Example Generating Prime Numbers using Sieve of Eratosthenes Algorithm

acarlstein.com/

Posted by Alejandro G. Carlstein Ramos Mejia on October 15, 2010 October 15, 2010 About Programming / Algorithms / C++

Example of how to generate prime numbers using Sieve of Eratosthenes algorithm.

NOTIFICATION: These examples are provided for educational purposes. Using this code is under your own responsibility and risk. The code is given ‘as is’. I do not take responsibilities of how they are used.

primes.cpp:

```
#include<iostream>
#include<vector>
#include<stdio.h>
#include<ctime>
#include<math.h>
#include<string.h>
using namespace std;

//
// generatePrimesSlow()
// The array arr store the prime numbers up to and including x
// To test if a number i is prime, see if any prime number
// before i divides i evenly. If no prime number does, i is prime
// and is added to our array of prime numbers.
// Note: We should never see a number less than 2 in our array!
//
void generatePrimesSlow(int x, int * arr, int * amount)
{
   int count = 0; // numbers currently in the array

   for(int i=2; i<=x; i++) {
      bool isPrime = true;
      for (int j=0; j<count; j++) {
         if (i % arr[j] == 0) {
            isPrime = false;
            break;
  }
      }
      if (isPrime) {
         arr[count] = i;
         count++;
      }
   }
   *amount = count;
}
```

```
//
// generatePrimesFast()
// The array arr stores our prime numbers up to and including x
// To generate these numbers, we use the the Sieve of Eratosthenes
// Visit http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
// for the explanation of the algorithm.
//
void generatePrimesFast(int x, int * arr, int * amount)
{
   int num = 2;
   int count = 0;
   bool *boolArr = new bool[x+1];
   //bool boolArr[x];
   memset(boolArr, true, (x+1)*sizeof(bool));
   for(int i=2; i<=x; i++) {
      if (boolArr[i]) {
         arr[count] = i;
         count++;
         int temp = i+i;
         while (temp <= x) {
            boolArr[temp] = false;
            temp += i;
         }
      }
   }
   *amount = count;
}

int main() {
   cout << 'Up to what number would you like to generate primes? ';
   int max;
   cin >> max;
   cout << endl;

   // This will be our array to hold the prime numbers.
   // Notice that this is probably way too much memory, since not every
   // number up to max will be prime
   // In fact, very very few numbers will be, let's try to free up some
   // memory by using some math
   // I think a bound for the number of primes up to N is N/lg(N)
   // Let's try it

   int amount = 0;
   int num_primes;
   if (max > 150) {
      double x = max;
      double actualNumberOfPrimes = x/(log(x)-4);
      //double actualNumberOfPrimes = x/(log(x));
      num_primes = (int) actualNumberOfPrimes;
   }
   else
      num_primes = max;
   int arr[num_primes+1];
   memset(arr,0,sizeof(arr));
   //generatePrimesFast(max, arr, &amount);
   generatePrimesSlow(max, arr, &amount);

   cout << 'Here are the prime numbers from 2 to ' << max << endl;
   for (int i=0; i < amount; i++) {
      cout << arr[i] << endl;
   }
}
```

If you encounter any problems or errors, please let me know by providing an example of the code, input, output, and an explanation. Thanks.