Sieve of Eratosthenes Algorithm

The Sieve of Eratosthenes is an efficient algorithm to find all prime numbers up to a given integer N. It works by marking the multiples of each prime number starting from 2. After marking all multiples of each prime number, the numbers that remain unmarked are prime.

Steps:

1. **Initialize:** Create a boolean array of size N+1, with all values initially set to true, indicating that every number is "assumed" to be prime.
2. **Mark Non-Primes:** Starting from the smallest prime number (2), mark all multiples of each prime as non-prime. For each number that remains true (indicating it's a prime), mark all its multiples as false.
3. **Collect Primes:** Finally, the numbers that are still true in the array are prime numbers.

C++ Code Implementation:

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

vector<int> sieveOfEratosthenes(int N) {
    vector<bool> isPrime(N + 1, true);
    vector<int> primes;

    isPrime[0] = isPrime[1] = false; // 0 and 1 are not prime numbers

    for (int i = 2; i * i <= N; i++) {
        if (isPrime[i]) {
            for (int j = i * i; j <= N; j += i) {
                isPrime[j] = false; // Mark multiples of i as non-prime
            }
        }
    }

    for (int i = 2; i <= N; i++) {
        if (isPrime[i]) {
            primes.push_back(i);
        }
    }

    return primes;
```

```
}

int main() {
    int N;
    cout << "Enter the upper limit N: ";
    cin >> N;

    vector<int> primes = sieveOfEratosthenes(N);

    cout << "Prime numbers up to " << N << " are: ";
    for (int prime : primes) {
        cout << prime << " ";
    }
    cout << endl;

    return 0;
}
```