Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Disciplina: Estrutura de Dados I
Professor: Jacson Luiz Matte

Trabalho NP2

Grupo: 2 pessoas

Data de entrega: 18/11/2016

Formato de entrega: apresentação do item (a) da avaliação e arquivo(s) entregues no Moodle, do item (b) da avaliação

Objetivo: Realizar uma pesquisa e implementar os algoritmos de ordenação de dados descritos abaixo utilizando como base um código incompleto.

Forma de entrega: O programa deve ser implementado em linguagem C, padrão ANSI. Todos devem submeter um arquivo nomeado .Caso alguém implementar com bibliotecas próprias, então deve submeter um arquivo compactado em formato .zip, contendo todas as bibliotecas necessárias, sob o nome .

Possibilidades de trabalho:

Opção 1: RadixSort e QuickSort

Implemente um programa com as especificações descritas a seguir, que permita a ordenação de dados pelos métodos **RadixSort e QuickSort.**

Para tal implementação deve ser reutilizado o programa **RadixQuick.c** que está com funções incompletas.

Além da implementação deve ser feito um pequeno relatório, contendo uma breve explicação, um exemplo prático e o pseudocódigo de cada algoritmo.

Opção 2: ShellSort e MergeSort

Implemente um programa com as especificações descritas a seguir, que permita a ordenação de dados pelos métodos **ShellSort e MergeSort.**

Para tal implementação deve ser reutilizado o programa com os arquivos **ShellMerge.c, ShellMerge.h** e **main.c** que estão com funções incompletas.

Além da implementação deve ser feito um pequeno relatório, contendo uma breve explicação, um exemplo prático e o pseudocódigo de cada algoritmo.

Especificações do programa

Cada método de ordenação deve ser implementado como uma função, seguindo o código base que deve ser "**completado**". Todos os métodos devem, obrigatoriamente, serem implementados usando listas duplamente ligadas e vetores.

O programa deve exibir um menu principal com as seguintes opções:

1) Criar Lista

2) Criar Vetor

3) Método 1

4) Método 2

5) Sair

Ao executar o programa, o usuário deve informar o tamanho da lista a ser criada (opção 1).

Já o tamanho do vetor deve ser fixado em 40 posições (opção 2).

Após esta entrada, o programa deve criar uma lista com o tamanho indicado, populada com dados que seguem a seguinte estrutura.

```
typedef struct _contato{
char nome[40];
char fone[30];
}TpContato;
```

Obs.: A inserção dos contatos deve ser feita de maneira automática, ou seja, dado o tamanho da lista, o programa deve criar contatos com nomes aleatórios.

Ex. de nomes: Fulano 123, Fulano 924, Fulano 837. Neste caso a constante Fulano será concatenada com um número gerado aleatoriamente (função rand()). O número do telefone também pode ser gerado aleatoriamente.

Após populada, a lista deve ser impressa na tela, um contato abaixo do outro, no seguinte formato:

Vetor:

Nome: Fulano 123
Fone: 92384739

Nome: Fulano 924
Fone: 92384222

Nome: Fulano 837
Fone: 33384222

Lista:

Nome: Fulano 123
Fone: 92384739

Nome: Fulano 924
Fone: 92384222

Nome: Fulano 837
Fone: 33384222
---

1) Criar Lista
2) Criar Vetor
3) Método 1
4) Método 2
5) Sair

Note que o menu principal foi exibido depois da impressão da lista.

A partir de agora, o usuário deverá escolher uma opção de ordenação (3 - 5). O programa deve fazer uma cópia da lista original e usar esta cópia como argumento para a função de ordenação escolhida. Após, o programa deve imprimir a lista ordenada segundo o algoritmo escolhido e, indicar o tempo decorrido em segundos.

Exemplo de saída vetor:

Nome: Fulano 123

Fone: 92384739

Nome: Fulano 837

Fone: 33384222

Nome: Fulano 924

Fone: 92384222

Tempo: 1 segs.

Exemplo de saída lista:

Nome: Fulano 123

Fone: 92384739

Nome: Fulano 837

Fone: 33384222

Nome: Fulano 924

Fone: 92384222

Tempo: 3 segs.

1) Criar Lista

2) Criar Vetor

3) Método 1

4) Método 2

5) Sair

Note que a ordenação foi realizada com base no campo NOME.

Dicas:

Comparar strings com strcmp

Concatenar nome + número aleatório com strcat

Avaliação

Serão utilizados 2 instrumentos de avaliação neste trabalho:

a) apresentação das estratégias de implementação e funcionamento da mesma correspondendo a 20% da nota, divididos em 10% para a organização da apresentação e 10% para o funcionamento;

b) (80% da nota). O restante da nota será a avaliação do professor em relação ao código, usabilidade, estruturação e funcionamento.

Observações: Para cada erro de indentação e *warning* será descontado 0.1 na nota do trabalho e caso não compile, a nota é zero.

Será compilado em ambiente Linux com o comando: gcc arq.c -o arq -Wall