Atividade Prática 02

Aluno: Raquel Gonçalves Rosa

Matrícula: 2020105815

1. OBJETIVO

O objetivo dessa atividade prática é avaliar e comparar o desempenho de códigos recursivos e iterativos para a execução de uma mesma tarefa.

Além disso, durante as atividades, serão desenvolvidas habilidades e conhecimentos para mensurar o desempenho de um código em C - isto é, quanto tempo é gasto para compilar e executar o código.

2. DESENVOLVIMENTO

Foram codificadas 4 funções, disponíveis na biblioteca func.h.

```c
#ifndef FUNCH
#define FUNCH

int fatorialIterativo(int n);
int fatorialRecursivo(int n);
int fibonacciIterativo(int n);
int fibonacciRecursivo(int n);

#endif
```

Além dessas funções, foram desenvolvidas mais três funções para o cálculo de tempo de execução. Elas estão disponíveis na biblioteca custo tempo.h

```c
#ifndef TEMPOH
#define TEMPOH

#include <sys/resource.h>
#include <time.h>
#include <inttypes.h>

#define MILISECONDS_OPTION 0
#define MICROSECONDS_OPTION 1
#define NANOSECONDS_OPTION 2

int64_t getUnixTimestamp(int option);

float difUserTime(struct rusage *start, struct rusage *end);
float difSystemTime(struct rusage *start, struct rusage *end);

#endif
```

3. PLANO DE EXPERIMENTOS

Para testar o desempenho do algoritmo de Fibonacci, o vetor A foi arbitrariamente escolhido para ser um vetor de 10 inteiros, dado por:

$$A = [1, 3, 5, 10, 15, 20, 25, 30, 35, 40]$$

Esse valores de A permitem testar uma faixa de tempo de execução variada, sem tomar tempo demais esperando códigos muito lentos (por exemplo: termo de posição 80 da sequência de Fibonacci) terminarem.

Para o teste de desempenho do algoritmo do fatorial, é usado o vetor B dado por:

$$B = [100, 300, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000]$$

Como o algoritmo de recursividade do fatorial é menos complexo que o algoritmo recursivo de Fibonacci, foram usados valores maiores para ver diferenças mais expressivas nos resultados.

O plano de experimentos consiste em iterar sobre todos os elementos do vetor A ou B, e em cada iteração será calculado o i-ésimo termo da sequência de fibonacci ou o valor de $i!$ com as funções descritas acima. Para cada operação, será calculado os seguidos valores de tempo transcorridos desde o início e fim da chamada da função que calcula o Fibonacci ou o Fatorial:

- Tempo de relógio: a diferença em microssegundos entre o início da operação e o fim da operação. Calculado via função getUnixTimestamp();
- Tempo de usuário: a diferença em microssegundos entre o tempo de usuário do início da operação e o fim da operação. Calculado via função difUserTime();
- Tempo de sistema: a diferença em microssegundos entre o tempo de sistema do início da operação e o fim da operação. Calculado via função difSystemTime();

Cabe aqui destacar, a diferença entre o tempo de sistema e o tempo de usuário. O tempo de usuário corresponde à quantidade de tempo que o programa gastou efetuando cálculos e operações no código, enquanto o tempo de sistema é a quantidade de tempo que o programa ficou esperando o kernel do sistema

operacional responder e efetuar operações de sistema, tais como printar conteúdos na tela, ler arquivos salvos em disco, etc. Assim, o tempo total gasto pela operação será a soma do tempo de usuário com o tempo de sistema.

4. RESULTADOS

A Tabela 1 mostra os resultados obtidos para os cálculos de Fatorial, enquanto a Tabela 2 mostra os resultados obtidos para os cálculos de Fibonacci.

Tabela 1. Medições de tempo para os algoritmos recursivo e iterativo de cálculo do Fatorial.

i	Tempo de relógio iterativo (µs)	Tempo de usuário iterativo (µs)	Tempo de sistema iterativo (µs)	Tempo de relógio recursivo (µs)	Tempo de usuário recursivo (µs)	Tempo de sistema recursivo (µs)
100	0	1	1	-213	1	1
300	0	1	1	-595	1	1
500	-2	1	1	-990	1	1
1000	-3	1	1	-1983	1	1
1500	-5	1	1	-3426	1	1
2000	-6	1	1	-3987	1	1
2500	-7	1	1	-5697	1	1
3000	-9	1	1	-6040	1	1
3500	-10	1	1	-7327	1	1
4000	-12	1	1	-7933	1	1

Tabela 2. Medições de tempo para os algoritmos recursivo e iterativo de Fibonacci.

i	Tempo de relógio iterativo (µs)	Tempo de usuário iterativo (µs)	Tempo de sistema iterativo (µs)	Tempo de relógio recursivo (µs)	Tempo de usuário recursivo (µs)	Tempo de sistema recursivo (µs)
1	140727021246528	1	1	0	1	1
3	140727021247560	1	1	140035221632596	1	1
5	94652015310049	1	1	94281985615073	1	1
10	94652015324512	1	1	94281985614989	1	1
15	140386213867584	1	1	140730186169512	1	1
20	140386212285163	1	1	20338384224	1	1
25	140386213433728	1	1	140730186169552	1	1
30	0	1	1	-4294967296	1	1
35	140727021246544	1	1	140730186169232	1	1
40	140727021246544	1	1	94281985615154	1	1