Universidade de São Paulo
Instituto de Matemática e Estatistica
IME

miniEP IV - Programação Concorrente e Paralela

Patrícia da Silva Rodrigues (nºUSP 11315590),

A
2023

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10000000

double funcao() {
    clock_t inicio = clock();
    int acertos = 0;
    float x, y;
    for (int i = 0; i < N; i++) {
        x = (float)rand() / RAND_MAX;
        y = (float)rand() / RAND_MAX;
        if (x * x + y * y <= 1.0) {
            acertos++;
        }
    }
    double tempo_iteracao = (double)(clock() - inicio) / CLOCKS_PER_SEC;
    printf("%.6f\n", (4.0 * acertos) / N);
    return tempo_iteracao;
}

int main() {
    double tempo_total = 0.0;
    int qntd;
    printf("Quantas iteracoes quer testar: ");
    scanf("%d", &qntd);
    srand(time(NULL)); // Move srand para a main
    for (int i = 0; i < qntd; i++) {
        double tempo_iteracao = funcao();
        tempo_total += tempo_iteracao;
    }
    double tempo_medio = tempo_total / qntd;
    printf("Tempo medio: %.6f segundos.\n", tempo_medio);
    return 0;
}
```

Resultados: Quantas iteracoes quer testar: 10 3.140906 3.141755 3.141781 3.142541 3.141336 3.142938 3.140758 3.141702 3.142019 3.142216 Tempo medio: 0.338644 segundos.

Mudancas:

Arquivo de cabeçalho desnecessário removido stdlib.h.

A chamada de função srand() foi movida para fora da função funcao() para evitar a propagação do gerador de números aleatórios em cada chamada de função.

Este código pode ser considerado otimizado em termos de execução sequen-

cial, pois já está utilizando um algoritmo rápido para estimar o valor de pi, evita alocação desnecessária de memória e inicializa a semente do gerador de números aleatórios apenas uma vez.

No entanto, é importante observar que esse código ainda pode ser melhorado usando o paralelismo. Ao usar vários threads, podemos dividir a carga de trabalho entre eles e executar o cálculo simultaneamente, reduzindo potencialmente o tempo geral de execução.