4.4 For loops

Basics

A loop commonly must iterate a specific number of times, such as 10 times. Though achievable with a while loop, that situation is so common that a special kind of loop exists. A **for loop** is a loop with three parts at the top: a loop variable initialization, a loop expression, and a loop variable update. A for loop describes iterating a specific number of times more naturally than a while loop.

```
Construct 4.4.1: For loop.

for (initialExpression; conditionExpression; updateExpression) {
    // Loop body
}
// Statements after the loop
Feedback?
```

```
PARTICIPATION
                 4.4.1: For loops.
ACTIVITY
            int i;
                                                  int i;
            i = 0;
                                                  for ( i = 0; i < 5; i = i + 1 ) {
            while (i < 5) {
                                                     // Loop body
               // Loop body
                i = i + 1;
            i: 0 (Iterates)
                1 (Iterates)
                                 5 iteratons
                2 (Iterates)
                3 (Iterates)
                4 (Iterates)
                5 (Does not iterate)
```

Note that semicolons separate the three parts. No semicolon is needed at the end.

Feedback?

The statement i = i + 1 is so common that the language supports the shorthand ++i, with ++i known as the **increment operator**. (Likewise, --i is the **decrement operator**, --i means i = i - 1). As such, a standard way to loop N times is shown below.

Figure 4.4.1: A standard way to loop N times, using a for loop.

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

Feedback?

PARTICIPATION ACTIVITY

4.4.2: For loops.

1) What are the values of i for each iteration of:

```
for (i = 0; i < 6; ++i) {
    ...
}</pre>
```

- **O** 1, 2, 3, 4, 5
- 0, 1, 2, 3, 4, 5
- **O** 0, 1, 2, 3, 4, 5, 6
- 2) How many times will this loop iterate?

```
for (i = 0; i < 8; ++i) {
    ...
}</pre>
```

- O 7 times
- O 8 times
- O 9 times

3) Goal: Loop 10 times

```
for (i = 0; ____; ++i) {
 0 i < 9
 O i < 10
 O i < 11
```

4) Goal: Loop 99 times

```
for (i = 0; ____; ++i) {
 O i < 99
```

- O i <= 99
- 0 i == 99

5) Goal: Loop 20 times

```
for (____; i < 20; ++i) {</pre>
  O = i
```

Oi = 1

6) Goal: Loop numYears times (numYears is an int variable).

```
for (i = 0; ____; ++i) {
```

- O numYears
- O i <= numYears
- O i < numYears

Feedback?

PARTICIPATION ACTIVITY

4.4.3: For loops.

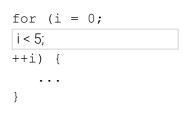


Write for loops using the following form:

for
$$(i = 0; i < 10; ++i)$$
 {

Note: Using i = 1, $\leq i + 1$, or a variable other than i, are not accepted by this activity.

1) Complete the for loop to iterate 5 times. (Don't forget the semicolon).



Correct



The loop will iterate with i of 0, 1, 2, 3, and 4, so 5 iterations.

Check

Show answer

2) Complete the for loop to iterate 7 times.

```
for (
    i = 0; i < 7;
++i) {
        ...
}</pre>
```

Correct



The loop will iterate with i of 0, 1, 2, 3, 4, 5, and 6, so 7 iterations.

Check

Show answer

3) Complete the for loop to iterate 500 times. (Don't forget the parentheses).

```
for (i = 0; i < 500; ++i) {
...
}
```

Correct



The loop will iterate with i of 0, 1, ..., 488, and 499, which is 500 iterations.

Check

Show answer

4) Complete the for loop to iterate numDogs times. numDogs is an int variable.

Check

Show answer





i < numDogs;

Correct

The loop will iterate with i of 0, 1, ..., numDogs-1. So if numDogs is 5, the loop will iterate with i as 0, 1, 2, 3, and 4 (so 5 times).

Feedback?

Note: Actually two increment operators exist: ++i (**pre-increment**) and i++ (**post-increment**). ++i increments before evaluating to a value, while i++ increments after. Ex: If i is 5, outputting ++i outputs 6, while outputting i++ outputs 5 (and then i becomes 6). This material primarily uses ++i for simplicity and safety, although many programmers use i++, especially in for loops.

Example: Savings with interest

The following program outputs the amount of a savings account each year for 10 years, given an input initial amount and interest rate. A for loop iterates 10 times, such that each iteration represents one year, outputting that year's savings amount.

Figure 4.4.2: For loop: Savings interest program.

```
#include <iostream>
using namespace std;
int main() {
   double initialSavings; // User-entered initial savings
   double interestRate; // Interest rate
   double currSavings;
                           // Current savings with interest
                           // Loop variable
   int i;
   cout << "Enter initial savings: ";</pre>
   cin >> initialSavings;
   cout << "Enter interest rate: ";</pre>
   cin >> interestRate;
   cout << endl << "Annual savings for 10 years: " << endl;</pre>
   currSavings = initialSavings;
   for (i = 0; i < 10; ++i) {
      cout << "$" << currSavings << endl;</pre>
      currSavings = currSavings + (currSavings * interestRate);
   return 0;
```

```
Enter initial savings: 10000
Enter interest rate: 0.05

Annual savings for 10 years: $10000 $10500 $11025 $11576.2 $12155.1 $12762.8 $13401 $14071 $14774.6 $15513.3
```

Feedback?

PARTICIPATION ACTIVITY

4.4.4: Savings interest program.



Consider the example above.

1) How many times does the for loop iterate?

O 5

10

2) During each iteration, the loop body's statements output the current savings amount, and then

O increment i

update currSavings

3) Can the input values change the number of loop iterations?

O Yes

No

Correct

The loop is:

```
for (i = 0; i < 10; ++i) {
   ...
}</pre>
```

Those values cause 10 iterations.

Correct

The loop body is:

```
cout << "$" << currSavings << endl;
currSavings = currSavings + (currSavings *
interestRate);
```

The second statement updates currSavings for the next year, by adding the interest amount. Note that ++i is in the loop header; that update implicitly occurs after the loop body.

Correct

For this loop, the number of iterations is fixed at 10. In other programs, the number of iterations may depend on input values.

Feedback?

Example: Computing the average of a list of input values

The example below computes the average of an input list of integer values. The first input indicates the number of values in the subsequent list. That number controls how many times the subsequent for loop iterates.

Figure 4.4.3: Computing an average, with first value indicating list size.

```
4 10 1 6 3
Average: 5
...
5 -75 -50 30 60 80
Average: 9
```

```
#include <iostream>
using namespace std;
// Outputs average of list of integers
// First value indicates list size
// Ex: 4 10 1 6 3 yields (10 + 1 + 6 + 3) / 4, or 5
int main() {
  int currValue;
  int valuesSum;
  int numValues;
  int i;
  cin >> numValues; // Gets number of values in list
  valuesSum = 0;
  for (i = 0; i < numValues; ++i) {
          cin >> currValue; // Gets next value in list
          valuesSum += currValue;
  cout << "Average: " << (valuesSum / numValues) << endl;</pre>
  return 0;
```

Feedback?

PARTICIPATION ACTIVITY

4.4.5: Computing the average.



Consider the example above, with input 4 10 1 6 3. Note: The first input indicates the number of values in the subsequent list.

1) Before the loop is entered, what is valuesSum?



Check

Show answer

2) What is valuesSum after the first iteration?



Check

Show answer

Correct



valuesSum = 0; appears just before the for loop.
The sum of the values seen so far is 0, because no
values have been seen so far. Such initialization
just before a loop is common.

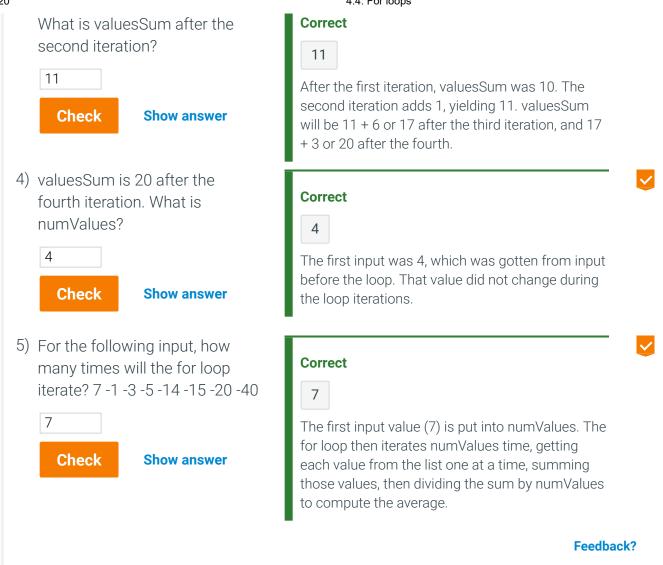
Correct

10

valuesSum was initially 0. The first iteration gets currValue (10) from input, and executes valuesSum += currValue, thus assigning valuesSum with 0 + 10 or 10.







Choosing among for and while loops

Generally, a programmer uses a for loop when the number of iterations is known (like loop 5 times, or loop numItems times), and a while loop otherwise.

Table 4.4.1: Choosing between while and for loops: General guidelines (not strict rules though).

for	Number of iterations is computable before the loop, like iterating N times.
while	Number of iterations is not (easily) computable before the loop, like iterating until the input is 'q'.

Feedback?

PARTICIPATION ACTIVITY	1.4.6: While loops and for loops.	
Choose the mos	t appropriate loop type.	
1) Iterate as long c is not 'q'.	g as user-entered char	
O for		
	ne values of x and y are x and y are changed in v.	
3) Iterate 100 tir O while O for	mes.	
		Feedback?
CHALLENGE 4.4.	.1: Enter the for loop's output.	✓
ACTIVITY 4.4		t.
ACTIVITY 4.4	1	t.

