

Atividade Prática 01

Algoritmos de Ordenação

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana
Curso de Engenharia de Computação
Disciplina de Estrutura de Dados 2 - EDCO4B
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Leia todas as instruções corretamente para poder desenvolver sua atividade/programa;
- Evite plágio (será verificado por meio de ferramentas automatizadas). Faça seu programa com os seus nomes de variáveis e lógica de solução. Plágios identificados anularão as atividades entregues de todos os envolvidos.
- Adicione comentários nos códigos explicando seu raciocínio e sua tomada de decisão. Porém, não exagere nos comentários, pois a própria estrutura do programa deve ser auto-explicativa.
- Salve sua atividade em um arquivo único, com todas as funções e procedimentos desenvolvidos. É esse **arquivo único** que deverá ser enviado ao professor.

1 Descrição da atividade

Implemente um programa que receba uma quantidade de números inteiros a ser ordenada e o método de geração desses números. O programa deve criar um vetor de N posições e preenchê-lo de acordo com o método de geração apresentado:

- 'c' - crescente sequencialmente de 1 a N;
- 'd' - decrescente sequencialmente de N a 1; ou
- 'r' - randômico, com os valores entre 0 e 32000;

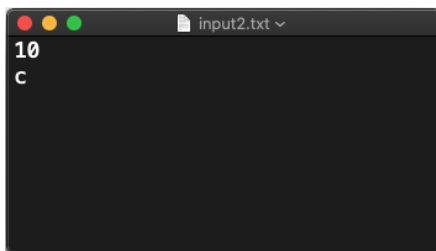
Gere um arquivo de saída com o vetor gerado e, para cada método indicado abaixo (nessa ordem), o nome do método, os elementos classificados em ordem crescente, o tempo gasto para obter a ordenação e o total de comparações feitas:

- insertionSort, selectionSort, bubbleSort, mergeSort, quickSort, e heapSort;

2 Entradas do programa

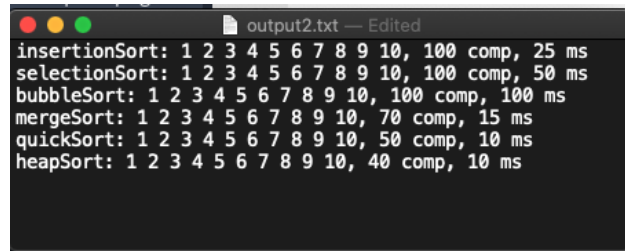
O programa receberá dois arquivos texto como parâmetros de entrada:

- **arquivo de entrada:** um arquivo texto contendo duas linhas. Na primeira linha é fornecido um número inteiro que corresponde ao tamanho do vetor que será manipulado. Na segunda linha consta o modo de geração do vetor (c, d, ou r);
- **arquivo de saída:** um arquivo texto onde deverá ser impressa a saída desejada. Uma linha para cada método de ordenação, com as seguintes informações:
 1. o nome do método de ordenação;
 2. os números ordenados;
 3. o número de comparações realizadas;
 4. e o tempo gasto para rodar o algoritmo (em milissegundos).



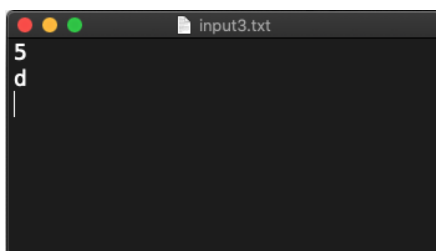
```
10
c
```

(a) Exemplo de arquivo de entrada para impressão dos registros em ordem crescente de código.



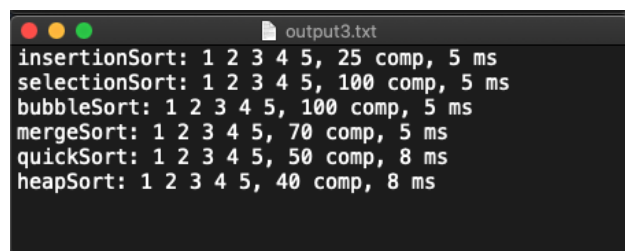
```
insertionSort: 1 2 3 4 5 6 7 8 9 10, 100 comp, 25 ms
selectionSort: 1 2 3 4 5 6 7 8 9 10, 100 comp, 50 ms
bubbleSort: 1 2 3 4 5 6 7 8 9 10, 100 comp, 100 ms
mergeSort: 1 2 3 4 5 6 7 8 9 10, 70 comp, 15 ms
quickSort: 1 2 3 4 5 6 7 8 9 10, 50 comp, 10 ms
heapSort: 1 2 3 4 5 6 7 8 9 10, 40 comp, 10 ms
```

(b) Exemplo de arquivo de saída com os registros impressos em ordem crescente de código.



```
5
d
```

(c) Exemplo de arquivo de entrada para impressão dos registros em ordem decrescente de código.



```
insertionSort: 1 2 3 4 5, 25 comp, 5 ms
selectionSort: 1 2 3 4 5, 100 comp, 5 ms
bubbleSort: 1 2 3 4 5, 100 comp, 5 ms
mergeSort: 1 2 3 4 5, 70 comp, 5 ms
quickSort: 1 2 3 4 5, 50 comp, 8 ms
heapSort: 1 2 3 4 5, 40 comp, 8 ms
```

(d) Exemplo de arquivo de saída com os registros impressos em ordem decrescente de código.

Figura 1: Valores de entrada e arquivos de saída gerado pelo programa. **Os valores de tempo e quantidade comparações apresentados nos arquivos de saída são fictícios.**

Exemplos de arquivos de entrada e correspondentes saídas são apresentados na Figura 1. **Dica:** Para rodar o programa por linha de comando, manipular os argumentos **argc** e **argv** da função **main**. Para executar o programa por linha de comando, deve-se obedecer o seguinte padrão:

[nome do programa] [arquivo de entrada] [arquivo de saída]

Exemplo de execução de um programa chamado `teste.c`:

```
./teste entrada.txt saida.txt
```

3 Orientações gerais

Obviamente, a entrada para cada método tem que ser o vetor original, portanto, deve ser passada uma **cópia** desse vetor para cada método. Para calcular o tempo gasto em segundos, use a ideia do programa a seguir (para cada método de ordenação):

```
#include <time.h>
clock_t start, end, diff;
start = clock();
/* chamar o algoritmo de ordenação */
end = clock();
/* a divisão por 1000 é para obter em milissegundos */
diff = (end - start)/(CLOCKS_PER_SEC/1000);
```

Além da funcionalidade desejada, implementar também o controle de erros, para lidar com exceções que possam ocorrer, como por exemplo:

- problemas nas aberturas dos arquivos de entrada e saída;
- arquivo de entrada vazio (sem informação);
- arquivo de entrada fora do padrão esperado, etc.

Opcionalmente, para acompanhamento do desenvolvimento, pode-se criar um repositório individual no `github`.

3.1 Critério de correção

A nota na atividade será contabilizada levando-se em consideração alguns critérios:

1. pontualidade na entrega;
2. não existir plágio;
3. completude da implementação (tudo foi feito);
4. o código compila e executa;
5. uso de `argc` e `argv` para controle dos arquivos de teste;
6. implementar o parser para entrada dos dados via arquivo texto;
7. implementação correta das estruturas e algoritmos necessários;
8. legibilidade do código (identação, comentários nos blocos mais críticos);
9. implementação dos controles de erros (arquivos de entrada inválidos, e erros no programa principal);
10. controle de memória: chamar o destrutor e desalocar a memória de tudo se usar estruturas dinâmicas, fechar os arquivos, etc;
11. executar corretamente os casos de teste.

Em cada um desses critérios, haverá uma nota intermediária valorada por meio de conceitos:

- **Sim** - se a implementação entregue cumprir o que se esperava daquele critério;
- **Parcial** - se satisfizer parcialmente o tópico;
- e **Não** se o critério não foi atendido.

Ao elaborar seu programa, crie um único arquivo fonte (.c) seguindo o padrão de nome especificado:

ED2-<ANO>-<SEMESTRE>-AT01-Ordenacao-<NOME>.c

Exemplo:

ED2-2022-1-AT01-Ordenacao-RafaelMantovani.c

A entrega da atividade será via Moodle: o link será disponibilizado na página da disciplina.

4 Links úteis

Arquivos em C:

- <https://www.inf.pucrs.br/~pinho/LaproI/Arquivos/Arquivos.htm>

- <https://www.geeksforgeeks.org/basics-file-handling-c/>
- <https://www.programiz.com/c-programming/c-file-input-output>

Arquivos em Python:

- <https://www.geeksforgeeks.org/reading-writing-text-files-python/>
- https://www.w3schools.com/python/python_file_open.asp
- <https://www.pythontutorial.net/python-basics/python-read-text-file/>

Argumentos de Linha de comando em C(argc e argv):

- https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
- <http://linguagemc.com.br/argumentos-em-linha-de-comando/>
- http://www.univasf.edu.br/~marcelo.linder/arquivos_pc/aulas/aula19.pdf
- http://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-31_Argumentos_linha_comando.html
- <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node145.html>

Argumentos de Linha de comando no Python:

- https://www.tutorialspoint.com/python3/python_command_line_arguments.htm
- <https://realpython.com/python-command-line-arguments/>
- <http://devfuria.com.br/python/sys-argv/>

Tempo de execução de scripts em Python:

- <https://realpython.com/python-timer/>
- <https://www.w3resource.com/python-exercises/python-basic-exercise-57.php>

Referências

- [1] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.
- [2] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.
- [3] Adam Drozdek. Estrutura De Dados E Algoritmos Em C++. Cengage, 2010.