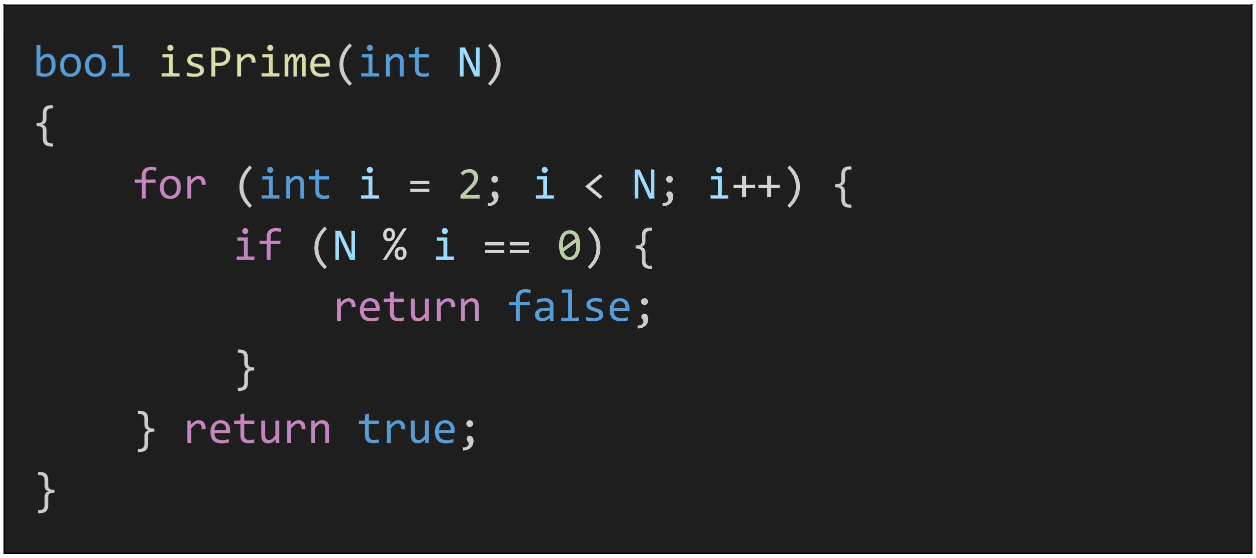
Primality Test – C++ [Algorithm For Prime Number]

1 ব্যাতিত যে সকল ধনাত্মক পূর্ণসংখ্যা শুধুমাত্র 1 এবং সেই সংখ্যাটি নিজে দ্বারা বিভাজ্য হয় এবং অন্য কোনো সংখ্যা দ্বারা নিঃশেষে বিভাজ্য হয় না, সেই সকল সংখ্যাই মৌলিক সংখ্যা (Prime Number) হিসেবে পরিচিত । যেমন, 17 একটি মৌলিক সংখ্যা কারণ 17 কেবলমাত্র 1 এবং 17 দ্বারা ভাগ করা যায়, অন্য কোনো সংখ্যা দ্বারা ভাগ করলে নিঃশেষে ভাগ করা সম্ভব হয় না ।

ধরি কোনো একটি সংখ্যা **N**, এখন **N** এই সংখ্যাটি মৌলিক কি না তা যাচাই করার জন্য প্রোগ্রামিং এ আমরা যে কাজটি করতে পারি, **2** থেকে শুরু করে **N-1** পর্যন্ত যতগুলো পূর্ণসংখ্যা পাওয়া যাবে তার মধ্যে কোনো সংখ্যা দ্বারাই যদি **N** নিঃশেষে ভাগ না যায় তবে **N** একটি মৌলিক সংখা হবে । নিম্নে একটি বুলিয়ান ডাটা টাইপের ফাংশন **isPrime()** এর সাহায্যে আলোচিত বিষয়টি কোড করা হলো । উল্লেখ্য, ফাংশনটি যদি **true** রিটার্ন করে তবে সেই সংখ্যাটি অর্থাৎ **N** একটি Prime Number আর **false** রিটার্ন করলে Prime Number না ।



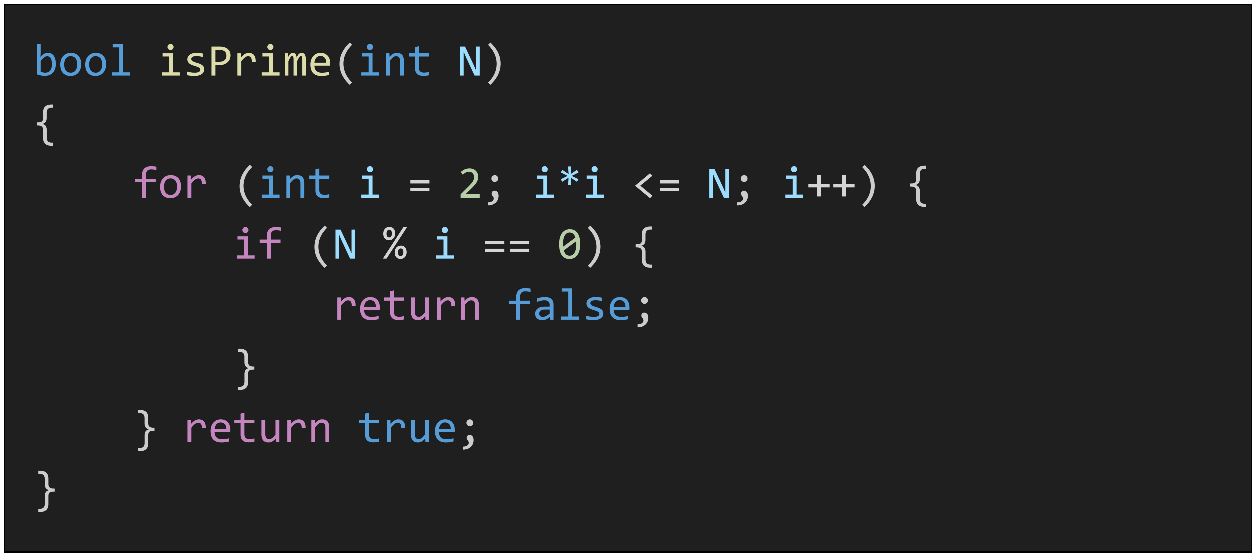
```
bool isPrime(int N)
{
    for (int i = 2; i < N; i++) {
        if (N % i == 0) {
            return false;
        }
    } return true;
}
```

এখন, এখানে একটা সমস্যা হলো যদি N বড় কোনো সংখ্যা হয় (যেমন, 100000007) তখন N যদি Prime Number হয় তবে, Loop টা ঠিক 2 থেকে শুরু করে N-1 অবধি চলবে ! এতে তুলনামূলক সময়ও বেশি লাগবে । এই সমস্যা সমাধানের জন্য / কোড টা কে আরো efficient করার জন্য আমরা এখানে কিছু জিনিস বিশ্লেষণ করে, যেন কম সময়ে এবং কম Loop চালিয়ে N সংখ্যাটি মৌলিক সংখ্যা কি না তা যাচাই করতে পারি ।

ধরি N = 24 । আমরা জানি, 24 কোনো মৌলিক সংখ্যা না কারণ 1 এবং 24 ব্যাতিত 2, 3, 4, 6, 8 ও 12 এই সংখ্যা গুলো দিয়েও 24 কে নিঃশেষে ভাগ করা যায় । এখানে আমরা যদি গুণিতক সংখাগুলি খেয়াল করি তবে, (2×12) = 24 এখানে 2 আর 12 গুণ করলে 24 পাওয়া যায় এবং 2 আর 12 এর মধ্যে ছোট গুণিতক সংখ্যাটি 2 । আবার, (3×8) = 24 এখানে 3 আর 8 গুণ করলে 24 পাওয়া যায় এবং ছোট গুণিতক সংখ্যাটি 3 । একইভাবে, (4×6) = 24 এখানে 4 আর 6 গুণ করলে 24 পাওয়া যায় এবং ছোট গুণিতক সংখ্যাটি 4 । সুতরাং ছোট গুণিতক সংখ্যাগুলি হলো 2, 3, 4 এবং এদের মধ্যে 4 হলো Maximum । এখন, N এর value ছিলো 24 এবং এই 24 কে যদি আমরা Square Root করি **sqrt(24)** তবে integer value পাবো 4 যা, ছোট গুণিতক সংখ্যাগুলির মধ্যে Maximum ছিলো ।

(প্রশ্ন: sqrt(24) = 4 কিভাবে হলো, দশমিকের পরের মান গুলা কই গেলো ? উত্তর: আমরা sqrt(24) এর মানকে একটি integer variable এ store করতেছি । তাই সংখ্যাটির বর্গমূলের মান ভগ্নাংশে / float data type এর ভেরিয়েবলে না বরং integer data type এর ভেরিয়েবলে store হবে । 24 এর বর্গমূল হলো 4.898979486 । যেহেতু এই value টা আমরা একটি integer type variable এ assign করতেছি তাই সেখানে শুধু 4 store হবে, তাই sqrt(24) = 4)

অর্থাৎ কোনো নাম্বার N, মৌলিক সংখ্যা কি না তা যাচাইয়ের জন্য আমরা এখন 2 থেকে শুরু করে sqrt(N) অবধি Loop চালিয়ে চেক করলেই পারি, তাই না ?

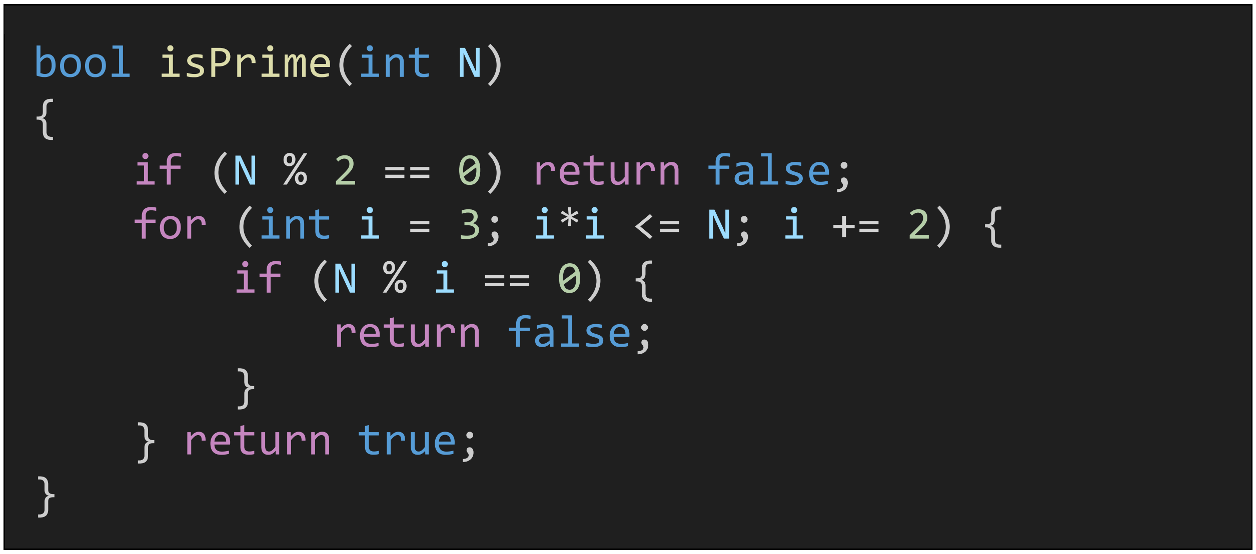


```
bool isPrime(int N)
{
    for (int i = 2; i*i <= N; i++) {
        if (N % i == 0) {
            return false;
        }
    } return true;
}
```

এখানে Loop এর ভেতরে condition এ আমাদের **i <= sqrt(N)** লিখা যে কথা, **i*i <= N** লিখা একই কথা । এভাবে লিখার কারণ হলো, যেহেতু **sqrt()** একটি built in Function

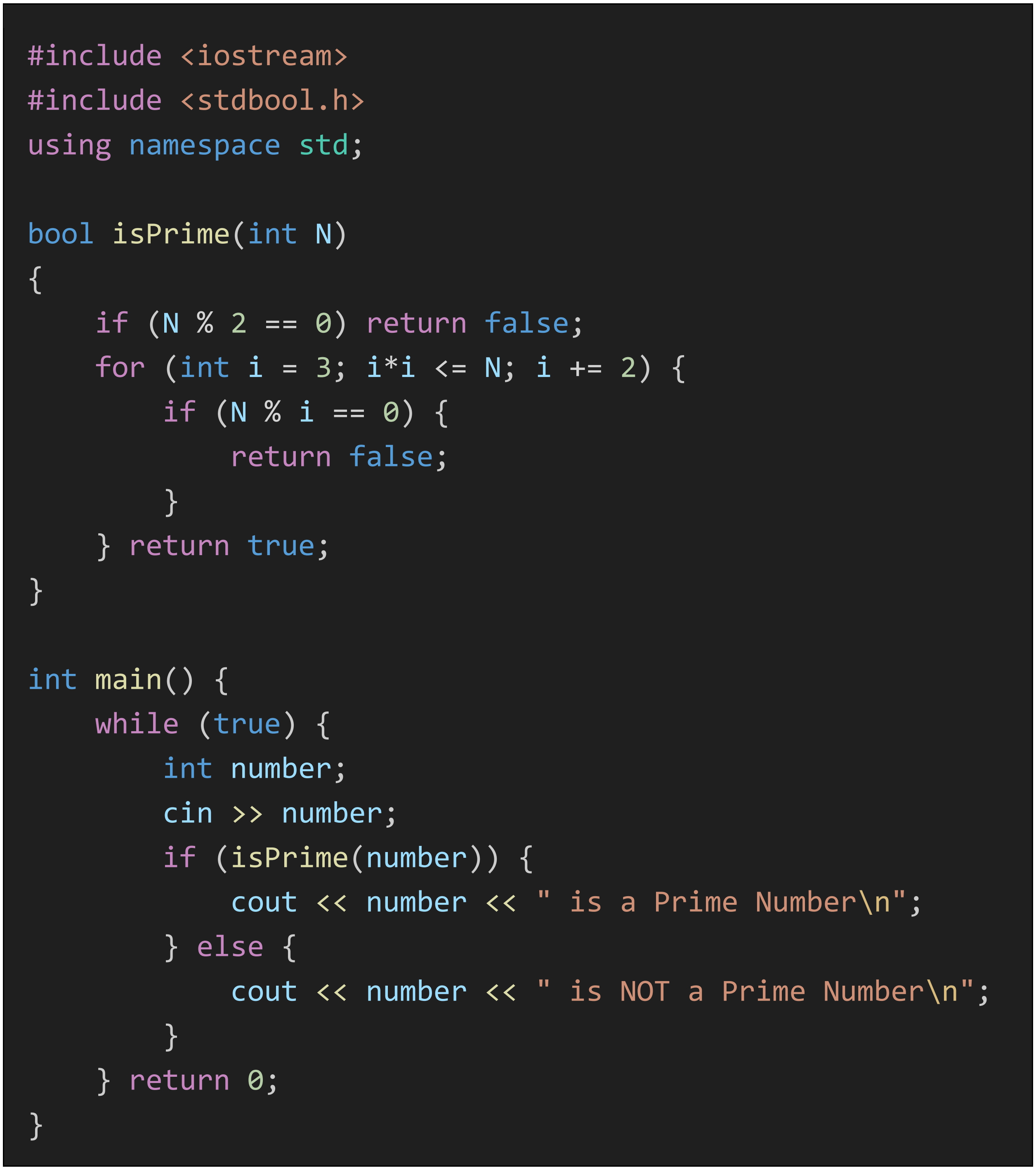
math.h header File এর under এ, তাই এটি ব্যাবহার করতে গেলে আমাদের আবার সেই header file টা include করতে হবে । তাই বাপারটা আরও সহজ এবং কম ঝামেলায় করার জন্য আমরা **i*i < N** এভাবে লিখবো (কারণ, ঝামেলা যত কম হবে, কোড তত বেশি efficient হবে) ।

এখান অবধি সবটুকু ঠিক আছে তবে, আমরা চাইলে কোডটি আরও efficient করতে পারি ! (কিভাবে ?) এখানে খেয়াল করলে দেখা যাবে যে Loop টা শুরু হচ্ছে 2 থেকে এবং **i** এর মান বৃদ্ধি পাচ্ছে এক (**i++** increment operation) করে । অর্থাৎ **i** এর value যথাক্রমে 2, 3, 4, 5, 6, 7, 8 … এক এক করে বাড়তে বাড়তে sqrt(N) পর্যন্ত Loop চলতেছে । কিন্তু এটাও তো সত্য যে, কোনো সংখ্যা যদি 2 দ্বারা ভাগ না যায় তাহলে ঐ সংখ্যাটি আর অন্য কোনো জোড় সংখ্যা দ্বারা ভাগ যাবে না । তাই Loop এর ভেতরে 4, 6, 8, 10, 12 … ইত্যাদি জোড় সংখ্যা দ্বারা **N** ভাগ যাচ্ছে কি না তা চেক করা বৃথা ! তাই আমরা ফাংশনের ভিতর প্রথমেই **if** condition দিয়ে একবার চেক করে নিবো যে **N** কি 2 দ্বারা ভাগ যাচ্ছে কি না ! যদি ভাগ যায় তাইলে **false** রিটার্ন করবে । আর Loop এর ভেতরে **i** চলবে 3 থেকে এবং **i** এক এক করে বৃদ্ধি না করে 2 করে বৃদ্ধি করা হবে **i += 2** (অর্থাৎ সকল বিজোড় সংখ্যা দ্বারা চেক করা হবে এখন) ।



```
bool isPrime(int N)
{
    if (N % 2 == 0) return false;
    for (int i = 3; i*i <= N; i += 2) {
        if (N % i == 0) {
            return false;
        }
    } return true;
}
```

Source Code



```cpp
#include <iostream>
#include <stdbool.h>
using namespace std;

bool isPrime(int N)
{
    if (N % 2 == 0) return false;
    for (int i = 3; i*i <= N; i += 2) {
        if (N % i == 0) {
            return false;
        }
    } return true;
}

int main() {
    while (true) {
        int number;
        cin >> number;
        if (isPrime(number)) {
            cout << number << " is a Prime Number\n";
        } else {
            cout << number << " is NOT a Prime Number\n";
        }
    } return 0;
}
```