

Simple C++ Programs

July 1, 2010

Calculate the Volume of a Box

preprocessor directives

```
int main()  
{
```

variable
declarations

statements

```
}
```

```
//  
// This program computes the volume of a box  
//  
  
#include <cstdlib>  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    /* Declare and initialize variables */  
  
    double length(20.75), width(11.5);  
    double height = 9.5;  
    double volume;  
  
    /* Calculate volume. */  
    volume = length * width * height;  
    /* Print the volume. */  
    cout << "The volume is " << volume << endl;  
  
    system("PAUSE");  
    // Exit program.  
    return 0;  
}
```

Calculate the Volume of a Box

preprocessor directives

```
int main()  
{
```

variable
declarations

statements

```
}
```

```
//  
// This program computes the volume of a box  
//  
  
#include <cstdlib>  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    /* Declare and initialize variables */  
  
    double length(20.75), width(11.5);  
    double height = 9.5;  
    double volume;  
  
    /* Calculate volume. */  
    volume = length * width * height;  
    /* Print the volume. */  
    cout << "The volume is " << volume << endl;  
  
    system("PAUSE");  
    // Exit program.  
    return 0;  
}
```

Calculate the Volume of a Box

preprocessor directives

```
int main()  
{
```

variable
declarations

statements

```
}
```

```
//  
// This program computes the volume of a box  
//  
  
#include <cstdlib>  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    /* Declare and initialize variables */  
  
    double length(20.75), width(11.5);  
    double height = 9.5;  
    double volume;  
  
    /* Calculate volume. */  
    volume = length * width * height;  
    /* Print the volume. */  
    cout << "The volume is " << volume << endl;  
  
    system("PAUSE");  
    // Exit program.  
    return 0;  
}
```

Calculate the Volume of a Box

preprocessor directives

```
int main()  
{
```

variable
declarations

statements

```
}
```

```
//  
// This program computes the volume of a box  
//  
  
#include <cstdlib>  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    /* Declare and initialize variables */  
  
    double length(20.75), width(11.5);  
    double height = 9.5;  
    double volume;  
  
    /* Calculate volume. */  
    volume = length * width * height;  
    /* Print the volume. */  
    cout << "The volume is " << volume << endl;  
  
    system("PAUSE");  
    // Exit program.  
    return 0;  
}
```

Calculate the Volume of a Box

```
/* Comment */
```

```
// Comment
```

```
/* Multiple  
Line  
Comment */
```

```
// Multiple
```

```
// Line
```

```
// Comment
```

```
//  
// This program computes the volume of a box  
//  
  
#include <cstdlib>  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    /* Declare and initialize variables */  
  
    double length(20.75), width(11.5);  
    double height = 9.5;  
    double volume;  
  
    /* Calculate volume. */  
    volume = length * width * height;  
    /* Print the volume. */  
    cout << "The volume is " << volume << endl;  
  
    system("PAUSE");  
    /* Exit program.  
    return 0;  
}
```

Why Indent?

```
1 #include <cstdlib>
2 #include <iostream>
3 using namespace std; int main()
4 { double length(20.75), width(11.5); double height = 9.5; double
5 volume; volume = length * width * height; cout << "The volume is " <<
6 volume ; cout << " units cubed." << endl; system("PAUSE"); return 0; }
```

Why Comment?

```
1
2
3
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9
10     double a(-1.2), b(2.1);
11
12     double lhs , rhs ;
13
14
15     rhs = (1.0/3.0)*pow(b,3);
16     lhs = (1.0/3.0)*pow(a,3);
17
18
19     cout << ( rhs - lhs ) << endl;
20     return 0;
21 }
```

Why Comment?

```
1
2
3
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9
10     double a(-1.2), b(2.1);
11
12     double lhs , rhs;
13
14
15     rhs = (1.0/3.0)*pow(b,3);
16     lhs = (1.0/3.0)*pow(a,3);
17
18
19     cout << ( rhs - lhs ) << endl;
20     return 0;
21 }
```

```
1
2
3
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9     // define endpoints
10    double a(-1.2), b(2.1);
11    // endpoint antiderivative values
12    double lhs , rhs;
13    // calculate the anti-derivative values
14    // at the interval endpoints
15    rhs = (1.0/3.0)*pow(b,3);
16    lhs = (1.0/3.0)*pow(a,3);
17    // the net area is the difference
18    // from right to left
19    cout << ( rhs - lhs ) << endl;
20    return 0;
21 }
```

Why Comment?

```
1
2
3
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9
10     double a(-1.2), b(2.1);
11
12     double lhs , rhs;
13
14
15     rhs = (1.0/3.0)*pow(b,3);
16     lhs = (1.0/3.0)*pow(a,3);
17
18
19     cout << ( rhs - lhs ) << endl;
20     return 0;
21 }
```

```
1 /**
2  * Calculate the area under  $y=x*x$  from  $x=-1.2$  to  $x=2.1$ .
3  */
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9     // define endpoints
10    double a(-1.2), b(2.1);
11    // endpoint antiderivative values
12    double lhs , rhs;
13    // calculate the anti-derivative values
14    // at the interval endpoints
15    rhs = (1.0/3.0)*pow(b,3);
16    lhs = (1.0/3.0)*pow(a,3);
17    // the net area is the difference
18    // from right to left
19    cout << ( rhs - lhs ) << endl;
20    return 0;
21 }
```

Why Comment?

```
1
2
3
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9
10     double a(-1.2), b(2.1);
11
12     double lhs , rhs;
13
14
15     rhs = (1.0/3.0)*pow(b,3);
16     lhs = (1.0/3.0)*pow(a,3);
17
18
19     cout << ( rhs - lhs ) << endl;
20     return 0;
21 }
```

```
1 /**
2  * Calculate the area under  $y=x*x$  from  $x=-1.2$  to  $x=2.1$ .
3  */
4 #include <cmath>
5 #include <iostream>
6 using namespace std;
7 int main()
8 {
9     // define endpoints
10
11     // endpoint antiderivative values
12
13     // calculate the anti-derivative values
14     // at the interval endpoints
15
16
17     // the net area is the difference
18     // from right to left
19
20
21 }
```

Notice the contrast: even a well written program is difficult to understand without ample commentary — but ample commentary (by definition) **is solely sufficient** to understand what a program does.

Pre-Processor Directives (Line 4–5)

- ▶ pre-processing performed before program is compiled
- ▶ generally included after initial comments
- ▶ begin with a '#', only one on a line
- ▶ **inserts source code to be processed by the compiler**

```
1 //
2 // This program computes the volume of a box
3 //
4 #include <cstdlib>
5 #include <iostream>
6 using namespace std;
7
8 int main()
9 {
10     /* Declare and initialize objects */
11     double length(20.75), width(11.5);
12     double height = 9.5;
13     double volume;
14
15     /* Calculate volume. */
16     volume = length * width * height;
17     /* Print the volume. */
18     cout << "The volume is " << volume ;
19     cout << " units cubed." << endl;
20
21     system("PAUSE");
22     // Exit program.
23     return 0;
24 }
```

Pre-processor Demonstration

Compiler Directives (Line 6)

- ▶ tells compiler which library names or external references to use
- ▶ generally included between *the associated* pre-processor statements and the beginning of `int main()`
- ▶ ends with `';`

```
1 //  
2 // This program computes the volume of a box  
3 //  
4 #include <cstdlib>  
5 #include <iostream>  
6 using namespace std;  
7  
8 int main()  
9 {  
10     /* Declare and initialize objects */  
11     double length(20.75), width(11.5);  
12     double height = 9.5;  
13     double volume;  
14  
15     /* Calculate volume. */  
16     volume = length * width * height;  
17     /* Print the volume. */  
18     cout << "The volume is " << volume ;  
19     cout << " units cubed." << endl;  
20  
21     system("PAUSE");  
22     /* Exit program. */  
23     return 0;  
24 }
```

Forgotten Directive Error

```
1 /**
2  * This will not compile.
3  */
4 #include <iostream>
5
6 int main()
7 {
8     // What is cout? We forgot the using ... std;
9     // directive above!
10    cout << "Hello_World." << endl;
11    return 0;
12 }
```

In function 'int main()':

10: error: 'cout' was not declared in this scope

10: error: 'endl' was not declared in this scope

Variables

The declaration of a variable “names” a memory region:

1. The memory region has an *address* and a *size*, but you just need to be concerned with the *name you give it*.
2. Make the name representative of how the value stored is used in your algorithm.
3. The names of variables are called **identifiers**.

Naming Variables

1. Characters in names may come from a-z, A-Z, _ (underscore), and 0-9.
2. Variable names **cannot** begin with a number, they must begin with an underscore or letter.
3. Variable names cannot be **reserved keywords** (using, namespace, ... page 41!).
 - ▶ Names are case-sensitive! fooBar is not FooBar.

CamelCase	WindSpeed, InitialVelocity, finalVelocity
under_scored_names	wind_speed, init_velocity, final_velocity

Variable Types

The fundamental types of information used in programming are:

1. Boolean values (either true or false),
use bool.
2. Integer Numbers,
use int.
3. Real numbers (fractions),
use double.
4. Strings (words, phrases, sentences, paragraphs, ...),
use the string class...

String Variable Example

```
1 // An example of declaring a string variable
2 // with an initial value of "Hello World"
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 int main()
9 {
10     string theGreeting( "Hello World" );
11     cout << theGreeting << endl;
12     return 0;
13 }
```

Mathematical Constants and Functions

```

1  /**
2  * Declaring mathematical constants and using
3  * mathematical functions.
4  */
5  #include <cmath> // For all the math stuff
6  #include <iostream>
7  using namespace std;
8  int main()
9  {
10     const double PI = acos(-1);
11     const double E = exp(1), PI4 = atan(1);
12     double x; // a regular variable
13     cout.setf( ios::fixed );
14     cout.precision(14);
15     cout << "Natural_base:_" << E << endl;
16     cout << "_____pi:_" << PI << endl;
17     cout << "_____pi/4:_" << PI4 << endl;
18     x = ( PI / PI4 ) * E;
19     cout << "_____4e:_" << x << endl;
20     x = sqrt(E);
21     cout << "Sq-root_of_e:_" << x << endl;
22     x = cos(PI);
23     cout << "_____cos(pi):_" << x << endl;
24     x = pow(PI,3);
25     cout << "_____pi_cubed:_" << x << endl;
26     x = log(E);
27     cout << "_____log(e):_" << x << endl;
28     return 0;
29 }

```

```

Natural base: 2.71828182845905
           pi: 3.14159265358979
           pi/4: 0.78539816339745
           4e: 10.87312731383618
Sq-root of e: 1.64872127070013
           cos(pi): -1.00000000000000
           pi cubed: 31.00627668029982
           log(e): 1.0000000000000000

```

RUN

EDIT

const_and_cmath.cxx

Standard Output (`cout << numbers`)

Standard output is a *stream*, an ordered sequence of bytes. Typically, these bytes represent the English alphabet and Latin numerals. **So standard output is an easy way for a programmer to tell the user something.**

```
1 cout << 3.456 << endl;
```

```
3.456
```

RUN

EDIT

output_float.cxx

```
1 cout << 123 << 456 << endl;
```

```
123456
```

RUN

EDIT

concat_ints.cxx

```
1 cout << 123 << endl << 456 << endl;
```

```
123
```

```
456
```

RUN

EDIT

newline_ints.cxx

What's this endl thing?

```
cout << "text";
```

Specify text to be written in double quotes (").

```
1 cout << "Here_is_the_text!" << endl;
```

```
Here is the text!
```

RUN

EDIT

cout_simple_text.cxx

Now we can separate numbers with spaces:

```
1 cout << 123 << 456 << endl;
```

```
2 cout << 123 << " " << 456 << endl;
```

```
123456
123 456
```

RUN

EDIT

cout_space_ints.cxx

What will this print?

```
1 cout << "C" << "+" << "+" << endl;
```

```
2 cout << " " << "ROCKS!" << " " << endl;
```

```
cout << "text";
```

Specify text to be written in double quotes (").

```
1 cout << "Here_is_the_text!" << endl;
```

```
Here is the text!
```

RUN

EDIT

cout_simple_text.cxx

Now we can separate numbers with spaces:

```
1 cout << 123 << 456 << endl;
2 cout << 123 << " " << 456 << endl;
```

```
123456
123 456
```

RUN

EDIT

cout_space_ints.cxx

What will this print?

```
1 cout << "C++" << "+" << "ROCKS!" << endl;
2 cout << "ROCKS!" << "C++" << endl;
```

```
C++
ROCKS!
```

RUN

EDIT

cout_cxx_rocks.cxx

Standard Input `cin >> variable;`

Just as `cout` “knows” how to print Boolean values, integers, \Re numbers, and “text”...

The *standard input* (`cin`) knows how to read data into these variable types.

```

1 bool X, Y;
2 cout << "Enter Booleans X and Y" << endl;
3 cin >> X >> Y ;
4 cout << "Read X=" << X << " and "
5     << "Y=" << Y << endl;
6 cout.setf( ios::boolalpha );
7 cout << "Read X=" << X << " and "
8     << "Y=" << Y << endl;

```

```
<<Interactive Program>>
```

```
RUN
```

```
EDIT
```

```
cin_bools.cxx
```

Standard Input `cin >> variable;`

```

1 int Integer;
2 string Text;
3 cout << "Enter an integer and a word: "
4   << flush;
5 cin >> Integer >> Text ;
6 cout << "Read Integer=" << Integer <<
7   " and " << "Word=" << Text
8   << endl;

```

```

1 double X, Y;
2 cout << "Enter Numbers X & Y" << endl;
3 cin >> X >> Y ;
4
5 cout << "Read X=" << X << " and "
6   << "Y=" << Y << endl;

```

<<Interactive Program>>

RUN

EDIT

cin_int_string.cxx

<<Interactive Program>>

RUN

EDIT

cin_doubles.cxx

finis