

Victor Antonio Silva de Queiroga

RA: 22117054-1

GitHub: <https://github.com/VittQ/SpeedUp2>

Atividade de Laboratório: SpeedUp 2

1. Detalhe o algoritmo utilizado para determinar os números primos e a soma deles.

De acordo com dados coletados no SpeedUp 1, vimos que o TestaPrimo4 obteve melhores resultados, por esse motivo ele foi o escolhido para este exercício com threads. O professor forneceu a função TestaPrimo4, porém foi necessário fazer algumas alterações para que ela fosse adaptada para funcionar com threads.

Foram elas: Agora a função recebe dois parâmetros, assim é possível dividir metade do trabalho para cada thread, a primeira thread vai de p até a metade dele, ou seja, $p/2$. A segunda thread vai da metade de $p/2$ até 0. Lembrando que o for neste caso trabalha de forma decrescente. Também foi acrescentado um if dentro da TestaPrimo para que pudéssemos controlar até onde o for percorreria, desta forma ele não trabalha desnecessariamente depois que os 221 primos forem encontrados.

```
// Victor Antonio Silva de Queiroga
// RA: 22117054-1

#include <iostream>
#include <stdlib.h>
#include <stdio.h>
#include <chrono>
#include <thread>

using namespace std;
using namespace chrono;
int soma = 0;
int primos = 0; //variavel primos para controlar o for

void TestaPrimo4(int n, int f) {
    for (int i = n; i > f; i--) {
        int EhPrimo,
            d = 3;
        if (n <= 1 || (n != 2 && n % 2 == 0) || (n % 6 != 1 && n % 6 != 5))
            EhPrimo = 0;
        else
            EhPrimo = 1;
        while (EhPrimo && d <= n / 2) {
            if (n % d == 0)
                EhPrimo = 0;
            d = d + 2;
        }

        //If para controlar a variavel soma e saber quantos primos foram encontrados
        if (EhPrimo != 0) {
            soma = soma + n; // 20044239
            printf("\n %d - %d", primos+1, n);
            primos++;
        }
        else if (primos == 221) {
            i = 0;
        }

        n--;
    }
}
```

2. Detalhe o tempo de execução de cada processo e mostre como a distribuição aumentou o SpeedUp (quando comparada com a execução de todo o projeto em um único processo).

Quando o algoritmo foi executado sem usar threads obtemos o tempo de execução de 138ms. Já quando as threads dividiram a execução o tempo obtido foi 130ms.

Para isso utilizei a biblioteca chrono.

Sem thread:

```
A soma foi: 20044239
Tempo de execucao: 138ms
```

Com thread:

```
A soma foi: 20044239
Tempo de execucao: 130ms
```

```
//22117054 - 1
// aaa = 221
// bbb = 170
// ccc = 541

int main()
{
    int a = 221; // valor para aaa
    int b = 170; // valor para bbb
    int c = 541; // valor para ccc
    int p = b * c; // = 91.970

    auto start = std::chrono::steady_clock::now(); //Início do contador

    //Com thread
    std::thread thread1(TestaPrimo4, p, (p / 2)); //Chamando primeira thread
    thread1.join();
    std::thread thread2(TestaPrimo4, ((p / 2) - 1), 0); //Chamando segunda thread
    thread2.join();

    //Sem Thread
    //TestaPrimo4(p,0); //Função sem thread

    auto end = std::chrono::steady_clock::now(); //Final do contador
    milliseconds tempexec = duration_cast<milliseconds>(end - start);

    cout << "\n\nA soma foi: " << soma << endl;
    cout << "\nTempo de execucao: " << tempexec.count() << "ms" << endl ;
}
```

3. Coloque também em uma folha anexa no final do relatório, a lista dos números primos encontrados em ordem crescente.

O algoritmo foi construído de forma decrescente.

1 - 91969
2 - 91967
3 - 91961
4 - 91957
5 - 91951
6 - 91943
7 - 91939
8 - 91921
9 - 91909
10 - 91873
11 - 91867
12 - 91841
13 - 91837
14 - 91823
15 - 91813
16 - 91811
17 - 91807
18 - 91801
19 - 91781
20 - 91771
21 - 91757
22 - 91753
23 - 91733
24 - 91711
25 - 91703
26 - 91691
27 - 91673
28 - 91639

29 - 91631
30 - 91621
31 - 91591
32 - 91583
33 - 91577
34 - 91573
35 - 91571
36 - 91541
37 - 91529
38 - 91513
39 - 91499
40 - 91493
41 - 91463
42 - 91459
43 - 91457
44 - 91453
45 - 91433
46 - 91423
47 - 91411
48 - 91397
49 - 91393
50 - 91387
51 - 91381
52 - 91373
53 - 91369
54 - 91367
55 - 91331
56 - 91309
57 - 91303
58 - 91297
59 - 91291

60 - 91283
61 - 91253
62 - 91249
63 - 91243
64 - 91237
65 - 91229
66 - 91199
67 - 91193
68 - 91183
69 - 91163
70 - 91159
71 - 91153
72 - 91151
73 - 91141
74 - 91139
75 - 91129
76 - 91127
77 - 91121
78 - 91099
79 - 91097
80 - 91081
81 - 91079
82 - 91033
83 - 91019
84 - 91009
85 - 90997
86 - 90989
87 - 90977
88 - 90971
89 - 90947
90 - 90931

91 - 90917
92 - 90911
93 - 90907
94 - 90901
95 - 90887
96 - 90863
97 - 90847
98 - 90841
99 - 90833
100 - 90823
101 - 90821
102 - 90803
103 - 90793
104 - 90787
105 - 90749
106 - 90731
107 - 90709
108 - 90703
109 - 90697
110 - 90679
111 - 90677
112 - 90659
113 - 90647
114 - 90641
115 - 90631
116 - 90619
117 - 90617
118 - 90599
119 - 90583
120 - 90547
121 - 90533

122 - 90529
123 - 90527
124 - 90523
125 - 90511
126 - 90499
127 - 90481
128 - 90473
129 - 90469
130 - 90439
131 - 90437
132 - 90407
133 - 90403
134 - 90401
135 - 90397
136 - 90379
137 - 90373
138 - 90371
139 - 90359
140 - 90353
141 - 90313
142 - 90289
143 - 90281
144 - 90271
145 - 90263
146 - 90247
147 - 90239
148 - 90227
149 - 90217
150 - 90203
151 - 90199
152 - 90197

153 - 90191
154 - 90187
155 - 90173
156 - 90163
157 - 90149
158 - 90127
159 - 90121
160 - 90107
161 - 90089
162 - 90073
163 - 90071
164 - 90067
165 - 90059
166 - 90053
167 - 90031
168 - 90023
169 - 90019
170 - 90017
171 - 90011
172 - 90007
173 - 90001
174 - 89989
175 - 89983
176 - 89977
177 - 89963
178 - 89959
179 - 89939
180 - 89923
181 - 89917
182 - 89909
183 - 89899

184 - 89897
185 - 89891
186 - 89867
187 - 89849
188 - 89839
189 - 89833
190 - 89821
191 - 89819
192 - 89809
193 - 89797
194 - 89783
195 - 89779
196 - 89767
197 - 89759
198 - 89753
199 - 89689
200 - 89681
201 - 89671
202 - 89669
203 - 89659
204 - 89657
205 - 89653
206 - 89633
207 - 89627
208 - 89611
209 - 89603
210 - 89599
211 - 89597
212 - 89591
213 - 89567
214 - 89563

215 - 89561

216 - 89533

217 - 89527

218 - 89521

219 - 89519

220 - 89513

221 – 89501

A soma foi: 20044239

Tempo de execucao: 138ms