

Strict Standard C++

The ISO/ANSI C++ Standard was finalized in late 1998. As the standard evolved, few of the changes made by the committee required modifications to the majority of existing C++ code. One change, however, will “break” most C++ code—the addition of **namespaces**. The newest releases of most C++ compilers move the C++ Standard Library into the namespace **std**. For example, the correct name for **cout** is now **std::cout**. The programs in the course notes and textbook still use the old form, which most compilers should continue to support for the near term. Here’s one quick solution if you encounter a compiler that requires the use of **std**:

Remove the .h from each include file name and add a using directive. If present, delete a line including stdlib.h.

<i>Old</i>	<i>New</i>
#include <iostream.h> #include <fstream.h> #include <string.h> #include <stdlib.h>	#include <iostream> #include <fstream> #include <string> using namespace std;

How To Debug Any Program

The best tool for debugging a program is an interactive debugger, like the one built into Quincy and most other interactive development environments. Sometimes, however, such a tool isn’t available to you. You can always see the value of any expression by adding an output statement, then removing it later. In C++ this is usually done using the `cerr` error output statement:

```
float power(float x, int pow)
{
    float result = 1.0;

    cerr << "power(): x = " << x << " pow = " << pow << endl;

    if(pow < 0) {
        while(pow < 0) {
            result /= x;
            pow++;
        }
        cerr << "pow = " << pow << " result = " << result << endl;
    }
    else {
        while(pow > 0) {
            result *= x;
            pow--;
        }
        cerr << "pow = " << pow << " result = " << result << endl;
    }

    cerr << "power() returning " << result << endl;
    return(result);
}
```

Notice how I start all my `cerr` debugging statements at the far left. This is so they stand out and I’m reminded to remove them later, once I’m comfortable that my program is working correctly. A programmer leaving debugging output statements in a program is like a surgeon forgetting a clamp inside a patient.

Exercise 1: power.cpp

1. Type in the program below.
2. Get it to compile and run on your computer.
3. Set a breakpoint at the beginning of `main()` and use the debugging features of your compiler to watch the values of the variables in `main()` and in `power()` as you step through the program.

```
#include <iostream.h>

float power(float x, int pow);

void main()
{
    float answer;

    answer = power(10.26,5);
    cout << "10.26 to the 5th power is " << answer << endl;

    answer = power(9.9,-4);
    cout << "9.9 to the power of -4 is " << answer << endl;

    answer = power(4.2,0);
    cout << "4.2 to the power of 0 is " << answer << endl;
}

float power(float x, int pow)
{
    float result = 1.0;

    if(pow < 0) {
        while(pow < 0) {
            result /= x;
            pow++;
        }
    }
    else {
        while(pow > 0) {
            result *= x;
            pow--;
        }
    }

    return(result);
}
```

Additional Homework Exercises

- **integers.cpp** Write a program to print the whole numbers from 0 to 20, one per line.
- **powers.cpp** Using the routine `power()`, write a program that prints the whole numbers from 0 to 20 along with their 2nd, 3rd, and 4th powers.
- **fibonacci.cpp** Write a program to print the first 20 numbers in the Fibonacci series (1, 1, 2, 3, 5, 8, 13, etc.) Start with 1 and 1, then each number is the sum of the previous two numbers.
- Examples of the output and answers at:
<http://www.nsi.edu/users/mercurio/exercises>