Universidade Federal do Espíto Santo
Centro de Ciências Exatas, Naturais e de Saúde-CCENS
Departamento de Computação

COM10078-Estrutura de Dados-II	**Professor**	Dayan de Castro Bissoli
Período 2019-1	**Data de entrega:**	**07/06/2019**
10/05/2019	**Valor:**	10 Pontos

Algoritmos de Ordenação

O **objetivo** deste trabalho é implementar os algoritmos de ordenação estudados, comparando-os entre si.

1. Implementação em C de um programa gerador automático de números.

   Desenvolver um programa denominado *gera*, que deverá ter o seguinte comportamento:

   - Ao digitarmos:
     .\gera -a n.
     o programa deverá gerar n números em ordem **aleatória**.
   - Ao digitarmos:
     .\gera -c n.
     o programa deverá gerar n números em ordem **crescente**.
   - Ao digitarmos:
     .\gera -d n.
     o programa deverá gerar n números em ordem **decrescente**.

   Os números gerados deverão ser apresentados um por linha (sem espaço depois do número). Esses dados obtidos com o algoritmo *gera* serão utilizados para testar os algoritmos de ordenação da seção seguinte. Segue um exemplo:

   .\gera -c 5.
   1
   2
   3
   4
   5

2. Implementação em C dos doze seguintes algoritmos de ordenação:

   bolha, inserção direta, inserção binária, shellsort, seleção direta, heapsort, quicksort, mergesort, radixsort e bucketsort.*

   *: O algoritmo Quicksort utiliza um método de particionamento que, escolhido um elemento pivô, gera uma partição de elementos maiores ou iguais ao pivô e outra partição cujos elementos são menores ou iguais ao pivô. Consideraremos, neste trabalho, três formas de escolher o pivô:

   - O primeiro elemento do bloco (*quicksortini*);
   - O elemento central do bloco (*quicksortcentro*);
   - Mediana de 3 elementos, onde os elementos escolhidos para tal média serão o primeiro, o central e o último(*quicksortmediana*).

- Formatação de entrada e saída.
.\ordena [algoritmo] n [entrada.txt]
onde:

- *entrada.txt* será o arquivo com o resultado obtido com o algoritmo *gera* para n elementos;
- n é o número de elementos a serem ordenados
- *algoritmo* é o algoritmo que será utilizado na ordenação. As opções abrangem os doze algoritmos listados no início da Seção 2.

Como saída deve ser gerado o arquivo *saida.txt* com os números da entrada ordenados, sendo cada número em uma linha (sem espaço depois do número).

Um exemplo da utilização do trabalho escolhendo-se o algoritmo *bolha* para a ordenação de 1000 números:
.\ordena bolha 1000 entrada.txt

3. Documentação

- Como resultado deste trabalho deverá ser produzido um artigo técnico em LaTeX utilizando o padrão de artigos. Na documentação deverão ser apresentadas comparações entre os algoritmos com entradas de tamanho igual a 100, 1000 e 10000. Para discutir os resultados, construir gráficos e tabelas, apresentando o número de comparações, número de trocas e o tempo de execução do algoritmo para cada tipo de entrada proposto: aleatória, crescente e decrescente. Uma detalhada discussão sobre os resultados também deverá ser apresentada.

4. Considerações importantes

- Preze pelas boas práticas de programação. Modularize o seu código adequadamente. Crie arquivos .c e .h para cada módulo do seu sistema. Em especial, crie arquivos exclusivos para manipular as estruturas de dados dos tipos abstratos de dados que você estiver representando.

- Entrega:
Este trabalho deve ser feito em grupos de no máximo duas pessoas e deve ser enviado para pelo ambiente AVA.
O arquivo com o trabalho a ser enviado deve ser compactado. Deverá ser gerado um arquivo chamado Makefile, com as regras de compilação do programa, os arquivos com os códigos dos programas e o arquivo com a documentação em LaTeX ou pdf.

- Ao digitar:
make all
deve ser gerado os executáveis *gera* e *ordena*.

- O valor da nota será dividido da seguinte forma: 10% para a apresentação, 40% para o artigo e 50% para as implementações. Trabalhos identificados com plágio serão zerados. A cada dia de atraso, o valor do trabalho valerá menos 1 ponto.