Sieve of Eratosthenes

Given a number n, print all primes smaller than or equal to n. It is also given that n is a small number.
For example, if n is 10, the output should be “2, 3, 5, 7″. If n is 20, the output should be “2, 3, 5, 7, 11, 13, 17, 19″.

The sieve of Eratosthenes is one of the most efficient ways to find all primes smaller than n when n is smaller than 10 million or so (Ref Wiki).

Following is the algorithm to find all the prime numbers less than or equal to a given integer *n* by Eratosthenes’ method:

1. Create a list of consecutive integers from 2 to *n*: (2, 3, 4, …, *n*).
2. Initially, let *p* equal 2, the first prime number.
3. Starting from *p*, count up in increments of *p* and mark each of these numbers greater than *p* itself in the list. These numbers will be 2*p*, 3*p*, 4*p*, etc.; note that some of them may have already been marked.
4. Find the first number greater than *p* in the list that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this number (which is the next prime), and repeat from step 3.

When the algorithm terminates, all the numbers in the list that are not marked are prime.

Following is C++ implementation of the above algorithm. In the following implementation, a boolean array arr[] of size n is used to mark multiples of prime numbers.

```
#include <stdio.h>
#include <string.h>

// marks all mutiples of 'a' ( greater than 'a' but less
void markMultiples(bool arr[], int a, int n)
{
    int i = 2, num;
    while ( (num = i*a) <= n )
    {
        arr[ num-1 ] = 1; // minus 1 because index start
        ++i;
    }
}

// A function to print all prime numbers smaller than n
void SieveOfEratosthenes(int n)
{
    // There are no prime numbers smaller than 2
    if (n >= 2)
    {
        // Create an array of size n and initialize all
        bool arr[n];
        memset(arr, 0, sizeof(arr));

        /* Following property is maintained in the below
           arr[i] == 0 means i + 1 is prime
           arr[i] == 1 means i + 1 is not prime */
        for (int i=1; i<n; ++i)
        {
            if ( arr[i] == 0 )
            {
                //(i+1) is prime, print it and mark its
                printf("%d ", i+1);
                markMultiples(arr, i+1, n);
            }
        }
    }
}

// Driver Program to test above function
int main()
{
    int n = 30;
    printf("Following are the prime numbers below %d\n",
    SieveOfEratosthenes(n);
    return 0;
}
```

Output:

```
Following are the prime numbers below 30
2 3 5 7 11 13 17 19 23 29
```