

Luis Alberto Vargas González.

Práctica 1.3 Criba de Eratóstenes.

Fecha: 20/09/2022.

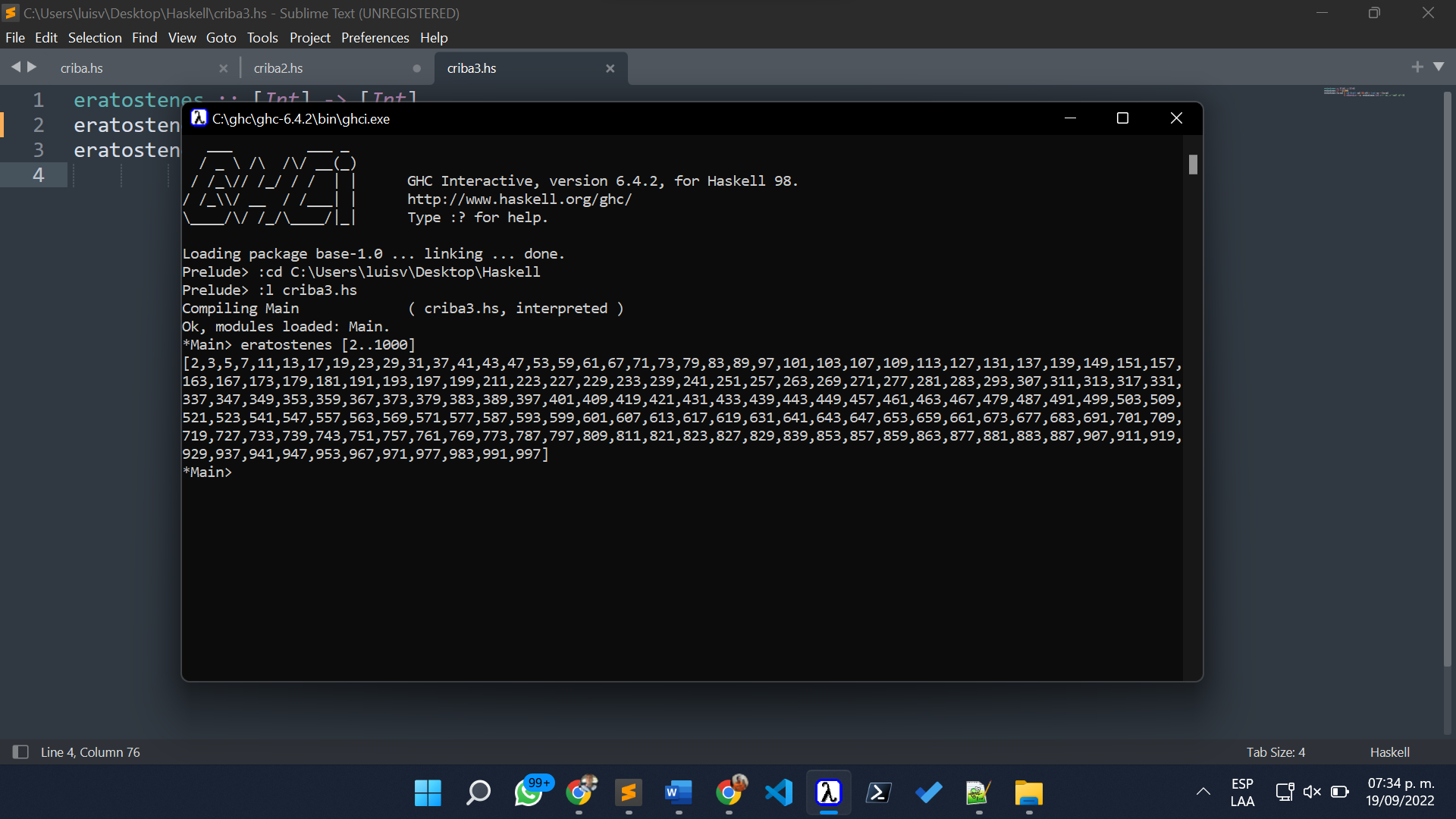
Programación Funcional y Lógica.

**Criba de Eratóstenes en Haskell.**

eratostenes [] = [1000]

eratostenes (x:xs) | not (null xs) && x^2 > last xs = (x:xs)

| otherwise = x: eratostenes [y| y <- xs, y `mod` x/= 0]



**Criba de Eratóstenes en C++.**

**#include <iostream>**

**#include <cmath>**

**using namespace std;**

**void criba(int n)**

**{**

**int a[n + 1];**

**int b=0;**

**for (int i = 0; i <= n; i++) {**

**a[i] = 1;**

**}**

**for (int i = 2; i <= sqrt(n); i++)**

**{**

**if (a[i] == 1)**

**{**

**for (int j = 2; i \* j <= n; j++) {**

**a[i \* j] = 0;**

**}**

**}**

**}**

**for (int i = 2; i <= n; i++)**

**{**

**if (a[i] == 1 ) {**

**cout << i<< " ";**

**}else(a[i]==0);{**

**cout<<b<<" ";**

**}**

**}**

**}**

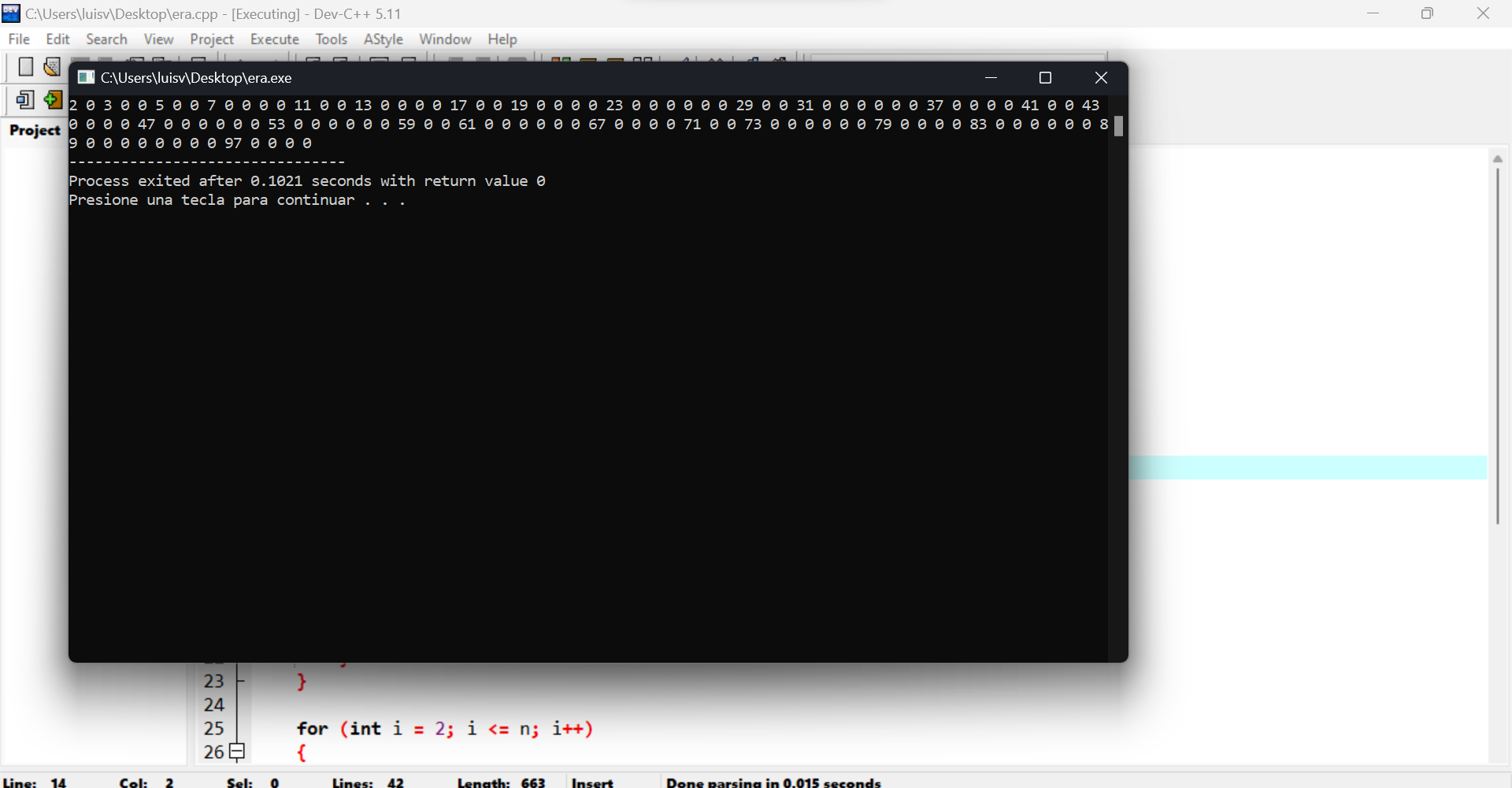
**int main()**

**{**

**criba(100);**

**return 0;**

**}**

****

**Conclusiones**

Ha sido compleja la programación del algoritmo de Criba de eratostenes , sobre todo por el uso de los vectores como para hacerlos funcionar como arreglos unidimensionales, que es justo lo que se sugiere hacer para el programa.

Como se puede claramente apreciar el programa hecho en c++ es mucho mas largo de implementar y de entender por ende , por lo que el uso de lenguajes declarativos que no contengan el uso de una estructuras tan definidas como los imperativos, en donde si no se hace de dicha manera, los programas no se pueden resolver y correr.