Simple C++ Programs

November 15, 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;
}
```

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): 1.00000000000000
```

RUN EDIT const_and_cmath.cxx

Computation Headaches

Integer Computations

Type Specifier	Positive Range
int	0 - 2,147,483,647

```
1 // note the absence of commas in the
2 // numbers
3 int bigInt1 = 2000000000;
4 int bigInt2 = 1111111111;
5
6 cout << bigInt1 << "␣+␣" <<
7          bigInt2 << "␣=␣" <<
8          bigInt1 + bigInt2 << endl;
```

Integer Computations

Type Specifier	Positive Range
int	0 - 2,147,483,647

Overflow

```
1 // note the absence of commas in the
2 // numbers
3 int bigInt1 = 2000000000;
4 int bigInt2 = 1111111111;
5
6 cout << bigInt1 << " + " <<
7           bigInt2 << " = " <<
8           bigInt1 + bigInt2 << endl;
```

```
2000000000 + 1111111111 = -1183856185
```

RUN EDIT int_overflow.cxx

Integer Computations

Type Specifier	Positive Range
int	0 - 2,147,483,647

Truncated Division

```
1  // initialize two variables to
2  // one and a half
3  int integerHalf( 3 / 2 );
4  double doubleHalf( 3 / 2 );
5
6  cout.setf( ios::fixed );
7  cout.precision(4);
8  cout << "integerHalf " <<
9          integerHalf << endl;
10
11 cout << "doubleHalf " <<
12         doubleHalf << endl;
```

```
integerHalf 1
doubleHalf 1.0000
```

RUN EDIT int_truncated.cxx

Integer Computations

Type Specifier	Positive Range
int	0 - 2,147,483,647

“Decimal Casting & Promotion”

```
1 // initialize a variable to
2 // one and a half using decimal casting
3 double doubleHalf( 3.0 / 2 );
4
5 cout.setf( ios::fixed );
6 cout.precision(4);
7
8 cout << "doubleHalf " <<
9         doubleHalf << endl;
```

```
doubleHalf 1.5000
```

RUN EDIT decimal_casting.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" << "+" <<  "+␣␣" << endl;
2 cout << "␣␣" << "ROCKS!" << "␣" << 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␣&␣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;
```

<<Interactive Program>>

RUN EDIT cin_int_string.cxx

```
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_doubles.cxx

finis