Instructions

For a each topic (if applicable) there are 5 sections:

Short Answer

Answer each question as well as you can.

Circle the Variables' Scopes

Circle the scope of the given variables – i.e., circle the part of the code in which each variable "exists". A good test, if you're not sure, is: if you inserted a statement to `cout` the variable at a given point in the code, would any errors be produced?

Fill in the Memory

Fill in (or draw) a picture representing the computer memory, showing how the variables might be allocated, and whether they have an assigned value, or are uninitialized (use `?` for "uninitialized").

Fix the Broken Code

Modify the code in the cleanest (and shortest) way possible, so that it will compile and run without errors or warnings.

Trace the Working Code

Trace through the execution of the code, keeping track of the value of each variable at each point in time, and the final output that the program produces.

1 Variables and Assignment

Short Answer

Question

Describe undeclared variables, uninitialized variables, and initialized variables.

Answer

Undeclared variables don't really exist. This term usually shows up in error messages when you try to use a variable that you haven't declared yet (either because you forgot, or you misspelled or mis-capitalized it, etc.).

Uninitialized variables are declared, but have not yet been given a value. This means that their value is undefined (i.e., garbage, or whatever was in that location in memory before it was assigned to that variable). Variables should not be used before being given a value.

Initialized variables are both declared an initialized. This means that they have been assigned a location in memory, and that that memory has been set to some desired value.

Question

Describe what we mean when we refer to a variable's scope.

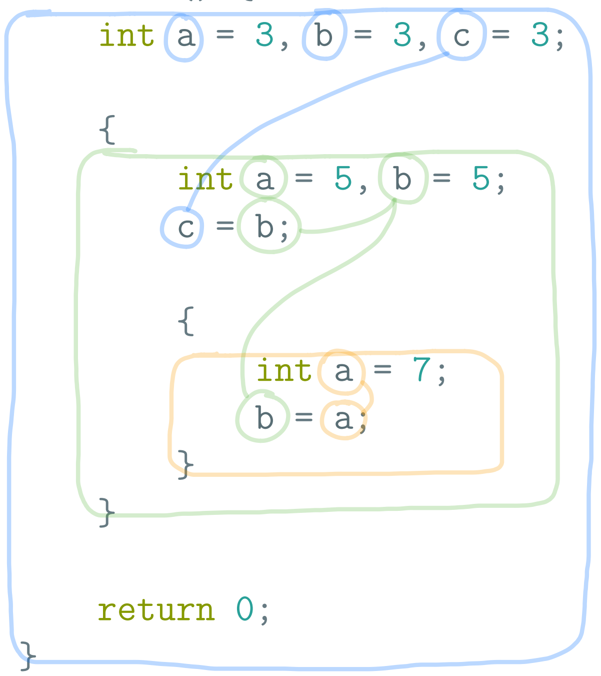
Answer

A variable's scope is the region in the code where the variable "exists". Inside this scope, after the variable is declared, the variable may be used. Outside this scope, or before the variable has been declared, attempting to use the variable will result in a compile time error (in C++).

In practice, the scope of most variables is determined by the innermost pair of curly braces containing it.

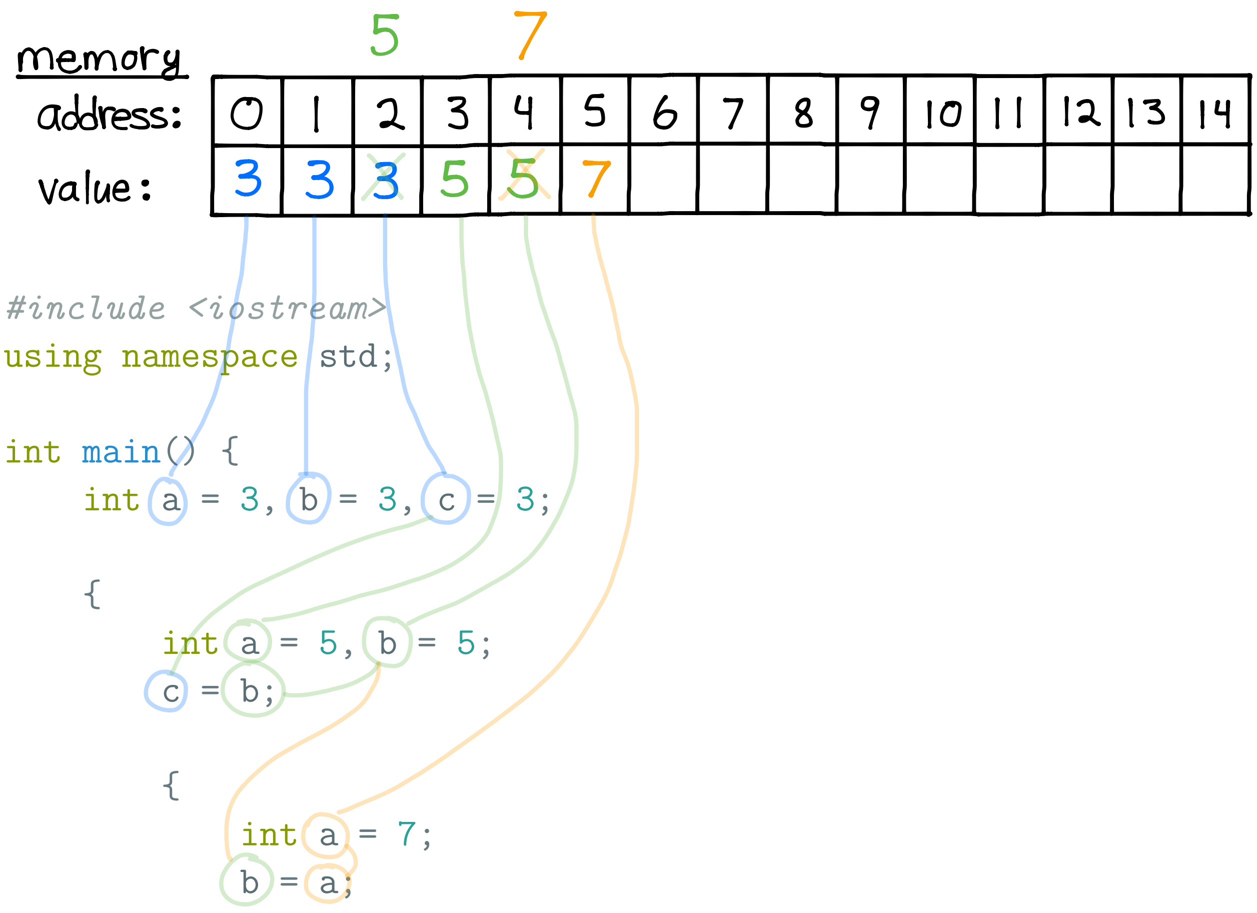
It is possible for a variable in an inner scope to have the same name as a variable in an outer scope. If this is the case, the variable in the inner scope is said to "mask" the variable from the outer scope, beginning from the point at which it is declared. Once execution proceeds outside the inner scope, the variable from the outer scope will be "visible" (i.e., usable) again.

Circle the Variables' Scopes



```
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 3, c = 3;

    {
        int a = 5, b = 5;
        c = b;

        {
            int a = 7;
            b = a;
        }
    }

    return 0;
}
```

Fill in the Memory



memory address:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
value:	3	3	~~3~~ 5	5	~~5~~ 7	7									

```
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 3, c = 3;

    {
        int a = 5, b = 5;
        c = b;

        {
            int a = 7;
            b = a;
        }
    }

    return 0;
}
```

Fix the Broken Code

```
#include <iostream>
using namespace std;

int main() {
    int a = 5;
    int b = 7;

    cout << "before swapping: " << a << " " << b << endl;

    // swap
    int temp = a;
    a = b;
    b = temp;

    cout << "after swapping: " << a << " " << b << endl;

    return 0;
}
```

```
--- code/1-fix-1.cpp        2014-05-07 18:19:13.000000000 -0700
+++ code/1-fix-1.answer.cpp        2014-04-30 22:26:24.000000000 -0700
@@ -2,17 +2,15 @@
 using namespace std;

 int main() {
-    // 5 mistakes
-
     int a = 5;
     int b = 7;

     cout << "before swapping: " << a << " " << b << endl;

     // swap
-    int temp = int a;
-    int a = int b;
-    int b = int temp;
+    int temp = a;
+    a = b;
+    b = temp;

     cout << "after swapping: " << a << " " << b << endl;
```

```
before swapping: 5 7
after swapping: 7 5
```

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    int    i = '5' + 4.7;  // '5' == 53
    char   c = 42;         // '*' == 42
    bool   b = 10;
    double d =  7;
    string s = "hello world!";

    int u = 42;  // otherwise we're using an uninitialized variable below

    cout << i << " " << c << " " << b << " "
         << d << " " << s << " " << u << endl;

    return 0;
}
```

```
--- code/1-fix-2.cpp        2014-05-07 18:19:37.000000000 -0700
+++ code/1-fix-2.answer.cpp        2014-05-02 11:27:14.000000000 -0700
@@ -3,15 +3,13 @@
 using namespace std;

 int main() {
-    // 1 mistake
-
     int    i = '5' + 4.7;  // '5' == 53
     char   c = 42;         // '*' == 42
     bool   b = 10;
     double d =  7;
     string s = "hello world!";

-    int u;
+    int u = 42;  // otherwise we're using an uninitialized variable below

     cout << i << " " << c << " " << b << " "
          << d << " " << s << " " << u << endl;
```

```
57 * 1 7 hello world! 42
```

Trace the Working Code

```
#include <iostream>
using namespace std;

int main() {
    cout << "a b temp\n" << "--------\n";

    int a = 5;     cout<<a<<endl;
    int b = 7;     cout<<a<<" "<<b<<endl;

    int temp = a;  cout<<a<<" "<<b<<"  "<<temp<<endl;
    a = b;         cout<<a<<" "<<b<<"  "<<temp<<endl;
    b = temp;      cout<<a<<" "<<b<<"  "<<temp<<endl;

    return 0;
}
```

```
a b temp
--------
5
5 7
5 7  5
7 7  5
7 5  5
```

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "a b c d\n" << "-------\n";

    int a, b, c;

    a = b = c = 3;  cout<<a<<" "<<b<<" "<<c<<endl;
    b = c = 5;      cout<<a<<" "<<b<<" "<<c<<endl;
    c = 7;          cout<<a<<" "<<b<<" "<<c<<endl;

    int d = b;      cout<<a<<" "<<b<<" "<<c<<" "<<d<<endl;

    return 0;
}
```

```
a b c d
-------
3 3 3
3 5 5
3 5 7
3 5 7 5
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 3;
    cout<<"a: "<<a<<"  b: "<<b<<endl;
    {
        int a = 5;
        b = a;
        cout<<"a: "<<a<<"  b: "<<b<<endl;
    }
    cout<<"a: "<<a<<"  b: "<<b<<endl;

    return 0;
}
```

```
a: 3  b: 3
a: 5  b: 5
a: 3  b: 5
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 3, c = 3;
    cout<<"a: "<<a<<"  b: "<<b<<"  c: "<<c<<endl;
    {
        int a = 5, b = 5;
        c = b;
        cout<<"a: "<<a<<"  b: "<<b<<"  c: "<<c<<endl;
        {
            int a = 7;
            b = a;
            cout<<"a: "<<a<<"  b: "<<b<<"  c: "<<c<<endl;
        }
        cout<<"a: "<<a<<"  b: "<<b<<"  c: "<<c<<endl;
    }
    cout<<"a: "<<a<<"  b: "<<b<<"  c: "<<c<<endl;

    return 0;
}
```

```
a: 3  b: 3  c: 3
a: 5  b: 5  c: 5
a: 7  b: 7  c: 5
a: 5  b: 7  c: 5
a: 3  b: 3  c: 5
```

2 Data Types and Expressions

Short Answer

Question

What are the 5 main datatypes we studied during this course, and what is each typically used for?

Answer

Integer: `int`. Used for numbers *without* a decimal point, which may be either positive or negative (may represent, on typical modern systems, any number in the range $[-2^{31}, 2^{31}-1]$).

Character: `char`. Used for single characters, such as `'a'` or `'C'`. Note that characters are considered a special type of integer.

Boolean: `bool`. Used for `true/false` values. The value `true` is equivalent to `1`, and `false` is equivalent to `0`.

Double: `double`. Used for numbers *with* a decimal point. Cautions: often introduce strange rounding errors; larger in memory, and more complicated than integers.

String: `string`. Used for strings of text, such as `"hello world!"`. Can typically also be treated as arrays of single characters. Must `#include <string>`.

Question

Give examples of what we mean by "order of operations' – why does it matter? What is the order of operations for the following common operators:
`!=`, `==`, `=`, `<`, `>=`, `+`, `&&`, `||`, `*`, `()`, `>`, `/`, `<=`, `-`, `%`.

Answer

The order of operations is the order in which different operators are performed. Operators with higher precedence will be performed first, even if they occur later in an expression.

Consider the expression `int x = 3 + 5 * 7 + 11`. Because of the order of operations (i.e., operator precedence) this is the same as `int x = 3 + (5 * 7) + 11`, and the answer is `49`. All operators in C++ have a defined level of precedence.

Recall the operator precedence chart from some time back:
`http://en.cppreference.com/w/cpp/language/operator_precedence`

The correct order is: `()`, `*`, `/`, `%`, `+`, `-`, `<`, `<=`, `>`, `>=`, `==`, `!=`, `&&`, `||`, `=`,
where operators with less space between them have equal precedence and (for all of these groups of operators) are executed from left to right within their groups.

Fix the Broken Code

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3;
    cout << "One third is: " << (1.0 / a) << endl;

    // any of these will also work
    cout << "One third is: " << (1 / double(a)) << " and "
                             << (double(1) / a) << " and "
                             << (double(1) / double(a)) << endl;

    double b = 5;
    cout << "One fifth is: " << (1.0 / b) << endl;

    // any of these will also work
    cout << "One fifth is: " << (1 / b) << " and "
                             << (1 / double(b)) << " and "
                             << (double(1) / b) << " and "
                             << (double(1) / double(b)) << endl;

    return 0;
}
```

```
--- code/2-fix-1.cpp        2014-05-07 18:20:29.000000000 -0700
+++ code/2-fix-1.answer.cpp        2014-05-06 16:06:50.000000000 -0700
@@ -2,13 +2,22 @@
 using namespace std;

 int main() {
-    // 1 mistake
-
     int a = 3;
-    cout << "One third is: " << (1 / a) << endl;
+    cout << "One third is: " << (1.0 / a) << endl;
+
+    // any of these will also work
+    cout << "One third is: " << (1 / double(a)) << " and "
+                             << (double(1) / a) << " and "
+                             << (double(1) / double(a)) << endl;

     double b = 5;
-    cout << "One fifth is: " << (1 / b) << endl;
+    cout << "One fifth is: " << (1.0 / b) << endl;
+
+    // any of these will also work
+    cout << "One fifth is: " << (1 / b) << " and "
+                             << (1 / double(b)) << " and "
+                             << (double(1) / b) << " and "
+                             << (double(1) / double(b)) << endl;

     return 0;
 }
```

```
One third is: 0.333333
One third is: 0.333333 and 0.333333 and 0.333333
One fifth is: 0.2
One fifth is: 0.2 and 0.2 and 0.2 and 0.2
```

```
#include <iostream>
using namespace std;

int main() {
    // formula (in math notation): C = (F-32)*(5/9)

    double fahrenheit = 100;  // getting close to summer!

    cout << "The temperature is:\n";
    cout << "    " << fahrenheit << " degrees Fahrenheit\n";
    cout << "    " << (fahrenheit - 32) * (5.0/9.0)
         << " degrees Celcius\n";

    return 0;
}
```

```
--- code/2-fix-2.cpp        2014-05-07 18:21:45.000000000 -0700
+++ code/2-fix-2.answer.cpp        2014-05-06 16:02:52.000000000 -0700
@@ -2,15 +2,13 @@
 using namespace std;

 int main() {
-    // 1 line has mistakes
-
     // formula (in math notation): C = (F-32)*(5/9)

     double fahrenheit = 100;  // getting close to summer!

     cout << "The temperature is:\n";
     cout << "    " << fahrenheit << " degrees Fahrenheit\n";
-    cout << "    " << fahrenheit-32 * 5/9
+    cout << "    " << (fahrenheit - 32) * (5.0/9.0)
          << " degrees Celcius\n";

     return 0;
```

```
The temperature is:
    100 degrees Fahrenheit
    37.7778 degrees Celcius
```

Trace the Working Code

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 5, c = 7;

    cout << ( 23%17 || ( -c && c != a && !!11 >= -a ) || 11/3 ) << endl;

    return 0;
}
```

```
1
```

```
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 5, c = 7;

    cout << ( b == +a == b <= !c > !(13%17) <= +c - ( a == c ) ) << endl;

    return 0;
}
```

```
0
```

```
#include <iostream>
using namespace std;

int main() {
    // 'a' == 97
    cout << ('a' + 5) << endl;
    cout << char('a' + 5) << endl;

    cout << 7 << endl;
    cout << (7/3) << endl;
    cout << double(7) << endl;

    cout << 3.14159265359 << endl;
    cout << int(3.14159265359) << endl;
    cout << char( 100 + int(3.14159265359) ) << endl;

    cout << (!53) << endl;
    cout << bool(5) << endl;

    cout << "this is a string" << endl;

    cout << (3 + 5 * 3 + 5 / 2) << endl;

    cout << (10 < 5 < 7) << endl;

    return 0;
}
```

```
102
f
7
2
7
3.14159
3
g
0
1
this is a string
20
1
```

3 If and If-Else

Fix the Broken Code

```
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    if (0 < a && a < 2)
        cout << "true" << endl;
    else
        cout << "false" << endl;

    return 0;
}
```

```
--- code/3-fix-1.cpp        2014-05-07 18:22:23.000000000 -0700
+++ code/3-fix-1.answer.cpp        2014-05-06 17:19:16.000000000 -0700
@@ -2,12 +2,10 @@
 using namespace std;

 int main() {
-    // 2 mistakes
-
     int a = 3;

-    if (0 < a < 2)
-        cout << "true" << endl
+    if (0 < a && a < 2)
+        cout << "true" << endl;
     else
         cout << "false" << endl;
```

```
false
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    if (a == 5)
        cout << "true" << endl;
    else
        cout << "false" << endl;

    return 0;
}
```

```diff
--- code/3-fix-2.cpp        2014-05-07 18:22:42.000000000 -0700
+++ code/3-fix-2.answer.cpp        2014-05-06 17:19:28.000000000 -0700
@@ -2,11 +2,9 @@
 using namespace std;

 int main() {
-    // 1 mistake
-
     int a = 3;

-    if (a = 5)
+    if (a == 5)
         cout << "true" << endl;
     else
         cout << "false" << endl;
```

```
false
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    if (a == 5) {
        cout << "true" << endl;
        a++;
    } else {
        cout << "false" << endl;
        a--;
    }

    cout << "a = " << a << endl;

    return 0;
}
```

```
--- code/3-fix-3.cpp        2014-05-07 18:23:25.000000000 -0700
+++ code/3-fix-3.answer.cpp        2014-05-06 17:39:30.000000000 -0700
@@ -2,14 +2,15 @@
 using namespace std;

 int main() {
-    // 1 type of mistake
-
     int a = 3;

-    if (a = 5)
-        cout << "true" << endl; a++;
-    else
-        cout << "false" << endl; a--;
+    if (a == 5) {
+        cout << "true" << endl;
+        a++;
+    } else {
+        cout << "false" << endl;
+        a--;
+    }

     cout << "a = " << a << endl;
```

```
false
a = 2
```

Trace the Working Code

```cpp
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;

int main() {
    srand(time(NULL));

    int a = rand()%10, b = rand()%10, c = rand()%10;

    cout << "a = " << a << ", b = " << b << ", c = " << c << endl;

    if (a <= b && a <= c)
        cout << "a is smallest" << endl;

    return 0;
}
```

```
a = 3, b = 3, c = 6
a is smallest
```

```cpp
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;

int main() {
    srand(time(NULL));

    int a = rand()%10, b = rand()%10, c = rand()%10;
    int smallest = 0;

    cout << "a = " << a << ", b = " << b << ", c = " << c << endl;

    if (b < c)
        smallest = b;

    if (c <= b)
        smallest = c;

    if (a <= b && a <= c) {
        smallest = a;
        cout << "a is smallest" << endl;
    }

    return 0;
}
```

```
a = 3, b = 3, c = 6
a is smallest
```

```
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;

int main() {
    srand(time(NULL));

    int a = rand()%10, b = rand()%10, c = rand()%10;

    cout << "a = " << a << ", b = " << b << ", c = " << c << endl;

    if (a <= b && a <= c)
        cout << "a is smallest" << endl;
    else if (b <= a && b <=c)
        cout << "b is smallest" << endl;
    else
        cout << "c is smallest" << endl;

    return 0;
}
```

```
a = 3, b = 3, c = 6
a is smallest
```

```cpp
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;

int main() {
    srand(time(NULL));

    int a = rand()%10, b = rand()%10, c = rand()%10;
    int smallest;

    cout << "a = " << a << ", b = " << b << ", c = " << c << endl;

    if (a <= b && a <= c) {
        smallest = a;
        cout << "a is smallest" << endl;
    } else if (b <= a && b <=c) {
        smallest = b;
        cout << "b is smallest" << endl;
    } else {
        smallest = c;
        cout << "c is smallest" << endl;
    }

    cout << "smallest = " << smallest << endl;

    return 0;
}
```

```
a = 0, b = 2, c = 2
a is smallest
smallest = 0
```

```cpp
#include <cstdlib>
#include <ctime>
#include <iostream>
using namespace std;

int main() {
    srand(time(NULL));

    int a = rand()%10, b = rand()%10, c = rand()%10;
    int smallest;

    cout << "a = " << a << ", b = " << b << ", c = " << c << endl;

    if (a <= b && a <= c) {
        smallest = a;
        cout << "a is smallest" << endl;
    } else if (b <= a && b <=c) {
        smallest = b;
        cout << "b is smallest" << endl;
    } else if (c <= a && c <= b) {
        smallest = c;
        cout << "c is smallest" << endl;
    }

    cout << "smallest = " << smallest << endl;

    return 0;
}
```

```
a = 0, b = 2, c = 2
a is smallest
smallest = 0
```

4 Functions

Short Answer

Question

What is a function? What is a predefined function? Why do we care?

Answer

A function is a group of statements that we give (or someone else gives) a name, so that we can call it (execute it) later in the program.

A predefined function is a function defined in the C++ standard library (or by any other library you're using).

Functions are useful, among other things, for making code cleaner. If we have something that needs to be done many times at different points in our code (like printing the board for a maze program...), and we put it in a function, we only need to write it once: instead of copy+pasting the code every time we need to do that, we can now call the function.

Predefined functions are especially useful, because they are essentially code we don't have to write. If we need to generate a random number, for instance, we can call `rand()` – without needing to know or even look at how it's implemented.

Question

What library contains the `pow()` function? What are two other functions this library contains?

Answer

The `pow()` function is contained in the `<cmath>` (in C++; or `<math.h>` in C) library.

This library also contains the `sin()`, `cos()`, `tan()`, `log()`, `sqrt()`, `ceil()`, `floor()`, and `abs()` functions, among others.

`http://www.cplusplus.com/reference/cmath/`

Fix the Broken Code

```cpp
#include <cmath>
#include <iostream>
using namespace std;

int main() {
    int a = 3, b = 5, c = 7;

    cout << a << "^" << b << " (mod " << c << ") = "
         << int(pow(a,b)) % c << endl;

    return 0;
}
```

```diff
--- code/4-fix-1.cpp        2014-05-06 18:52:06.000000000 -0700
+++ code/4-fix-1.answer.cpp        2014-05-06 18:51:38.000000000 -0700
@@ -3,13 +3,10 @@
 using namespace std;

 int main() {
-    // 6 incorrect operators
-    // 6 incorrect delimiters
-
     int a = 3, b = 5, c = 7;

-    cout >> a >> '^' >> b >> ' (mod ' >> c >> ') = '
-         >> int(pow(a,b)) % c >> endl;
+    cout << a << "^" << b << " (mod " << c << ") = "
+         << int(pow(a,b)) % c << endl;

     return 0;
 }
```

```
3^5 (mod 7) = 5
```

```
#include <cmath>
#include <iostream>
using namespace std;

int main() {
    int a = 3,  b = 5,
        c = 7,  d = 11;
    double e = 13.3, f = 17.7;

    cout << "the sin of " << a << " is " << sin(a) << endl
         << "the cosine of " << b << " is " << sin(b) << endl
         << "the tangent of " << c << " is " << sin(c) << endl

         << "the natural log of " << d << " is " << log(d) << endl
         << "the ceil of " << e << " is " << ceil(e) << endl
         << "the floor of " << f << " is " << floor(f) << endl;

    return 0;
}
```

```
--- code/4-fix-2.cpp        2014-05-06 19:08:41.000000000 -0700
+++ code/4-fix-2.answer.cpp        2014-05-06 18:49:34.000000000 -0700
@@ -3,18 +3,16 @@
 using namespace std;

 int main() {
-    // 4 mistakes
-
-    int a = 3,  b = 5
-        c = 7,  d == 11;
+    int a = 3,  b = 5,
+        c = 7,  d = 11;
     double e = 13.3, f = 17.7;

     cout << "the sin of " << a << " is " << sin(a) << endl
          << "the cosine of " << b << " is " << sin(b) << endl
-         << "the tangent of " << c << " is " << sin(c) << endl;
+         << "the tangent of " << c << " is " << sin(c) << endl

          << "the natural log of " << d << " is " << log(d) << endl
-         << "the ceil of " << e << " is " << ceil(e) << << endl
+         << "the ceil of " << e << " is " << ceil(e) << endl
          << "the floor of " << f << " is " << floor(f) << endl;

     return 0;
```

```
the sin of 3 is 0.14112
the cosine of 5 is -0.958924
the tangent of 7 is 0.656987
the natural log of 11 is 2.3979
the ceil of 13.3 is 14
the floor of 17.7 is 17
```

```cpp
#include <cmath>
#include <iostream>
using namespace std;


// This function prints "hello world!"
//
// Arguments: none
// Returns: nothing
void print_hello() {
    cout << "hello world!" << endl;
}


int main() {
    // 1 error

    cout << "now we are going to print \"hello world!\"\n";

    print_hello();

    return 0;
}
```

```diff
--- code/4-fix-3.cpp        2014-05-07 18:14:36.000000000 -0700
+++ code/4-fix-3.answer.cpp        2014-05-07 18:14:21.000000000 -0700
@@ -13,11 +13,11 @@


 int main() {
-    // 1 mistake
+    // 1 error

     cout << "now we are going to print \"hello world!\"\n";

-    print_hello;
+    print_hello();

     return 0;
 }
```

```
now we are going to print "hello world!"
hello world!
```

5 Loops

Short Answer

Question

What are the three types of loops in C++? Explain what each does, or what you might use it for.

Answer

For loop: `for ([init] ; [condition] ; [iter]) { [body] }`. Executes `[init]`, checks `[condition]`, executes `[body]`, executes `[iter]`, and then goes back to checking `[condition]`, exiting when `[condition]` is `false`. Useful for everything, but especially for looping through arrays, or for anything else that requires a counter variable.

While loop: `while ([condition]) { [body] }`. Checks `[condition]`, executes `[body]`, and then goes back to checking `[condition]`, exiting when `[condition]` is `false`. Useful for doing things like reading from a file until you have reached the end.

Do-while loop: `do { [body] } while( [condition] );`. Executes `[body]`, checks `[condition]`, and then goes back to executing `[body]`, exiting when it `[condition]` is found to be false. Good for things like displaying a menu, responding to the user's choice, and then displaying the menu again until the user enters `'n'` for `"do you want to continue?"`.

Question

Is it ever okay to have an infinite loop? What usually causes unintentional infinite loops?

Answer

Yes! As long as you do it on purpose, for a good reason.

Most intentional infinite loops are called "event loops". Long running programs (like your operating system, or (the evil) Microsoft Word) typically have a toplevel infinite loop. Inside this loop, the program will do things like check to see if the user has entered any new input, update the display, maybe save a backup copy of the file you're working on (in the case of Word) or check to see if there's a WiFi signal nearby (in the case of your OS), and then sleep for a while until it's time to do it again. We saw a bit of this type of control flow in the maze program.

Unintentional infinite loops are usually caused by forgetting to update the counter variable (in a for loop), or by writing a condition who's value never changes, because none of the variables used in it are referenced in the body of the loop (in a while or do-while loop).

Circle the Variables' Scopes

```
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    for (int i = 7; i > a; i--) {

        for (int j = 11; j > i; j--) {

            cout << "*";

        }

        cout << endl;

    }

    return 0;
}
```

```
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    int i = 7;
    while (i > a) {

        int j = 11;
        while (j > i) {

            cout << "*";
            j--;

        }

        cout << endl;
        i--;

    }

    return 0;
}
```

```
#include <iostream>
using namespace std;

int main() {
    // have to be a little careful with do-while loops:
    // it's not always possible to write one that means the
    // same thing as the corresponding for or while loop

    int a = 3;

    int i = 7;
    do {

        int j = 11;
        do {

            cout << "*";
            j--;

        } while (j > i);

        cout << endl;
        i--;

    } while (i > a);

    return 0;
}
```

Fix the Broken Code

```
#include <iostream>
using namespace std;

int main() {
    cout << "i = ";
    for (int i=0; i<5; i++) {
        cout << i << ", ";
    }
    cout << endl;

    cout << "i*2 = ";
    for (int i=0; i<5; i++) {
        cout << i*2 << ", ";
    }
    cout << endl;

    return 0;
}
```

```
--- code/5-fix-1.cpp        2014-05-07 18:25:43.000000000 -0700
+++ code/5-fix-1.answer.cpp        2014-05-06 20:32:01.000000000 -0700
@@ -2,16 +2,14 @@
 using namespace std;

 int main() {
-    // 2 mistakes
-
     cout << "i = ";
-    for (int i=0; i<5; i+1) {
+    for (int i=0; i<5; i++) {
         cout << i << ", ";
     }
     cout << endl;

     cout << "i*2 = ";
-    for (int i=0; i<5; i+1) {
+    for (int i=0; i<5; i++) {
         cout << i*2 << ", ";
     }
     cout << endl;
```

```
i = 0, 1, 2, 3, 4,
i*2 = 0, 2, 4, 6, 8,
#include <iostream>
using namespace std;

int main() {
    int a = 0;
    while (a < 5) {
        cout << "a = " << a << endl;
        a++;
    }

    return 0;
}
```

```
--- code/5-fix-2.cpp        2014-05-07 18:26:07.000000000 -0700
+++ code/5-fix-2.answer.cpp        2014-05-06 20:22:38.000000000 -0700
@@ -2,11 +2,10 @@
 using namespace std;

 int main() {
-    // 1 mistake
-
     int a = 0;
     while (a < 5) {
         cout << "a = " << a << endl;
+        a++;
     }

     return 0;
```

```
a = 0
a = 1
a = 2
a = 3
a = 4
```

Trace the Working Code

```
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    for (int i = 7; i > a; i--) {

        for (int j = 11; j > i; j--) {

            cout << "*";

        }

        cout << endl;

    }

    return 0;
}
```

```
****
*****
******
*******
```

```cpp
#include <iostream>
using namespace std;

int main() {
    int a = 3;

    int i = 7;
    while (i > a) {

        int j = 11;
        while (j > i) {

            cout << "*";
            j--;

        }

        cout << endl;
        i--;

    }

    return 0;
}
```

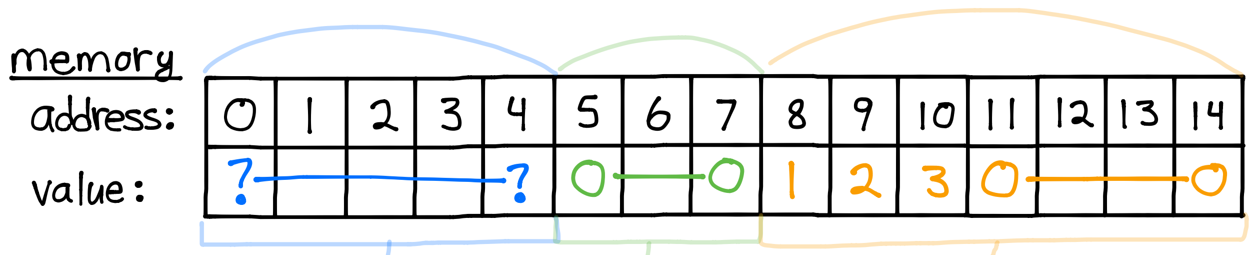
```
****
*****
******
*******
```

```
#include <iostream>
using namespace std;

int main() {
    // have to be a little careful with do-while loops:
    // it's not always possible to write one that means the
    // same thing as the corresponding for or while loop

    int a = 3;

    int i = 7;
    do {

        int j = 11;
        do {

            cout << "*";
            j--;

        } while (j > i);

        cout << endl;
        i--;

    } while (i > a);

    return 0;
}
```

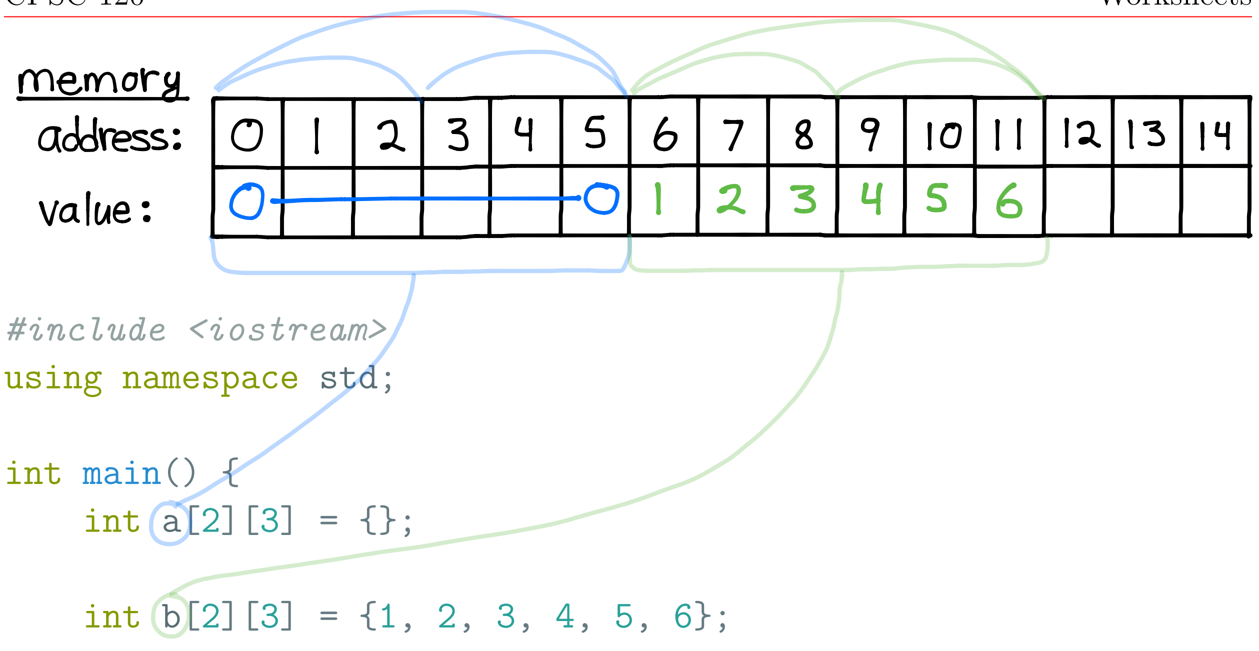
```
****
*****
******
*******
```

6 Arrays

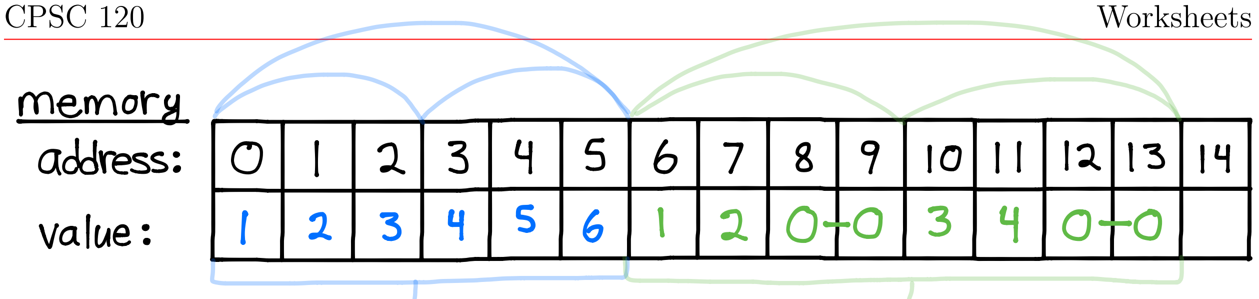
Fill in the Memory



```
#include <iostream>
using namespace std;

int main() {
    int a[5];

    int b[3] = {};

    int c[7] = {1,2,3};

    return 0;
}
```

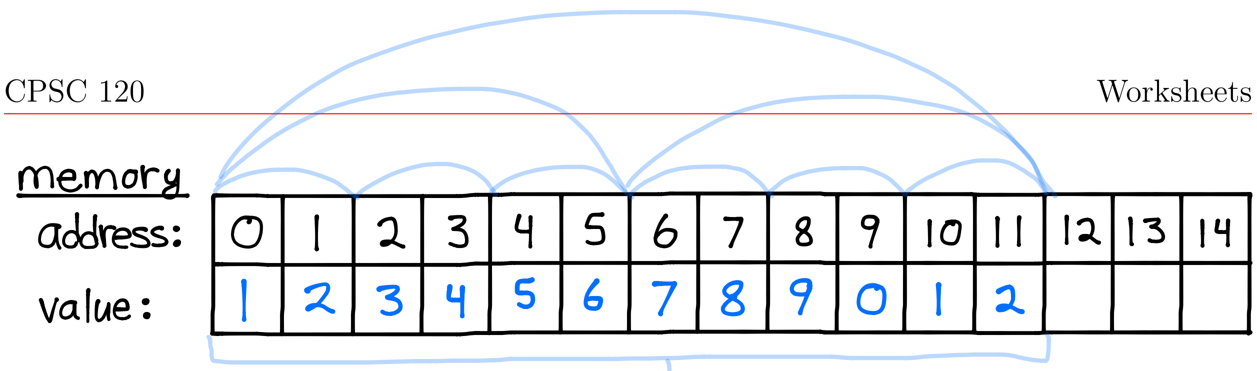


```
#include <iostream>
using namespace std;

int main() {
    int a[2][3] = {};

    int b[2][3] = {1, 2, 3, 4, 5, 6};

    return 0;
}
```

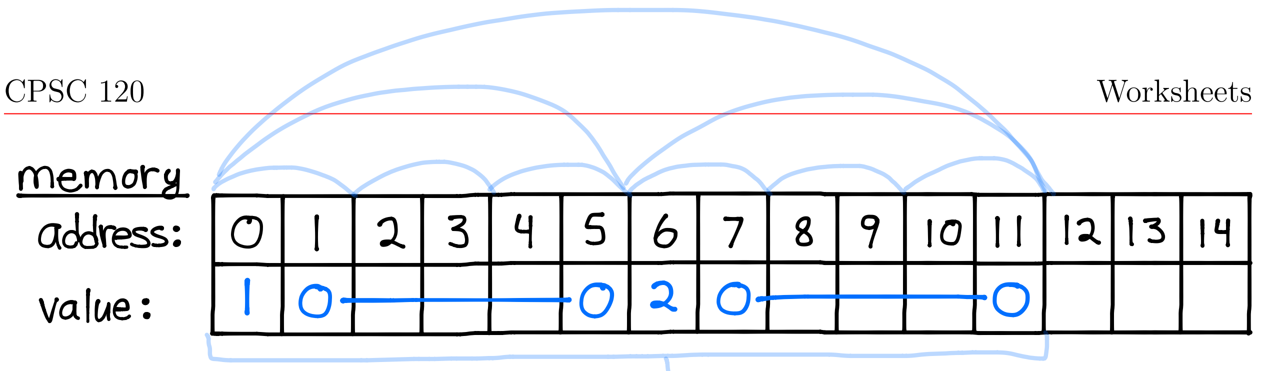


memory address:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
value:	1	2	3	4	5	6	1	2	0	0	3	4	0	0	

```
#include <iostream>
using namespace std;

int main() {
    int a[2][3] = { {1, 2, 3}, {4, 5, 6} };

    int b[2][4] = { {1, 2}, {3, 4} };

    return 0;
}
```



```
#include <iostream>
using namespace std;

int main() {
    int a[2][3][2] = { { {1, 2}, {3, 4}, {5, 6} },
                       { {7, 8}, {9, 0}, {1, 2} } };

    return 0;
}
```



```
#include <iostream>
using namespace std;

int main() {
    int a[2][3][2] = { {1}, {2} };

    return 0;
}
```

Fix the Broken Code

```
#include <iostream>
using namespace std;

int main() {
    int a[7] = {1, 3, 5, 7, 9, 11, 13};

    cout << "a = { ";

    for (int i=0; i<7; i++)
        cout << a[i] << ", ";

    cout << "};\n";

    return 0;
}
```

```
--- code/6-fix-1.cpp        2014-05-06 22:56:01.000000000 -0700
+++ code/6-fix-1.answer.cpp        2014-05-06 22:56:07.000000000 -0700
@@ -2,13 +2,11 @@
 using namespace std;

 int main() {
-    // 2 mistakes
-
-    a[7] = {1, 3, 5, 7, 9, 11, 13};
+    int a[7] = {1, 3, 5, 7, 9, 11, 13};

     cout << "a = { ";

-    for (int i=0; i<=7; i++)
+    for (int i=0; i<7; i++)
         cout << a[i] << ", ";

     cout << "};\n";
```

```
a = { 1, 3, 5, 7, 9, 11, 13, };
```

```
#include <iostream>
using namespace std;

int main() {
    int a[7] = {1, 3, 5, 7, 9, 11, 13};

    int b = a[6];

    cout << "b = " << b << endl;

    return 0;
}
```

```
--- code/6-fix-2.cpp        2014-05-06 22:47:27.000000000 -0700
+++ code/6-fix-2.answer.cpp        2014-05-06 22:47:48.000000000 -0700
@@ -2,11 +2,9 @@
 using namespace std;

 int main() {
-    // 2 mistakes
+    int a[7] = {1, 3, 5, 7, 9, 11, 13};

-    a[7] = {1, 3, 5, 7, 9, 11, 13};
-
-    int b = a[7];
+    int b = a[6];

     cout << "b = " << b << endl;

```

```
b = 13
```

Trace the Working Code

```
#include <iostream>
using namespace std;

int main() {
    int a[7];

    for (int i=0; i<7; i++)
        a[i] = (i*7)%5;

    // print the array
    cout << "a = { ";
    for (int i=0; i<7; i++)
        cout << a[i] << ", ";
    cout << "};\n";

    return 0;
}
```

```
a = { 0, 2, 4, 1, 3, 0, 2, };
```

```
#include <iostream>
using namespace std;

int main() {
    int a[7];

    for (int i=0; i<7; i++)
        cin >> a[i];

    // print the array (to see what the user entered)
    cout << "a = { ";
    for (int i=0; i<7; i++)
        cout << a[i] << ", ";
    cout << "};\n";

    return 0;
}
```

```
a = { 11, 7, 1, 13, 5, 9, 3, };
```

```
#include <iostream>
using namespace std;

int main() {
    int a[7] = { 6, 7, 3, 5, 2, 1, 4 };

    int sum = 0;
    for (int i=0; i<7; i++)
        sum += a[i];

    cout << "the average of these elements is " << sum/7.0 << endl;

    return 0;
}
```

```
the average of these elements is 4
```

```
#include <iostream>
using namespace std;

int main() {
    int a[7] = { 6, 7, 3, 5, 2, 1, 4 };

    int min = a[0];
    int max = a[0];

    for (int i=1; i<7; i++) {

        if (a[i] < min)
            min = a[i];

        if (a[i] > max)
            max = a[i];
    }

    cout << "the smallest of these elements is " << min << endl;
    cout << "the largest of these elements is " << max << endl;

    return 0;
}
```

```
the smallest of these elements is 1
the largest of these elements is 7
```