuss actual matrix data structures later):

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

This is called 'loop nest', with

*row*: outer loop

*col*: inner loop.

## 6. 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 upperbound unspecified:

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

## 7. 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:

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

## Exercise 3

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.

## 8. 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:

```
int var;
... code that sets var ...
for ( ; var<upper; var++) {
    ... statements ...
    if (some condition) break
    ... more statements ...
}
... code that uses the breaking value of var ...
```

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

## 9. 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:

```
bool need_to_stop{false};  
for (int var=low; !need_to_stop ; var++) {  
    ... some code ...  
    if ( some condition )  
        need_to_stop = true;  
}
```



## Exercise 4

Write an  $i, j$  loop nest that prints 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?

## Optional exercise 5

Find all triples of integers  $u, v, w$  under 100 such that  $u^2 + v^2 = w^2$ . Make sure you omit duplicates of solutions you have already found.

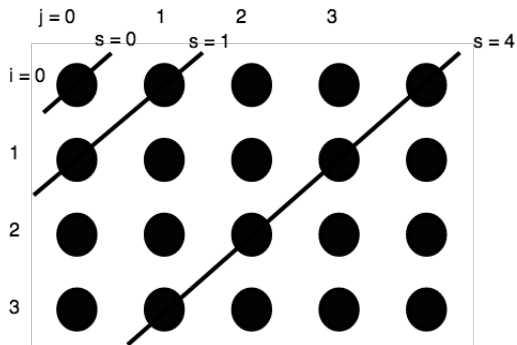
## Exercise 6

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  $N = 40$ .

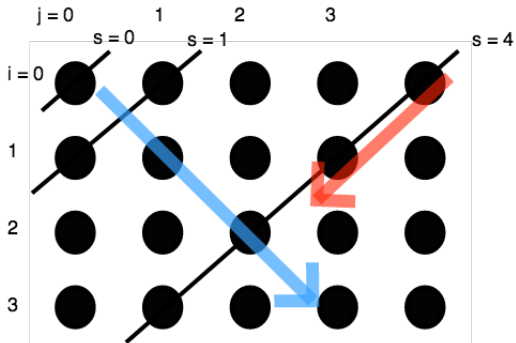
Secondly, find a pair with  $i \cdot j > N$ , but with the smallest value for  $i + j$ . (If there is more than one pair, report the one with lower  $i$  value.) Can you traverse the  $i, j$  indices such that they first enumerate all pairs  $i + j = 1$ , then  $i + j = 2$ , then  $i + j = 3$  et cetera?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

Your 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



# Turn it in!

- If you have compiled your program, do:

```
coe_ij yourprogram.cc
```

where 'yourprogram.cc' stands for the name of your source file.

- Is it reporting that your program is correct? If so, do:

```
coe_ij -s yourprogram.cc
```

where the -s flag stands for 'submit'.

## 10. Skip iteration

```
for (int var=low; var<N; var++) {  
    statement;  
    if (some_test) {  
        statement;  
        statement;  
    }  
}
```

Alternative:

```
for (int var=low; var<N; var++) {  
    statement;  
    if (!some_test) continue;  
    statement;  
    statement;  
}
```

The only difference is in layout.

# While loops



# 11. While loop

The other possibility for 'looping until' is a *while* loop, which repeats until a condition is met.

Syntax:

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

or

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

The while loop does not have a counter or an update statement; if you need those, you have to create them yourself.

## 12. Pre-test while loop

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

## 13. While syntax 1

Code:

```
cout << "Enter a positive number: " ;
cin >> invar; cout << endl;
cout << "You said: " << invar << endl;
while (invar<=0) {
    cout << "Enter a positive number: "
        ;
    cin >> invar; cout << endl;
    cout << "You said: " << invar <<
        endl;
}
cout << "Your positive number was "
    << invar << endl;
```

Output

```
[basic] whiledo:

Enter a positive
number:
You said: -3
Enter a positive
number:
You said: 0
Enter a positive
number:
You said: 2
Your positive number
was 2
```

Problem: code duplication.

## 14. While syntax 2

Code:

```
int invar;
do {
    cout << "Enter a positive number: "
        ;
    cin >> invar; cout << endl;
    cout << "You said: " << invar <<
        endl;
} while (invar<=0);
cout << "Your positive number was: "
    << invar << endl;
```

Output

```
[basic] dowhile:

Enter a positive
number:
You said: -3
Enter a positive
number:
You said: 0
Enter a positive
number:
You said: 2
Your positive number
was: 2
```

The post-test syntax leads to more elegant code.

## Optional exercise 7

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?