FOR loops

1E3
Topic 7

Objectives

- To be able to write a FOR statement
- To be able to predict what a given FOR statement will do.
- To recognise when a FOR loop is appropriate
- To be able to choose between a FOR and a WHILE
Increment/Decrement

- Adding or subtracting 1 from a variable is very common, especially for loops.
- `++` increment operator
  - Adds 1 to the value of a variable
    \[ x ++ \]
    is equivalent to \[ x = x + 1; \]
- `num_years++` replaces
  \[ num_years = num_years + 1 \]

Decrement

- `--` decrement operator
  - Subtracts 1 from the value of a variable
    \[ x -- \]
    is equivalent to \[ x = x - 1; \]
- Use `countdown--` instead of
  \[ countdown = countdown - 1; \]

These operators are especially used in FOR loops.
For loops

Our first program used a FOR loop to take some action for each of the integers from 1 to an entered integer, \( n \).

```c
for (i=1; i<=n; i++)
{
    //loop body
}
```

For each value of \( i \) starting at 1, as long as \( i \) is \( \leq n \), incrementing \( i \) by 1 each time, do the loop body.

Other Examples

Do <Something> 100 times:

```c
for (lc=0; lc<100; lc++)
    <Something>
```

Note: \( lc \) must be declared!

Can also count down:

```c
for (lc=100; lc>0; lc--)
    cout << lc << "\n";
```
The for Loop

- The general form of the for statement is:

```plaintext
for (initial statement;
     loop condition;
     update statement)
statement
```

- The initial statement, loop condition, and update statement are called for loop control statements.
Which one prints the most “Hello”s?

A: for (i=0; i<10; i++)
  cout << “Hello”;

B: for (i=10; i>0; i--)
  cout << “Hello”;

C: for (i=20; i>=0; i=i-2)
  cout << “Hello”;

D: for (i= -10; i<10; i=i+2)
  cout << “Hello”;

---

Be careful

- Never put a ; after the loop control section
  - for (i=0; i<10; i++); cout << “Hello”;  
  - does nothing ten times, then prints Hello once.
- As with while loops, make sure the combination of initial value, test and update action don’t result in an infinite loop.
  - for (i=0; i<20; i=i-2) cout << “Hello”;

7 FOR Loops
WHILE vs. FOR

- for (init; cond; update) stmt;

- is equivalent to

- init;
  while (cond)
  { stmt;
    update; }

A FOR is more specialised than a WHILE

Use a FOR when the problem can be formulated as
- for each integer from i to k ...
- for each element in the list ...
- for each entry in the array ...

Use WHILE if a FOR doesn’t fit the bill.
Examples

- Compute n! (factorial) for an entered integer n.
- Print multiplication tables
  - for a given integer (e.g. 5 times tables)
  - for entered series of integers
    - Using a while loop to repeat until termination condition...
  - or for a different operator (+, -, *, ...)  
  - or all tables 1*, ..., 10*.  
    - This requires nested for loops  
  - or a triangular grid of multiplication tables.

Example: Perfect Numbers

- Write a program to check if a number is ‘perfect’ or not.
- A positive whole number is ‘perfect’ if the sum of its divisors (apart from itself) equals the number.
- First try to do this yourself  
  - Is 4 perfect?  
  - Is 6 perfect?  
- You’ll need to check each integer from 2 to n to see if it’s a divisor; if it is add it to a running total.
Other exercises

- Read an integer, then print that many *s on a line.
- Write a loop to compute sine wave 0 to $2\pi$, in steps of 0.01.
  - Use sin() from cmath

Nested Loops

- We frequently need to nest one loop inside another.
  - E.g. to average each of 200 students’ 8 exam marks.
  - E.g. to print a graph of horizontal bars of a sentinel terminated sequence of integers.
  - E.g. to play the number guessing game repeatedly
  - E.g. to test all integers from 1 to 100 for perfection
To produce 20 lines of 10 *s

```cpp
for (i=1; i<=n; i++) {
    for (j=1; j<=m; j++)
        cout << '*';
    cout << endl;
}
```

A. n=20; m=10
B. m=20; n=10
C. Neither

Print multiplication grid:

<table>
<thead>
<tr>
<th>Times</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>18</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
<td>24</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>36</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
<td>40</td>
<td>45</td>
<td>50</td>
</tr>
</tbody>
</table>

- Print header
- Print row by row, 5 rows -
- For a given row
  - Print left margin
  - For each of 10 columns print row*col
  - Print newline
Which best handles input?

A.

```cpp
for (i=1; i<=12; i++) {
    cin >> month;
    cin >> temp;
    while (temp != -50){...
        cin >> temp; }
}
```

B.

```cpp
for (i=1; i<=12; i++) {
    cin >> month;
    cin >> temp;
    while (temp != -50){...
        cin >> temp; }
}
```

C.

```cpp
for (i=1; i<=12; i++) {
    cin >> month;
    cin >> temp;
    while (temp != -50){...
        cin >> temp; }
}
```

To average each months temps?

```cpp
for (i=1; i<=12; i++) {
    cin >> month;
    cin >> temp;
    while (temp != -50){...
        cin >> temp; }
    cout << sum/count;
}
```

- Identify how and where to
- Initialise sum
- Initialise count
- Update sum
- Update count