// Compute Greatest Common Divisor (GCD) using Euclid’s algorithm
// George F. Riley, Spring 2016

#include <iostream>
using namespace std;

// First declare some "global" variables
// Where in memory do these variables go?
int g1 = 10;
int g2;

int Euclid(int m, int n)
{
    int Ans = 0;
    while (Ans == 0)
    {
        // Loop until algorithm terminates
        int r = m % n; // Remainder when m divided by n
        // Note the double equal sign to check for "equivalence.
        // This is different than the single equal sign for assignment
        if (r == 0) // See if algorithm has terminated
        {
            // Algorithm has terminated, n is the answer
            Ans = n;
        }
        else
        {
            // Interchange
            m = n;
            n = r;
        }
    }
    return Ans;
}

int main(int argc, char ** argv)
{
    int Ans;
    // Illustrate the function call to function "Euclid"
    if (argc < 3)
    {
        Ans = Euclid(119, 544);
    }
    else
    {
        int m = atol(argv[1]);
        int n = atol(argv[2]);
        Ans = Euclid(m, n);
    }
    cout << "GCD is " << Ans << endl;

    // Illustrate a few more C syntax examples
    // First of course is the comments delimited by double shash //
    // Print the two global variables, g1 and g2
    cout << "g1 " << g1 << " g2 " << g2 << endl;

    Program gcd.cc
// Next declare two integer variables with initial values adn print them
// with cout and endl
int i = 100;
int x = -14;
unsigned int y = 12;
cout << "i " << i << " x " << x << " y " << y << endl;

// Declare integer variables without initializers
int x1;
int y1;
cout << "x1 is " << x1 << " y1 " << y1 << endl;

// Illustrate looping with "for" loop and "while"
// Be sure to notice that we used variable name "i" agin
// but it seems to work.
// The "for" loop has three fields:
// 1) Initializer..happens once
// 2) Loop exit check.. happens at the end of each iteration
for (int i = 0; i < 10; i++)
{
    cout << "Hello from for loop i is " << i << endl;
}
// What will happen if we print variable i here?
cout << "i is " << i << endl;
// Illustrate looping with while loop
while(y > 0)
{
    cout << "y is " << y << endl;
}
cout << "End of while loop" << endl;
// Illustrate arrays with and without initializer
int a[10]; // Variable a is an array with 10 elements but uninitialized
int b[] = { 10, 20, 30, 40, 50 };

// Print out a array, but the code below is potentially buggy
for (int i = 0; i < 10; ++i)
{
    cout << "a[" << i << "] is " << a[i] << endl;
}
// Print out b
for (int i = 0; i < 5; ++i)
{
    cout << "b[" << i << "] is " << b[i] << endl;
}
}