prog_paralel_for_critical.c

```c

#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 = 0;//variavel conpartilhada pelas thread

//versão Paralela
void pi_paralel_for_critical() {

    #pragma omp parallel
    {
        unsigned int seed_T = (unsigned int)time(NULL) ^ omp_get_thread_num();
        #pragma omp for
        for (long i = 0; i < NUM_PASSOS; i++){
            double x = (double)rand_r(&seed_T) / RAND_MAX * 2.0 - 1.0;
            double y = (double)rand_r(&seed_T) / RAND_MAX * 2.0 - 1.0;

            if (x * x + y * y < 1.0) {
                #pragma omp critical
                {
                    pontos_no_circulo++;
                }
            }
        }
    } // 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();
    end_time = omp_get_wtime();
    double tempo_paralelo = end_time - start_time;
    double pi_estimado = 4.0 * pontos_no_circulo / NUM_PASSOS;

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

    return 0;
}

```