paralelo_critical_priv.c

```c

////////////////////////////////////////////////
///////////////VERSÃO 03////////////////////////
////////////////////////////////////////////////

#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <time.h>

// Definição global do número de passos para consistência
const long NUM_PASSOS = 100000000;
long pontos_no_circulo_total = 0;

//versão Paralela
void pi_paralel_for_critical_private() {
    unsigned int seed = time(NULL);
    long pontos_no_circulo_local = 0;

    #pragma omp parallel firstprivate(pontos_no_circulo_local, seed)
    {
        #pragma omp for
        for (long i = 0; i < NUM_PASSOS; i++){
            double x = (double)rand_r(&seed) / RAND_MAX * 2.0 - 1.0;
            double y = (double)rand_r(&seed) / RAND_MAX * 2.0 - 1.0;

            if (x * x + y * y < 1.0) {
                pontos_no_circulo_local++;
            }
        }
        #pragma omp critical
        {
            pontos_no_circulo_total += pontos_no_circulo_local;
        }

    } // Fim da região paralela
}

int main() {
    double start_time, end_time;

    printf("Iniciando analise de desempenho para %ld passos.\n", NUM_PASSOS);
    start_time = omp_get_wtime();
    pi_paralel_for_critical_private();
    end_time = omp_get_wtime();
    double tempo_paralelo = end_time - start_time;
    double pi_estimado = 4.0 * pontos_no_circulo_total / NUM_PASSOS;

    printf("Estimativa paralela de pi = %f\n", pi_estimado);
    printf("Tempo Paralelo: %f segundos\n", tempo_paralelo);
```

```
51
52      return 0;
53  }
54
```