UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS PONTA GROSSA
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
Disciplina: AS32A – Estrutura de dados

TRABALHO FINAL

1. Introdução

A representação adequada dos dados, tendo em vista as funcionalidades a serem atendidas, constitui uma etapa fundamental para a obtenção de programas eficientes e confiáveis. Desse modo, não basta apenas ter conhecimento em linguagens de programação, é necessário utilizá-las de maneira eficiente. Este trabalho tem como objetivo avaliar a criticidade e capacidade de tomada de decisão dos alunos em relação à melhor abordagem para resolução do problema proposto, o qual exigirá a aplicação dos conceitos ministrados ao longo da disciplina.

2. Descrição

O trabalho final da disciplina é o desenvolvimento de uma aplicação para processar um arquivo de texto e salvar seu conteúdo numa estrutura de dados para posterior processamento. Tal processamento envolverá as operações de busca, inserção, remoção, ordenação e impressão dos valores existentes na estrutura. O que você deve fazer é escolher a estrutura de dados mais adequada (i) para armazenar os valores do arquivo e para executar as operações definidas, bem como (ii) o método mais eficiente para realizar as operações requeridas. Entretanto, algumas exigências devem ser atendidas:

- A estrutura de armazenamento dos valores **não pode ser estática**, portanto sua manipulação deve ser feita com alocação dinâmica de memória.
- O código-fonte deve ser modularizado.
- A operação padrão de busca será a busca sequencial, porém se for informado um valor diferente de zero para o parâmetro `index` da função que desempenha a busca, a mesma deverá ser realizada por outro método de busca, à sua escolha.

Exemplo:

```
void buscar(int index) {
   if (index == 0) busca_sequencial(...);
   else busca_xyz(...);
}
```

- A operação padrão de ordenação será o *BubbleSort*, porém se for informado um valor diferente de zero para o parâmetro `index` da função que realiza tal ordenação é necessário que outro método de ordenação, à sua escolha, seja utilizado.

  Exemplo:

```
void ordenar(int index) {
   if (index == 0) bubble_sort(...);
   else ordenacao_xyz(...);
}
```

- Um menu deve ser criado para permitir a seleção da operação que se deseja executar, o qual deve conter as seguintes opções:
  1. **Carregar arquivo:** Vai carregar na memória o conteúdo do arquivo.
  2. **Inserir novo valor:** Insere um novo valor na estrutura de armazenamento.
  3. **Remover valor:** Procura por um valor que será removido, caso exista.
  4. **Buscar valor:** Procura pelo valor na estrutura de armazenamento e imprime uma mensagem se o encontrou ou não. O método de busca deverá ser especificado.
  5. **Ordenar valores:** Ordena os valores da estrutura. O método de ordenação deverá ser especificado.
  6. **Imprimir valores:** Imprime todos os valores da estrutura.
  7. **Gerar relatório:** Gera um relatório da execução das operações do programa (mais detalhes abaixo).
  8. **Sair do programa**: Ao sair, a estrutura deve ser liberada da memória e o programa encerrado.
- A opção *Gerar relatório* realizará um conjunto de operações específicas, as quais terão o tempo de execução registrado e devem ser feitas na ordem apresentada abaixo. Ao final, você deve gerar um arquivo de texto (*.txt*) informando o tempo de execução de cada uma das operações e o tempo total de execução de todas as operações.
  1. Ler o arquivo e inserir seus valores na estrutura.
  2. Buscar por um valor existente na estrutura (`index != 0`)
  3. Buscar por um valor não existente na estrutura (`index != 0`)
  4. Ordenar **e imprimir no console** os valores da estrutura (`index != 0`)

  Para obter o tempo de execução em milissegundos de certa porção de código em linguagem C é necessário o seguinte código-fonte:

```c
#include <stdio.h>
#include <time.h>

int main() {

    double tempoGasto;
    clock_t tempo[2];

    /* obtém o momento do início da execução (ms) */
    tempo[0] = clock();

    /* AQUI VAI O CÓDIGO QUE TERÁ O TEMPO MEDIDO */

    /* obtém o momento de fim da execução (ms) */
    tempo[1] = clock();

    /* Obtém o tempo de execução em ms */
    tempoGasto = (tempo[1] - tempo[0]) * 1000.0 / CLOCKS_PER_SEC;

    printf("Tempo gasto: %g ms.", tempoGasto);
```

A equipe que obter a melhor média de tempo nas operações da opção *Gerar relatório* vai receber um bônus de 10% do valor do trabalho na nota atribuída a esse.

3. Informações adicionais

- O trabalho deverá ser postado no Moodle na data solicitada. Se feito em equipe, apenas um membro deve postar o arquivo e informar o nome e o RA de todos os integrantes.
- A apresentação do trabalho será feita na data especificada e vai consistir de uma pequena apresentação e explicação de partes do código-fonte, se solicitado.
- A execução dos programas para obtenção do tempo de execução será feita na mesma máquina a fim de tornar justa a comparação.
- Data prevista para início das apresentações: **06/12/2019**.
- Valor: 1,5 pontos