

paralelo_atomic_priv.c

```
1
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <omp.h>
6 #include <time.h>
7
8 // Definição global do número de passos para consistência
9 const long NUM_PASSOS = 100000000;
10 long pontos_no_circulo_total = 0;
11
12 //versão Paralela
13 void pi_paralel_for_critical_private() {
14     unsigned int seed = time(NULL);
15     long pontos_no_circulo_local = 0;
16
17     #pragma omp parallel firstprivate(pontos_no_circulo_local, seed)
18     {
19         #pragma omp for
20         for (long i = 0; i < NUM_PASSOS; i++){
21             double x = (double)rand_r(&seed) / RAND_MAX * 2.0 - 1.0;
22             double y = (double)rand_r(&seed) / RAND_MAX * 2.0 - 1.0;
23
24             if (x * x + y * y < 1.0) {
25                 pontos_no_circulo_local++;
26             }
27         }
28         #pragma omp atomic
29         pontos_no_circulo_total += pontos_no_circulo_local;
30
31     } // Fim da região paralela
32 }
33
34 int main() {
35     double start_time, end_time;
36
37     printf("Iniciando analise de desempenho para %ld passos.\n", NUM_PASSOS);
38     start_time = omp_get_wtime();
39     pi_paralel_for_critical_private();
40     end_time = omp_get_wtime();
41     double tempo_paralelo = end_time - start_time;
42     double pi_estimado = 4.0 * pontos_no_circulo_total / NUM_PASSOS;
43
44     printf("Estimativa paralela de pi = %f\n", pi_estimado);
45     printf("Tempo Paralelo: %f segundos\n", tempo_paralelo);
46
47     return 0;
48 }
49
```