**UENF**

Universidade Estadual do Norte Fluminense Darcy Ribeiro

**Curso:** Ciência de Computação **Data:** 04/04/2023 **Atividade:** LE1 **Período: 3**º

**Disciplina:** Estrutura de Dados I

**Professor:** Fermín Alfredo Tang

**Turno:** Diurno

**Nome do aluno:** [Mariana Cossetti Dalfior](mailto:20211100064@pq.uenf.br) / Enzo Picanço Alberoni

**Matrícula:** 20211100064 / 20211100053

**Questões da LE1**

1.- (2,0 Pontos) Considerando questões de alocação de memória e estruturas de dados, tomando como base o capítulo 2, do livro: “Estrutura de Dados com Algoritmos e C - Marcos Laureano”, realize as seguintes tarefas:

i) Implementar os programas 2.4, 2.5, 2.7 e 2.8, corrigindo caso necessário;

ii) Explique a diferença entre o programa 2.7 e 2.8 no tratamento das matriz.

R: A diferença entre os códigos está na alocação de memória para cada matriz. Na matriz do programa 2.7 a função chamada de “aloca” ela inicializa e aloca memória para a matriz de duas dimensões de forma dinâmica e retorna um ponteiro para essa matriz que foi alocada. E já a matriz no programa 2.8 a função chamada “aloca” faz a alocação apenas da memória necessária, já que ela recebe um ponteiro que já foi definido anteriormente, ou seja a matriz já foi criada. Resumindo, o programa 2.7 faz a inicialização da matriz e a alocação de memória para a mesma, já o programa 2.8 ele apenas aloca a memória que é fundamental para guardar a matriz.

2.- (2,0 Pontos) Considerando os conceitos sobre TADs, consulte o capítulo 4 do livro: Estruturas de Dados Descomplicada – Andrés Backes, e realize as seguintes tarefas:

i) Implementar um TAD que represente um cilindro.

a) Defina a estrutura para armazenar os dados do cilindro (posição, raio e altura);

b) Defina as funções para criar/inicializar uma instancia de cilindro; alterar os dados do cilindro; funções que retornem a sua área e o seu volume; e uma função que destrua uma instância do cilindro.

ii) Teste o seu TAD com uma aplicação de exemplo, que permita criar pelo menos dois cilindros, modificar seus dados, e calcular seus volumes e áreas.



3.- (2,0 Pontos) Considerando os conceitos sobre eficiência de algoritmos, consulte o capítulo 3 de “Estruturas de Dados Descomplicada - Andrés Backes”, e realize as tarefas:

i. Implementar os algoritmos de ordenação: BubbleSort, SelectionSort e InsertionSort e teste cada algoritmo usando um vetor de tamanho n=10.

ii. Gerar um vetor aleatório de tamanho n, para n suficientemente grande, (ver a sugestão dada na Tabela);

iii. Realizar um experimento, executando os três algoritmos usando o mesmo vetor aleatório gerado em ii), e registre o tempo de ordenação de cada um (em milissegundos);

iv. Repetir o experimento, modificando os algoritmos de forma a contar o número de operações realizadas (comparações e atribuições) por cada algoritmo. Considere que cada troca equivale a 3 atribuições, e que um deslocamento equivale a uma atribuição.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tam. *n*** | **Bubble Sort** | | | **Selection Sort** | | | **Insertion Sort** | | |
|  | *Tempo*  *(ms)* | *N*°  *Comp* | *N*°  *Atrib* | *Tempo*  *(ms)* | *N*°  *Comp* | *N*°  *Atrib.* | *Tempo*  *(ms)* | *N*°  *Comp* | *N*°  *Atrib* |
| 50.000 | 10280 | 1249975000 | 1863866592 | 5724 | 1249975000 | 149997 | 5516 | 621288864 | 621338863 |

4.- (2,0 Pontos) Considerando os conceitos sobre eficiência de algoritmos, realize as seguintes tarefas:

i) Implementar as versões:

A) iterativa e B) recursiva do algoritmo para calcular o n êsimo número da série de Fibonacci. Teste cada algoritmo usando parâmetro n=5;

ii) Realizar um experimento, executando os dois algoritmos, para n suficientemente grande, de forma a identificar o limite de execução de cada algoritmo. Registre o tamanho do problema e o tempo de execução em cada caso, conforme a tabela.

|  |  |  |  |
| --- | --- | --- | --- |
| **Fibonacci Iterativo** | | **Fibonacci Recursivo** | |
| **Tam.**  **n** | ***Tempo***  ***(seg.)*** | **Tam.**  **n** | ***Tempo***  ***(seg.)*** |
| 10 | 0.010 | 10 | 0.009 |
| 20 | 0.011 | 20 | 0.011 |
| 30 | 0.011 | 30 | 0.024 |
| 40 | 0.010 | 40 | 1.249 |
| 50 | 0.010 | 50 | Muito tempo |
| 100 | 0.011 |  |  |

5.- (2,0 Pontos) Escreva um **algoritmo recursivo** para determinar se uma *string* é um palíndromo. Uma *string* é um palíndromo se pode ser lida para frente e parta trás com o mesmo significado. As letras maiúsculas e os espaços podem ser ignorados. Teste seu algoritmo com os seguintes palíndromos:

• A base do teto desaba.

• A cara rajada da jararaca.

• Acuda cadela da Leda caduca.

• A dama admirou o rim da amada.

• A droga do dote é todo da gorda.

E como pelo menos um caso que não seja palíndromo.

**Atribuído:** Em Duplas.

**Entrega:** Código Fonte

**Apresentar:** Execução e Explicação