

Universidade Federal do Rio de Janeiro - UFRJ
Trabalho de Computação 1

SISTEMA DE COTAÇÃO DE PREÇOS
USANDO ARQUIVOS

Paula Macedo da Cruz DRE 113049909

Rio de Janeiro, 19 de julho de 2016.

1.0. Descrição do problema

Implementar um sistema de cotação de preços que tem como entrada dois arquivos textos. Um arquivo texto apresenta apenas o código dos produtos e sua descrição, já o outro, apresenta o código e o seu preço ou valor unitário. O programa utiliza as informações desses dois arquivos e as armazena num vetor dinâmico do tipo PRODUTO. É nesse vetor que as buscas são realizadas. Vale ressaltar que os códigos apresentam-se em ordem crescente com a finalidade de facilitar a busca. Ao final, o programa retorna ao usuário um arquivo csv contendo o conteúdo de todas as buscas na ordem que as mesmas foram realizadas.

2.0. Divisão do programa

Com a finalidade de facilitar a compreensão do código, o programa foi dividido em 3 arquivos : main.c , func.c e func.h.

O “main.c” apresenta a função main do programa, além da chamada das funções declaradas no arquivo “func.h” e implementadas no arquivo “func.c”.

No arquivo “func.h” também está definido o tipo PRODUTO que é utilizado ao longo de todo o programa.

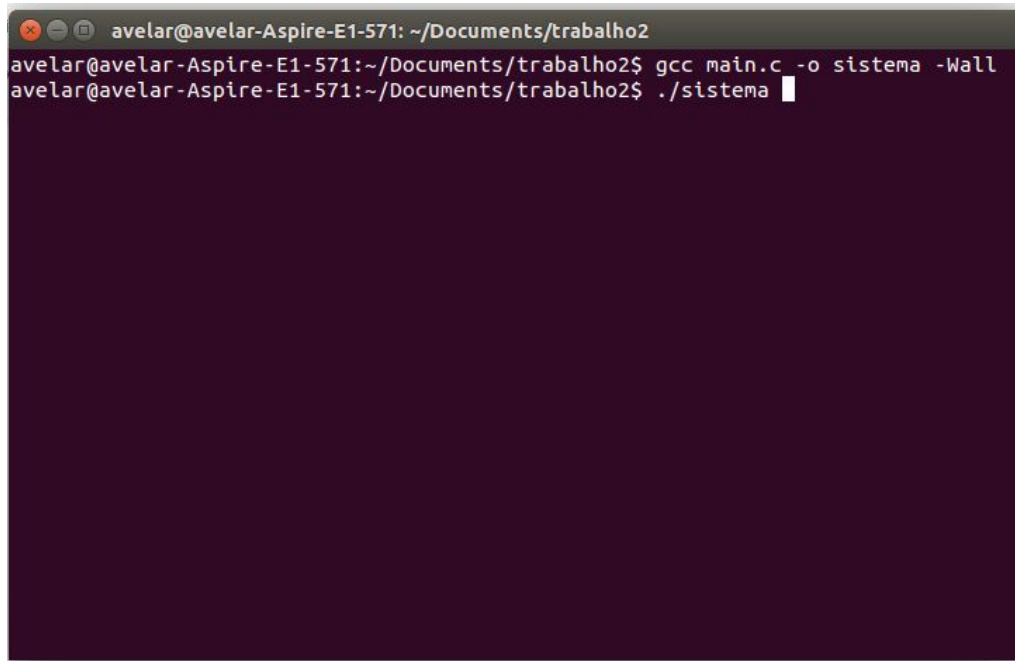
3.0. Solução do problema

- O programa solicita ao usuário que o mesmo informe o nome dos arquivos de entrada;
- É lido o nome dos arquivos e os mesmos são abertos. Caso a abertura dos arquivos não seja possível é retornado um número inteiro e o programa é encerrado;
- É chamada a função que descobre quantas linhas possui o arquivo;

- Com o número de linhas do arquivo, é chamada a função que passa o conteúdo dos arquivos para um vetor dinâmico;
- Antes disso, é alocada memória para esse vetor. Caso o mesmo não seja possível o programa é encerrado;
- O programa pergunta ao usuário se ele deseja realizar uma busca. Caso a resposta seja sim, o usuário digitará 1 e o programa lerá esse número e compreenderá que é para dar prosseguimento com o programa. Caso o usuário digite 0 ou qualquer outro número será o mesmo que não e o programa se encerrará;
- É solicitado novamente ao usuário que ele forneça um nome de arquivo, este será o nome do arquivo de saída no formato csv gerado pelo programa. Caso o programa não consiga criar esse arquivo, o mesmo será encerrado;
- O programa entra num loop, que só é possível de sair se o mesmo não desejar mais realizar buscas;
- Dentro desse while é solicitado ao usuário que o mesmo forneça o código e a quantidade do produto;
- Uma função de busca é chamada e caso seu retorno seja negativo, isso representa que tal código fornecido pelo usuário não existe. Nesse caso, o programa pede novamente o código e a quantidade;
- Caso o código seja encontrado, é calculado o preço total e todas as variáveis pertencentes a ele são escritas no arquivo de saída;
- É perguntado ao usuário se ele deseja realizar outra busca, se sim, o loop prossegue. Caso contrário, a busca é encerrada, o arquivo csv é gerado por completo e o programa se encerra.

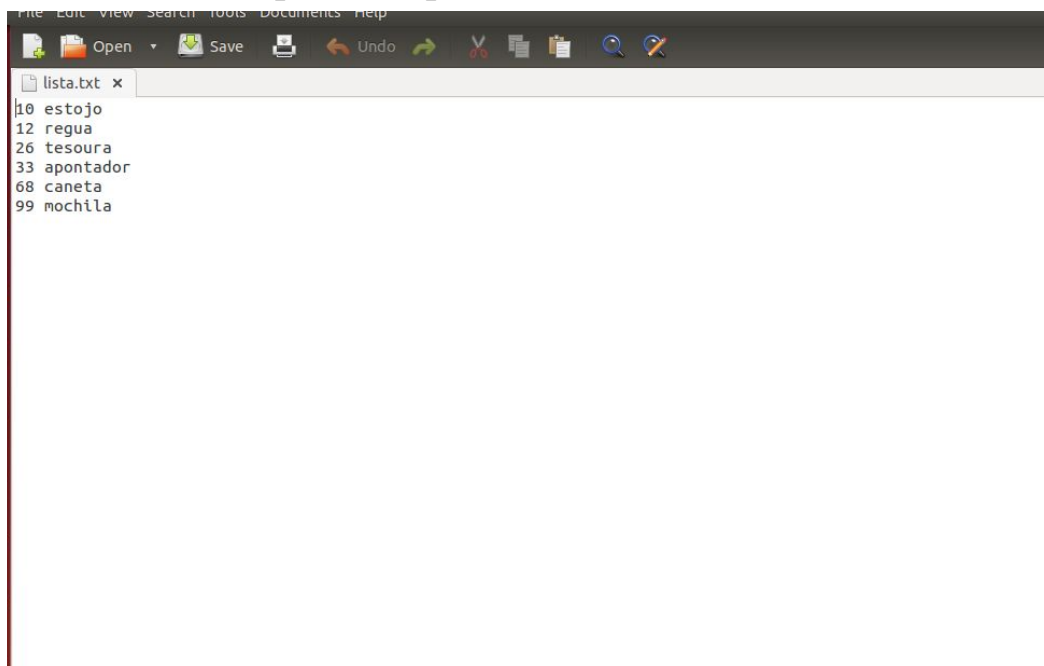
4.0. Conjunto de casos testes

Compilar o programa da seguinte maneira :



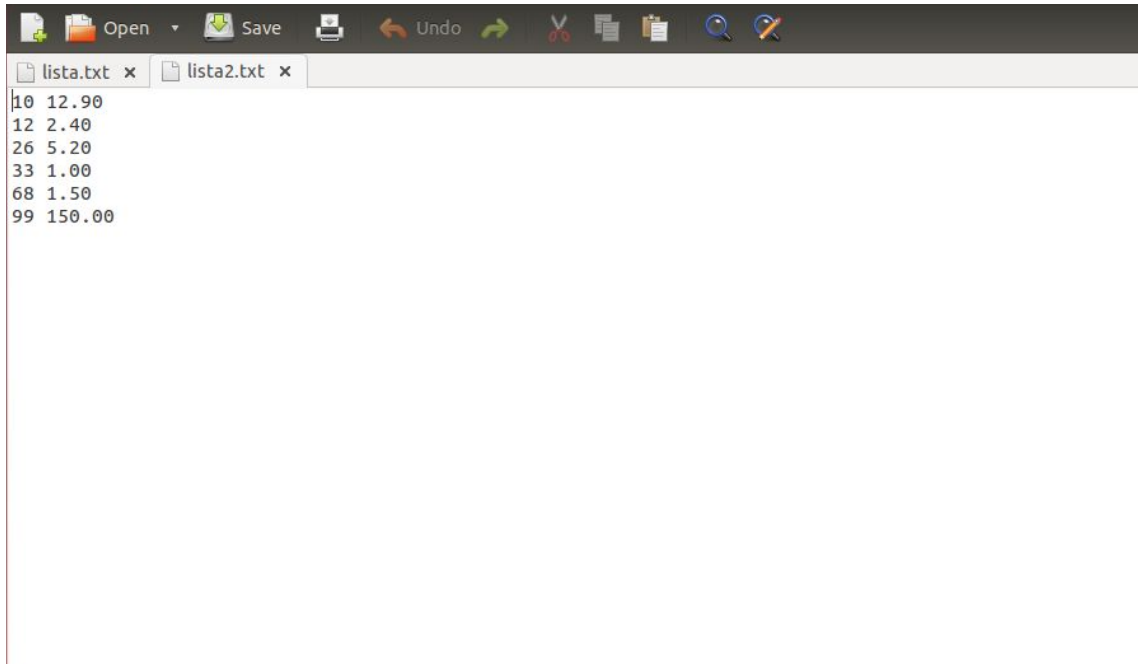
```
avelar@avelar-Aspire-E1-571: ~/Documents/trabalho2
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ gcc main.c -o sistema -Wall
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ ./sistema
```

Exemplo de arquivo utilizado como lista.txt



```
File Edit View Search Tools Documents Help
lista.txt x
10 estojo
12 regua
26 tesoura
33 apontador
68 caneta
99 mochila
```

Exemplo de arquivo utilizado como lista2.txt

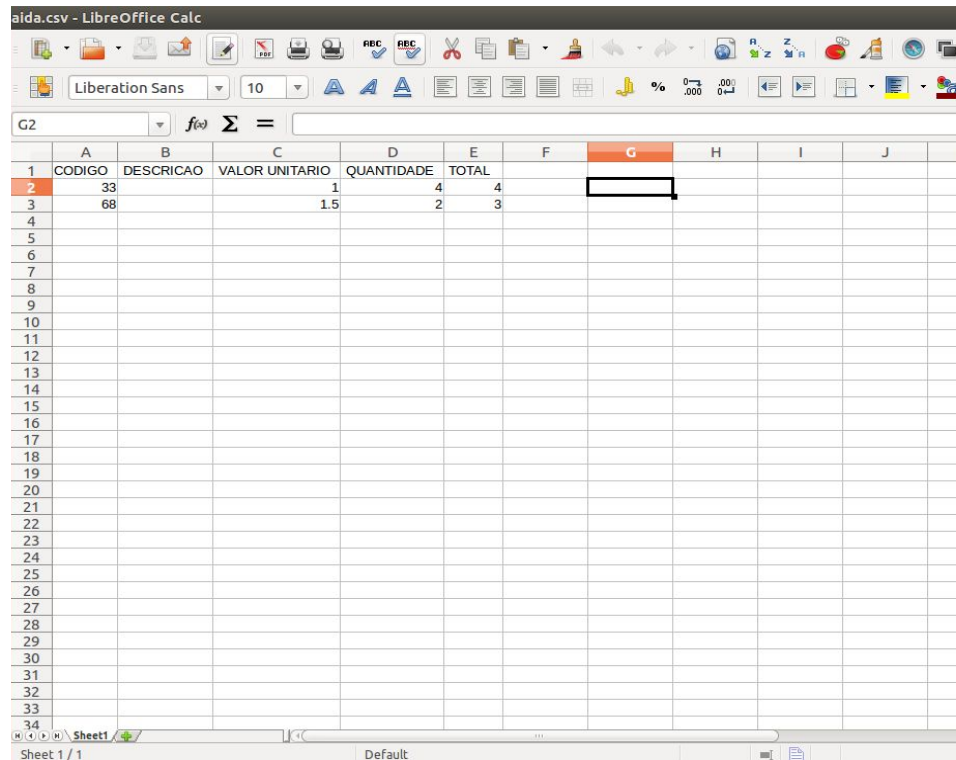


Caso 1 (Usuário digitando tudo certo e realizando duas buscas)

Entrada

```
avelar@avelar-Aspire-E1-571: ~/Documents/trabalho2
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ gcc main.c -o sistema -Wall
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ ./sistema
Por favor, entre com o nome do arquivo 1 (codigo e descrição). Lembrando : exemplo.txt
lista.txt
Por favor, entre com o nome do arquivo 2 (codigo e preço unitário). Lembrando : exemplo.txt
lista2.txt
Vamos começar a nossa busca?
1 . PARA SIM
0 . PARA NÃO
1
Digite o nome do arquivo de saída. Lembrando : exemplo.csv
arquivosaida.csv
Entre com o código do produto e a quantidade
33 4
Deseja continuar com a busca?
1 . Para continuar
0 . Para encerrar busca e gerar csv
1
Entre com o código do produto e a quantidade
68 2
Deseja continuar com a busca?
1 . Para continuar
0 . Para encerrar busca e gerar csv
0
Seu arquivo de saída já está salvo no seu diretório.
Fim do programa!
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$
```

Saída



aida.csv - LibreOffice Calc

	A	B	C	D	E	F	G	H	I	J
	CODIGO	DESCRICAO	VALOR UNITARIO	QUANTIDADE	TOTAL					
1										
2	33		1	4	4					
3	68		1.5	2	3					
4										
5										
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										
21										
22										
23										
24										
25										
26										
27										
28										
29										
30										
31										
32										
33										
34										

Sheet 1 / 1

Caso 2 (Usuário fornece os dois arquivos de entrada, mas desiste da busca)

Entrada

```
avelar@avelar-Aspire-E1-571: ~/Documents/trabalho2
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ gcc main.c -o sistema -Wall
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ ./sistema
Por favor, entre com o nome do arquivo 1 (codigo e descrição). Lembrando : exemplo.txt
lista.txt
Por favor, entre com o nome do arquivo 2 (codigo e preço unitário). Lembrando : exemplo.txt
lista2.txt
Vamos começar a nossa busca?
1 . PARA SIM
0 . PARA NÃO
0
Fim do programa!
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$
```

Saída

Nenhum arquivo é gerado.

Caso 3 (Usuário fornece um código para busca errado)

Entrada

```

avelar@avelar-Aspire-E1-571: ~/Documents/trabalho2
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ gcc main.c -o sistema -Wall
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ ./sistema
Por favor, entre com o nome do arquivo 1 (codigo e descrição). Lembrando : exemplo.txt
lo.txt
lista.txt
Por favor, entre com o nome do arquivo 2 (codigo e preço unitário). Lembrando : exemplo.txt
lista.txt
VAMOS COMEÇAR A NOSSA BUSCA?
1 . PARA SIM
0 . PARA NÃO
1
Digite o nome do arquivo de saída. Lembrando : exemplo.csv
arquivosaida.csv
Entre com o código do produto e a quantidade
33 4
Deseja continuar com a busca?
1 . Para continuar
0 . Para encerrar busca e gerar csv
1
Entre com o código do produto e a quantidade
67 3
Código não encontrado!
Entre com o código do produto e a quantidade
68 2
Deseja continuar com a busca?
1 . Para continuar
0 . Para encerrar busca e gerar csv
0
Seu arquivo de saída já está salvo no seu diretório.
Fim do programa!
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$

```

Saída

[illegible]

Caso 4 (Usuário fornece o primeiro arquivo de entrada inválido)

Entrada

```
avelar@avelar-Aspire-E1-571: ~/Documents/trabalho2
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ gcc main.c -o sistema -Wall
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ ./sistema
Por favor, entre com o nome do arquivo 1 (codigo e descrição). Lembrando : exemplo.txt
lista1.txt
Erro! Não foi possível realizar a leitura do arquivo.
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$
```

Saída

Nenhum arquivo é gerado.

Caso 5 (Usuário fornece o segundo arquivo de entrada inválido)

Entrada

```
avelar@avelar-Aspire-E1-571: ~/Documents/trabalho2
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ gcc main.c -o sistema -Wall
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$ ./sistema
Por favor, entre com o nome do arquivo 1 (codigo e descrição). Lembrando : exemplo.txt
lista.txt
Por favor, entre com o nome do arquivo 2 (codigo e preço unitário). Lembrando : exemplo.txt
lista3.txt
Erro! Não foi possível realizar a leitura do arquivo.
avelar@avelar-Aspire-E1-571:~/Documents/trabalho2$
```


Saída

Nenhum arquivo é gerado.

5.0. Dificuldades

- Modularizar o programa;
- Decidir qual seria o retorno e os parâmetros de algumas funções;
- Manipular ponteiros e passá-los por referência;
- Fazer com que a descrição do produto seja impressa no arquivo de saída csv;
- Entender o porquê do surgimento de erros do tipo de “falha de segmentação”;
- Fazer com que a função `fscanf` lê-se a string fornecida pelo arquivo e armazená-la no vetor dinâmico.

6.0. Conclusão

Fazer programas que façam uso de arquivos textos ou que gerem arquivos textos não é algo muito complexo. O mais difícil e trabalhoso é dividir o programa em vários arquivos, trabalhar com modularidade e ponteiros.