

This version has corrected sample dialogues near the top of page 2.

Electronic copies of handouts for L02 can be found at

<http://www.enel.ucalgary.ca/People/Norman/engg233-L02-fall2004/>

for-scope-problem.cc: This program can't be compiled ...

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 2; i < 10; i *= 16)
        cout << "inside the loop: " << i << endl;
    cout << "outside the loop: " << i << endl;
    return 0;
}
```

The (less than crystal-clear) error messages from g++ are ...

```
for-scope-problem.cc: In function 'int main()':
for-scope-problem.cc:7: error: name lookup of 'i' changed for new ISO 'for'
    scoping
for-scope-problem.cc:5: error: using obsolete binding at 'i'
```

If you can't see what the problem was, it may help to replace the `for` statement with an equivalent `while` statement ...

```
#include <iostream>
using namespace std;
int main()
{
    {
        int i = 2;
        while (i < 10) {
            cout << "inside the loop: " << i << endl;
            i *= 2;
        }
    } // The scope of i ends here, so the next line has an ERROR.
    cout << "outside the loop: " << i << endl;
    return 0;
}
```

Correcting the problem is easy—if you want to use a variable after a `for` statement ends, be sure to declare the variable before the `for` statement starts as in **for-scope-fixed.cc** ...

```
#include <iostream>
using namespace std;
int main()
{
    int i = 2; // Now we can use i after the loop ends.
    for ( ; i < 10; i *= 2)
        cout << "inside the loop: " << i << endl;
    cout << "outside the loop: " << i << endl;
    return 0;
}
```

When the above program is compiled and run, the output is

```
inside the loop: 2
inside the loop: 4
inside the loop: 8
outside the loop: 16
```

Programming problem #1: Find powers of two that are close to a positive integer. Here are two example dialogues, using a *slanted bold font* for user input:

Enter a positive int: **1000**
 $2^{**}9 < 1000$ and $1000 < 2^{**}10$. [The original handout wrongly said $2^{**}10 < 1000$ and $1000 < 2^{**}11$.]

Enter a positive int: **64**
 64 is exactly $2^{**}6$.

simple-nesting.cc: The program on the left is a demonstration of nested loops (one loop inside another). The program on the right has a little mistake in typing that results in a program that makes it through the compiler but goes into an infinite loop when run.

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < i; j++)
            cout << j << " ";
        cout << "END" << endl;
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < i; i++)
            cout << j << " ";
        cout << "END" << endl;
    }
    return 0;
}
```

Programming problem #2: Use the following infinite series formula to try to compute cosines:

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots$$

(Note: $n!$, called “ n factorial”, means $1 \times 2 \times 3 \times \dots \times (n - 1) \times n$.) Here is a function that adds up some of the terms in the formula. Maybe the function would be useful if for some reason the `<cmath>` functions were not available ...

```
double cosine_series(double x, double epsilon)
{
    double sum = 0.0;
    double power, facto, term = 1.0, sign = 1.0;
    int k;
    for (k = 2; term <= -epsilon || term >= epsilon; k += 2) {
        sum += term;
        facto = 1.0;
        for (int i = 1; i <= k; i++)
            facto *= i;
        power = 1.0;
        for (int i = 1; i <= k; i++)
            power *= x;
        sign = -sign;
        term = sign * power / facto;
    }
    return sum;
}
```

Here are the results of some experiments with the function, each time using 1.0×10^{-17} for `epsilon`.

x	exact $\cos x$	function result	remark
$\pi/4$	$1/\sqrt{2}$	0.70710678118654746	good!
$\pi/3$	0.5	0.50000000000000011	good!
π	-1	-1.00000000000000000	good!
6π	1	1.00000000031566949	cause for concern ...
14π	1	-88.15482110367766211	disaster!