Disciplina: CI067 - Oficina de Computação

# Trabalho III

Um supermercado possui duas filiais, cujos os dados de produto são armazenados em dois arquivos com nomes diferentes fornecidos na linha de comando. Os arquivos estão ordenados pelo nome do produto (string com no máximo 25 caracteres que pode conter brancos). Os arquivos contêm dados de 4 setores: 1: derivados de leite; 2: hortifruti; 3: higiene; 4: limpeza. Você pode supor que os arquivos estão corretamente ordenados e que não há nomes de produtos ou códigos de setores inválidos. Cada produto pode ser descrito da seguinte maneira.

```
#include <stdio.h>
typedef struct  {        // descreve o produto
        char nome[25];
        int codSetor;
        double preco;
        int qtidade;
} tipoProd;
```

Considere a seguinte estrutura para uma lista encadeada.

```
typedef struct tipoNo {     // descreve cada no da lista
  tipoProd prod;
  struct tipoNo * prox;
}tipoNo;

typedef struct tipoNo * apontador;
typedef struct {
  apontador inicio;            // define a estrutura da lista
  apontador fim;
} tipoLista;
```

E as seguintes funções:

```
#include <lista.h>
inicia(tipoLista * lista)
{
  /* no inicio ambos apontadores sao nulos */
  lista->inicio = lista->fim = NULL;
}

int vazia (tipoLista lista)
{
  /* lista vazia quando inicio = fim = nulo */
  return(lista.inicio == NULL);
}
```

**1. Implemente as seguintes funções associadas ao T.A.D**.

```
/* insere produto x no fim da lista */
int insereFim(tipoProd x, tipolista * lista);

/* insere x na lista de maneira que lista fique ordenada
 pelo nome de produto */
int insereOrd(tipoProd x, tipolista * lista);

/* imprime lista em arquivo de saida */
imprime(tipolista lista, FILE * arqsaida);

/* retorna o numero de elementos da lista */
int tamanhoLista (tipoLista lista);
```

**2. Gere uma biblioteca de funções compartilhada liblista.so**. Crie um diretório de includes e de bibliotecas e utilize um makefile para gerar a biblioteca.

**3. Implemente um programa que utiliza os arquivos lista.h, e a biblioteca liblista.so** e outros que você achar necessário, para montar uma lista encadeada para os produtos do supermercado. A lista é obtida intercalando os arquivos fornecidos e mantendo a ordem alfabética. Lembre-se que deve haver uma única entrada na lista para um produto de mesmo nome, mas a quantidade em estoque deverá considerar ambas filiais. Se o

preço do produto de mesmo nome for diferente em cada filial, o maior preo deverá ser considerado.

Ao final, os dados da lista devem ser armazenados em um terceiro arquivo, cujo o nome é também fornecido na linha de comando. O programa deverá mostrar na tela o nuúmero de diferentes produtos do supermercado e os produtos com maior e menor preço se nada for especificado para o primeiro argumento. Caso seja fornecido um valor para o primeiro argumento, o programa exibe na tela o número de produtos diferentes de um dado setor bem como os produtos com maior e menor preço do setor especificado.

Veja que o número de elementos dos arquivos é desconhecido. Não esqueça de considerar situações de erro, ou de mau uso do programa na linha de comando.

Modularize seu código, faça-o legível e organizado, fácil de manter, utilizando boas práticas de programação. Utilize as funções implementadas. Mas você pode usar outras que desejar.

Linha de comando:
> programaFilial [n] filial1 filial2 relatorio

n número do setor desejado. Este parâmetro é opcional, se nada for colocado serão considerados todos os setores do supermercado para calcular o número de produtos diferentes, e os produtos com maior e menor preço.

filial1 - nome do arquivo com os dados da filial 1. Ordenado por nome de produto

filial2 - nome do arquivo com os dados da filial 2. Ordenado por nome de produto

relatorio - nome do arquivo no qual serão colocados os dados dos produtos, no qual as quantidades devem considerar ambas filiais.

4. **Você deve gerar um makefile para** *programaFilial*.

5. **Entrega**:

Você deve criar um arquivo com todos os arquivos, incluindo os de teste com extensão .tar. Utilizar o programa entrega, assim como os exercícios en-

tregues em sala, com o comando abaixo. > /home/html/inf/silvia/entrega/bin/entrega oficinac 22 arquivo.tar

Os arquivos devem ser entregues até às 23:59 de domingo, dia 25/11. Trabalhos entregues depois do dia 25 terão 10 pontos descontados por dia de atraso. A data limite de entrega é dia 28/11.

O programa entregue será apresentado pelo próprio aluno em sala de aula no dia 28/11. Durante a apresentação serão feitas perguntas sobre a implementação.

**6. Correção**:

O trabalho vale 100 pontos. A avaliação seguirá os seguintes critérios e valores.

- funcionalidade (70): relacionada à implementação correta das funções e ao funcionamento do programa.
- a estruturção e organizção do código (5) e a legibilidade (5) conforme material passado em aula. Utilize funções, declare suas funções, use comentários adequados, arquivos .h e .c
- arquivo makefile é obrigatório (5) pelo menos com uma variável.
- biblioteca de funções + makefile para criar a biblioteca (10).
- defesa do trabalho (5): respostas corretas a perguntas feitas no dia da apresentção.

Declare nos .h somente macros, diretivas, tipos e protótipos de funções. Não inclua código C nestes arquivos. Não utilize variáveis globais.

Caso o programa enviado não compile, a nota será 0. Teste muito bem o seu programa e certifique-se que ele est funcionando corretamente. Preferivelmente guarde versões dele, pois se acontecer algum imprevisto, uma versão anterior poderá ser utilizada. Retire todos os warnings.

Caso o programa execute, porém falhe durante a execução por um erro de programação, então o programa será avaliado individualmente e será dada uma nota relativa aos requisitos atendidos. Durante a apresentação do trabalho serão feitas perguntas sobre como o aluno implementou o programa.

Quaisquer funcionalidades extras que não tenham sido pedidas no enunciado são opcionais e não contribuem para a nota.