BCC202 – Estruturas de Dados I (2013-02)
Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Reinaldo Fortes** (`www.decom.ufop.br/reinaldo`)
Estagiária docente: **Josiane Rezende**
Monitores: **Bruno H. M. dos Santos**

Trabalho Prático 02 (TP02)

- **Data de entrega: 15/12/2013 até 23:55. O que vale é o horário do Moodle, e não do *seu*, ou do *meu*, relógio!!!**
- **Decréscimo por atraso de até:** 12h = **30%**, 24h = **40%**; 36h = **60%**; 48h = **70%**; **Acima de 48h = 100%**.
- O padrão de entrada e saída deve ser respeitado **exatamente** como determinado no enunciado.
- **Parte da correção é automática, não respeitar as instruções enunciadas pode acarretar em perda de pontos**.
- ***Bom trabalho!***

1 Objetivos

Este trabalho prático tem como objetivo principal fundamentar os conceitos das estruturas Lista, Pilha e Fila e de algoritmos de ordenação.

2 Descrição do Problema

Tomando como base as práticas 04 e 05, implemente as seguintes funcionalidades adicionais para a **versão implementada por array**:

- Ordenar a lista de contatos de uma agenda usando o campo **nome** como chave.
- Ordenar a lista de contatos de uma agenda usando o campo **telefone** como chave.
- Ordenar a lista de agendas usando o campo **telefone** como chave.

Deverão ser implementados todos os métodos de ordenação vistos até a data de entrega do TP: *BubbleSort*, *SelectionSort*, *InsertionSort*, *MergeSort* e *QuickSort*.
Sendo assim, deverão ser acrescentadas à TAD **TListaContato** as seguintes operações:

7. Ordena contatos pelo **nome** usando *BubbleSort*.
8. Ordena contatos pelo **nome** usando *SelectionSort*.
9. Ordena contatos pelo **nome** usando *InsertionSort*.
10. Ordena contatos pelo **nome** usando *MergeSort*.
11. Ordena contatos pelo **nome** usando *QuickSort*.
12. Ordena contatos pelo **telefone** usando *BubbleSort*.
13. Ordena contatos pelo **telefone** usando *SelectionSort*.
14. Ordena contatos pelo **telefone** usando *InsertionSort*.
15. Ordena contatos pelo **telefone** usando *MergeSort*.
16. Ordena contatos pelo **telefone** usando *QuickSort*.

De forma semelhante, deverão ser acrescentadas à TAD **TListaAgenda** as seguintes operações:

6. Ordena contatos pelo **telefone** usando *BubbleSort*.
7. Ordena contatos pelo **telefone** usando *SelectionSort*.
8. Ordena contatos pelo **telefone** usando *InsertionSort*.
9. Ordena contatos pelo **telefone** usando *MergeSort*.
10. Ordena contatos pelo **telefone** usando *QuickSort*.

2.1 Exemplo de Entrada e Saída

A entrada e saída seguirão o mesmo padrão das práticas 4 e 5, acrescentando as novas operações definidas. As operações de ordenação não deverão imprimir nada como saída, quando desejar observar o resultado da ordenação deverá ser realizada a operação de impressão dos elementos da lista.

2.2 Análise comparativa dos métodos de ordenação

Adicionalmente, deverá ser realizada uma análise comparativa dos tempos de execução dos métodos de ordenação. Você deverá realizar as operações de ordenação para diferentes casos de teste, variando o número de elementos da lista, e a disposição inicial destes elementos (ordenado, invertido e aleatório).

Exemplo de código para obter o tempo de execução:

```
#include <time.h>
:
clock_t tFim, tIni = clock(); // instrucao exatamente anterior a chamada
Ordena(); // chamada da funcao de ordenacao
tFim = clock(); // instrucao exatamente posterior a chamada
TempoMiliSegundos = (double)(tFim - tIni) * 1000 / CLOCKS_PER_SEC;
:
```

3 Imposições e comentários gerais

- É fundamental que sejam respeitados os conceitos de: TAD, Listas, Pilhas, Filas e Agoritmos de Ordenação.
- Seu programa não pode ter "*memory leaks*", ou seja, toda memória alocada deve ser liberada pelo seu código.
- Clareza, identação e comentários no código também vão valer pontos. Por isso, escolha cuidadosamente o nome das variáveis e torne o código o mais legível possível.
- Trabalhos copiados (e FONTE) terão nota zero, além de os alunos envolvidos no plágio perderem toda a nota atribuída a participação e pontos extras, entre outros (...). **Isto vale para qualquer entregável** (seção 4).
- Caso seja necessário, alunos poderão ser convocados para entrevista.

4 Entregáveis

Deverão ser entregues para avaliação do trabalho:

- **Código fonte**: código fonte do programa em `C`.
- **Arquivos de entrada e saída**: arquivos de entrada e saída utilizados para testar seu programa. Não será considerada entrada igual ao exemplo dado neste enunciado, procure elaborar um conjunto de testes abrangente, envolvendo várias agendas e muitas tarefas.
- **Documentação**: a documentação deve ser entregue em um único arquivo `PDF`, e conter:
    1. **Implementação**: descrição da implementação do programa. Não faça "print screens" de telas e não inclua o código fonte. Ao contrário, procure resumir ao máximo a documentação, fazendo referência ao que julgar mais relevante. É importante, no entanto, que seja descrito o funcionamento das principais funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.

2. **Análise comparativa dos métodos de ordenação**: descreva e discuta os resultados da análise comparativa dos métodos de ordenação.
3. **Impressões gerais**: descreva o seu processo de implementação deste trabalho. Aponte coisas que gostou bem como aquelas que o desagradou. Avalie o que o motivou, conhecimentos que adquiriu, entre outros.
4. **Conclusão**: conclusões e comentários gerais sobre o trabalho.

4.1 Como fazer a entrega

Verifique se seu programa compila e executa na linha de comando (utilizando o compilador `GCC`) antes de efetuar a entrega. Quando o resultado for correto, entregue via Moodle dois arquivos `.ZIP` e um `.PDF` com seu nome e sobrenome. Exemplo:

- **PrimeiroNome-UltimoNome.zip**: este arquivo `.ZIP` deve conter apenas os arquivos `.c` e `.h`, utilizados na implementação, e os arquivos `.TXT` de entrada e saída, utilizados para testar seu programa.
- **PrimeiroNome-UltimoNome.pdf**: este arquivo `.PDF` deve conter a documentação do trabalho.

Serão criadas três tarefas no moodle:

1. Entrega dos arquivos de código fonte da implementação das listas usando ponteiros.
2. Entrega dos arquivos de código fonte da implementação das listas usando array (compreendendo as operações de ordenação).
3. Documentação (arquivo `.PDF`).