BCC202 – Estrutura de Dados I

DECOM-UFOP

2s2015

Trabalho Prática (TP) 02

Data de Entrega: 19/02/2016 até às 23h55 (via *Moodle*)

1. Uma pilha pode ser implementada usando duas filas auxiliares, conforme descrito a seguir. A cada "*push*" para a pilha, transferir todos os elementos da *fila1* para a *fila2*. Em seguida, adicionar o elemento para a *fila1*. Então deve-se passar todos os elementos da *fila2* de volta para a *fila1*. O "*pop*" da pilha seria então retirar um elemento da *fila1*, que seria justamente o último a ser adicionado, caracterizando uma pilha.

Exemplo:

Push no valor 3, em uma pilha com os valores 1 e 2

Situação inicial

Fila1: 1 2

Fila2: <vazia>

Transferir tudo da *Fila1* para a *Fila2*:

Fila1: <vazia>

Fila2: 1 2

Acrescentar elemento 3 na *Fila1*:

Fila1: 3

Fila2: 1 2

Passar de volta os elementos da *Fila2* para a *Fila1*:

Fila1: 1 2 3

Fila2: <vazia>

Se for dado um *pop* em seguida, o elemento retirado será o 3, caracterizando uma pilha(*"last in – first out"*).

a) Implemente uma pilha usando duas filas. Isto é, os métodos "*push* " e "*pop*" da pilha devem ser implementados a partir da inserção e retirada dos elementos de duas filas auxiliares, que são manipuladas via operações "*enqueue*" e "*dequeue*". Codifique o método *main* no qual sua implementação deve ser testada para entradas pré-definidas.

b) Especifique o tempo de execução usando notação assintótica da operação *push()* e *pop()*. Justifique sua resposta.

2. A *Ordenação Mexida* define o seguinte problema: dado uma série de listas contendo palavras e números, o objetivo é ordenar essas listas de tal modo que todas as palavras fiquem em ordem alfabética e todos os números fiquem em ordem numérica. Além disso, se a *n-ésima* posição na lista for um número ele deve continuar a ser um número, e se este for uma palavra deve continuar a ser uma palavra. Ou seja, é uma ordenação entre palavras e números, mas estes respeitam posições de palavras e números na lista.

Especificação da Entrada

A entrada irá conter várias listas, uma lista por linha. Cada elemento da lista será separado por uma vírgula seguida de um espaço, e a lista vai ser encerrada por um ponto. A entrada será encerrada por uma linha contendo apenas um único ponto.

Especificação da Saída

Para cada lista na entrada, a saída é uma lista ordenada *mexida2*, separando cada elemento da lista, com uma vírgula seguida por um espaço, e finalizando a lista com um ponto.

Exemplo de Entrada

0.
banana, strawberry, OrAnGe.
Banana, StRaWbErRy, orange.
10, 8, 6, 4, 2, 0.
x, 30, -20, z, 1000, 1, Y.
50, 7, kitten, puppy, 2, orangutan, 52, -100, bird, worm, 7, beetle.
.

Exemplo de Saída

0.
banana, OrAnGe, strawberry.
Banana, orange, StRaWbErRy.
0, 2, 4, 6, 8, 10.
x, -20, 1, Y, 30, 1000, z.
-100, 2, beetle, bird, 7, kitten, 7, 50, orangutan, puppy, 52, worm.

a) Dado um conjunto de listas compostas por números e palavras, implemente a Ordenação Mexida. Codifique o método main para testar sua implementação para entradas pré-definidas.

b) Qual a complexidade de tempo do seu algoritmo de ordenação? Justifique sua resposta.

3. Considere os algoritmos de ordenação "Bubble, Select, Insertion, Merge, Quick e ShellSort" para as questões seguintes:

a) Gere um único arquivo de entrada com um milhão de números aleatórios e menores ou iguais a um milhão. Ordene tais números utilizando cada um dos algoritmos de ordenação mencionados. Para cada um dos algoritmos, indique o número de comparações e movimentações realizadas e também o tempo de execução de cada um deles (e.g., em segundos) para ordenar os números gerados aleatoriamente.

b) Gere um único arquivo de entrada com um milhão de números (de 1,2,3, 4 até 1.000.000, ordem crescente). Ordene tais números utilizando cada um dos algoritmos de ordenação mencionados. Para cada um dos algoritmos, indique o número de comparações e movimentações realizadas e também tempo de execução de cada um deles (e.g., em segundos) para ordenar os números de entrada.

c) Gere um único arquivo de entrada com um milhão de números (1.000.000, 999.999, até 1). Ordene tais números utilizando cada um dos algoritmos de ordenação mencionados. Para cada um dos algoritmos, indique o número de comparações e movimentações realizadas e também tempo de execução (e.g., em segundos) de cada um deles para ordenar os números gerados aleatoriamente.

d) Comente brevemente sobre o desempenho dos diferentes algoritmos no casos anteriores.

Procedimento de Entrega

Você deve entregar os exercícios no *Moodle* até o dia **22/05/15** às **23:55**. Não serão aceitas soluções entregues após o horário estipulado.

Para cada questão:

1- Crie uma pasta para a questão: PrimeiroNome-UltimoNome-Questao (exemplo: amanda-nascimento-1).

2- Crie os arquivos de código fonte necessarios para a solução da questão (arquivos .c e .h)

3- Compile na linha de comando usando gcc *.c -o prog.exe -lm.

4- Apague os arquivos gerados na compilação (mantenha apenas os arquivos .c e .h).

Compacte todas as pastas criadas anteriormente num único arquivo ("*.zip*"), que deve ser entregue via Moodle. *O arquivo compactado deve também conter um arquivo ".txt" com seu nome e número de matrícula.*