

**Faculdade de Computação - FACOM**

**Bacharelado em Sistemas de Informação**

***FACOM32201 - Algoritmos e Programação II***

**Prof. Thiago Pirola Ribeiro**

# ARQUIVOS

## Parte 3

# Organização de Arquivos

- Informações em arquivos são, em geral, organizadas logicamente em **campos** e **registros**

# Organização de Arquivos

- Informações em arquivos são, em geral, organizadas logicamente em **campos** e **registros**
- Entretanto, campos e registros são conceitos lógicos, que não necessariamente correspondem a uma organização física

# Organização de Arquivos

- Informações em arquivos são, em geral, organizadas logicamente em **campos** e **registros**
- Entretanto, campos e registros são conceitos lógicos, que não necessariamente correspondem a uma organização física
- Dependendo de como a informação é mantida no arquivo, campos lógicos sequer podem ser recuperados...

# Sequência de bytes (*stream*)

## Exemplo:

- Suponha que desejamos armazenar em um arquivo os nomes e endereços de várias pessoas
- Suponha que decidimos representar os dados como uma sequência de bytes (sem delimitadores, contadores, etc.)

```
AmesJohn123  MapleStillwater0K74075MasonAlan90  EastgateAda0K74820
```

```
JoaoNavesDeAvila2000NicomedesAlvesDosSantos23RondonPacheco765  
XVDeNovembro19DeJulho34
```

## Sequência de bytes (*stream*)

- Uma vez escritas as informações, **não existe como recuperar porções individuais** (nome ou endereço)

## Sequência de bytes (*stream*)

- Uma vez escritas as informações, **não existe como recuperar porções individuais** (nome ou endereço)
- Desta forma, perde-se a integridade das unidades fundamentais de organização dos dados



## Sequência de bytes (*stream*)

- Uma vez escritas as informações, **não existe como recuperar porções individuais** (nome ou endereço)
- Desta forma, perde-se a integridade das unidades fundamentais de organização dos dados
- Os dados são agregados de caracteres com significado próprio

## Sequência de bytes (*stream*)

- Uma vez escritas as informações, **não existe como recuperar porções individuais** (nome ou endereço)
- Desta forma, perde-se a integridade das unidades fundamentais de organização dos dados
- Os dados são agregados de caracteres com significado próprio
- Tais agregados são chamados **campos** (*fields*)

# Organização em campos

- **Campo** é a menor unidade lógica de informação em um arquivo

# Organização em campos

- **Campo** é a menor unidade lógica de informação em um arquivo
- Uma noção lógica (ferramenta conceitual), não corresponde necessariamente a um conceito físico

# Organização em campos

- **Campo** é a menor unidade lógica de informação em um arquivo
- Uma noção lógica (ferramenta conceitual), não corresponde necessariamente a um conceito físico
- Existem várias maneiras de organizar um arquivo mantendo a identidade dos campos

# Organização em campos

- **Campo** é a menor unidade lógica de informação em um arquivo
- Uma noção lógica (ferramenta conceitual), não corresponde necessariamente a um conceito físico
- Existem várias maneiras de organizar um arquivo mantendo a identidade dos campos
- A organização anterior (sequência de bytes - stream) não proporciona isso...

# Métodos para organização em campos

- 1 Forçar todos os campos para um tamanho (comprimento) fixo
- 2 Começar cada campo com um indicador de tamanho (comprimento)
- 3 Colocar delimitadores entre campos
- 4 Usar expressões (tags) `keyword = value` para identificar cada campo e seu conteúdo

## Campos com tamanho fixo

- Cada campo ocupa no arquivo um tamanho fixo, pré-determinado
- O fato do tamanho ser conhecido garante que é possível recuperar cada campo

Maria	Rua 1	123	Ribeirão Preto
José	Rua J	32	Monte Carmelo
Ana	Rua 22	73	Uberlândia



## Campos com tamanho fixo

```
char nome[20];  
char sobrenome[20];  
char cidade[15];  
char estado[2];  
char cep[9];
```

```
struct{  
    char nome[20];  
    char sobrenome[20];  
    char cidade[15];  
    char estado[2];  
    char cep[9];  
} campos;
```

## Campos com tamanho fixo

- O espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (desperdício)

## Campos com tamanho fixo

- O espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (desperdício)
- Solução inapropriada quando se tem uma grande quantidade de dados com tamanho variável

## Campos com tamanho fixo

- O espaço alocado (e não usado) aumenta desnecessariamente o tamanho do arquivo (desperdício)
- Solução inapropriada quando se tem uma grande quantidade de dados com tamanho variável
- Razoável apenas se o comprimento dos campos é realmente fixo, ou apresenta pouca variação

## Campos com indicador de comprimento

- Tamanho de cada campo armazenado imediatamente antes do dado
- Tamanho do campo menor que 256 bytes
- Espaço necessário para armazenar a informação: um único byte

```
05Maria05Rua 10312314Ribeirão Preto
04José05Rua J023212Monte Carmelo
03Ana06Rua 22027310Uberlândia
```

# Campos separados por delimitadores

- Caractere(s) especial(ais) (que não fazem parte do dado)
  - Escolhido(s) para ser(em) inserido(s) ao final de cada campo
- Ex.: para o campo nome
  - pode-se utilizar /, tab, #, etc ...
  - não pode-se usar espaços em branco ...

```
Maria | Rua 1 | 123 | Ribeirão Preto  
José | Rua J | 32 | Monte Carmelo  
Ana | Rua 22 | 73 | Uberlândia
```

# Uso de tags expressão “keyword=value”

- **Vantagens**

- campo fornece informação semântica sobre si
- fica mais fácil identificar o conteúdo do arquivo
- fica mais fácil identificar campos “vazios”
- keyword não aparece

- **Desvantagem**

- as keywords podem ocupar uma porção significativa do arquivo

```
Nome=Maria | Endereço=Rua 1 | Número=123 | Cidade=Ribeirão Preto |  
Nome=José | Endereço=Rua J | Número=32 | Cidade=Monte Carmelo |  
Nome=Ana | Endereço=Rua 22 | Número=73 | Cidade=Uberlândia |
```

# Organização em registros

## Registro

- conjunto de campos agrupados, os quais estão logicamente associados a uma mesma entidade



# Organização em registros

## Registro

- conjunto de campos agrupados, os quais estão logicamente associados a uma mesma entidade
- permite a representação de um arquivo em um nível de organização mais alto

# Organização em registros

## Registro

- conjunto de campos agrupados, os quais estão logicamente associados a uma mesma entidade
- permite a representação de um arquivo em um nível de organização mais alto

Assim como o conceito de campo, um registro é uma ferramenta conceitual, que não necessariamente existe no sentido físico

# Métodos para organização em registros

- 1 Forçar todos os registros para um tamanho fixo
- 2 Forçar todos os registros para conterem um número fixo de campos
- 3 Começar cada registro com um indicador de tamanho
- 4 Usar índice para manter informação de endereçamento
- 5 Colocar delimitadores entre registros

# Registros de tamanho fixo

- Todos os registros têm o mesmo número de bytes

# Registros de tamanho fixo

- Todos os registros têm o mesmo número de bytes
- Um dos métodos mais comuns de organização de arquivos

# Registros de tamanho fixo

- Todos os registros têm o mesmo número de bytes
- Um dos métodos mais comuns de organização de arquivos
- Pode-se ter:

# Registros de tamanho fixo

- Todos os registros têm o mesmo número de bytes
- Um dos métodos mais comuns de organização de arquivos
- Pode-se ter:
  - registros de tamanho fixo com campos de tamanho fixo

# Registros de tamanho fixo

- Todos os registros têm o mesmo número de bytes
- Um dos métodos mais comuns de organização de arquivos
- Pode-se ter:
  - registros de tamanho fixo com campos de tamanho fixo
  - registros de tamanho fixo com campos de tamanho variável



# Registros de tamanho fixo

## Registro de tamanho fixo e campos de tamanho fixo:

Maria	Rua 1	123	Ribeirão Preto
José	Rua J	32	Monte Carmelo
Ana	Rua 22	73	Uberlândia

## Registro de tamanho fixo e campos de tamanho variável:

Maria	Rua 1	123	Ribeirão Preto	<----- Espaço Vazio ----->
José	Rua J	32	Monte Carmelo	<----- Espaço Vazio ----->
Ana	Rua 22	73	Uberlândia	<----- Espaço Vazio ----->

# Registros com número fixo de campos

- Cada registro contém um número fixo de campos
  - o tamanho do registro, em bytes, é variável
- Neste caso, os campos seriam separados por delimitadores

## Registros com número fixo de campos:

```
Maria | Rua 1 | 123 | Ribeirão Preto | José | Rua J | 32 | Monte Carmelo | Ana | Rua  
22 | 73 | Uberlândia |
```

# Indicador de tamanho para registros

- O indicador que precede o registro fornece o seu tamanho total, em bytes
- Os campos são separados internamente por delimitadores
- Método muito utilizado para manipular registros de tamanho variável

## Registros iniciados por indicador de tamanho:

```
27Maria | Rua 1 | 123 | Ribeirão Preto | 24José | Rua J | 32 | Monte Carmelo | 21Ana  
| Rua 22 | 73 | Uberlândia |
```

# Utilizar um índice

- Arquivo secundário que mantém informações sobre o endereço do primeiro byte de cada registro

# Utilizar um índice

- Arquivo secundário que mantém informações sobre o endereço do primeiro byte de cada registro
  - indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo

# Utilizar um índice

- Arquivo secundário que mantém informações sobre o endereço do primeiro byte de cada registro
  - indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo
- *Byte offset*

# Utilizar um índice

- Arquivo secundário que mantém informações sobre o endereço do primeiro byte de cada registro
  - indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo
- *Byte offset*
  - permite encontrar o começo de cada registro

# Utilizar um índice

- Arquivo secundário que mantém informações sobre o endereço do primeiro byte de cada registro
  - indica o deslocamento (*byte offset*) de cada registro relativo ao início do arquivo
- *Byte offset*
  - permite encontrar o começo de cada registro
  - permite calcular o tamanho dos registros



# Utilizar um índice

- Característica adicional
  - campos separados por delimitadores

## Arquivos de Dados + Arquivo de Índices:

```
Dados: Maria | Rua 1 | 123 | Ribeirão Preto | José | Rua J | 32 | Monte Carmelo |  
      Ana | Rua 22 | 73 | Uberlândia |  
Índice: 00 31 58
```

- Impacto
  - Vantagem: flexibilidade
  - Desvantagem: necessidade de acessar dois arquivos e armazenar conjuntamente

# Utilizar delimitadores

- Separar os registros com delimitadores análogos aos de fim de campo
  - o delimitador de campos é mantido, sendo que o método combina os dois delimitadores

## Registro delimitado por marcador (#):

```
Maria | Rua 1 | 123 | Ribeirão Preto | # José | Rua J | 32 | Monte Carmelo | # Ana | Rua  
22 | 73 | Uberlândia |
```

# Observações

- Nenhum dos métodos descritos é apropriado para todas as situações

# Observações

- Nenhum dos métodos descritos é apropriado para todas as situações
- Escolha do método depende da natureza dos dados para o que eles serão usados

# Acesso a arquivos x Organização de arquivos

- Organização de Arquivos

# Acesso a arquivos x Organização de arquivos

- Organização de Arquivos
  - registros de tamanho fixo

# Acesso a arquivos x Organização de arquivos

- Organização de Arquivos
  - registros de tamanho fixo
  - registros de tamanho variável

# Acesso a arquivos x Organização de arquivos

- Organização de Arquivos
  - registros de tamanho fixo
  - registros de tamanho variável
- Acesso a Arquivos



# Acesso a arquivos x Organização de arquivos

- Organização de Arquivos
  - registros de tamanho fixo
  - registros de tamanho variável
- Acesso a Arquivos
  - acesso sequencial

# Acesso a arquivos x Organização de arquivos

- Organização de Arquivos
  - registros de tamanho fixo
  - registros de tamanho variável
- Acesso a Arquivos
  - acesso sequencial
  - acesso direto

# Acesso a arquivos x Organização de arquivos

- Considerações a respeito da organização do arquivo

# Acesso a arquivos x Organização de arquivos

- Considerações a respeito da organização do arquivo
  - arquivo pode ser dividido em campos?

# Acesso a arquivos x Organização de arquivos

- Considerações a respeito da organização do arquivo
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?

# Acesso a arquivos x Organização de arquivos

- Considerações a respeito da organização do arquivo
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?
  - registros têm tamanho fixo ou variável?

# Acesso a arquivos x Organização de arquivos

- Considerações a respeito da organização do arquivo
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?
  - registros têm tamanho fixo ou variável?
  - como separar os registros?

# Acesso a arquivos x Organização de arquivos

- Considerações a respeito da organização do arquivo
  - arquivo pode ser dividido em campos?
  - os campos são agrupados em registros?
  - registros têm tamanho fixo ou variável?
  - como separar os registros?
  - como identificar o espaço utilizado e o “lixo”?



# Acesso a arquivos x Organização de arquivos

- Existem muitas respostas para estas questões:

# Acesso a arquivos x Organização de arquivos

- Existem muitas respostas para estas questões:
  - A escolha de uma organização em particular depende, entre outras coisas, do que se vai fazer com o arquivo

# Acesso a arquivos x Organização de arquivos

- Existem muitas respostas para estas questões:
  - A escolha de uma organização em particular depende, entre outras coisas, do que se vai fazer com o arquivo
  - Arquivos que devem conter registros com tamanhos muito diferentes, devem utilizar registros de tamanho variável

# Acesso a arquivos x Organização de arquivos

- Existem muitas respostas para estas questões:
  - A escolha de uma organização em particular depende, entre outras coisas, do que se vai fazer com o arquivo
  - Arquivos que devem conter registros com tamanhos muito diferentes, devem utilizar registros de tamanho variável
  - Como acessar esses registros diretamente?

# Acesso a arquivos x Organização de arquivos

- Existem muitas respostas para estas questões:
  - A escolha de uma organização em particular depende, entre outras coisas, do que se vai fazer com o arquivo
  - Arquivos que devem conter registros com tamanhos muito diferentes, devem utilizar registros de tamanho variável
  - Como acessar esses registros diretamente?
  - Existem também as limitações da linguagem

# Acesso a arquivos x Organização de arquivos

- Existem muitas respostas para estas questões:
  - A escolha de uma organização em particular depende, entre outras coisas, do que se vai fazer com o arquivo
  - Arquivos que devem conter registros com tamanhos muito diferentes, devem utilizar registros de tamanho variável
  - Como acessar esses registros diretamente?
  - Existem também as limitações da linguagem
    - C permite acesso a qualquer byte, e o programador pode implementar acesso direto a registros de tamanho variável

# Exercício 1

- Considere os seguintes produtos de um supermercado:
  - Refrigerante - Coca-cola - 600ml - R\$7,00
  - Suco de uva - Casa Madeira - 1L - R\$14,00
  - Sabonete - Lux - 1un - R\$3,00
- Codifique um programa em C que armazene as informações acima em um arquivo, utilizando a estratégia de campos separados pelo delimitador '|'

```
1 #include <stdio.h>
2 #include <string.h>
3
4 struct produto {
5     char tipo[50];
6     char nome[50];
7     int volume;
8     char unidade_de_medida[50];
9     float preco;
10 };
11
12 void preencher_produto(struct produto *prod, char tipo[], char nome[], int volume,
13     char unidade_de_medida[], float preco) {
14     strcpy(prod->tipo, tipo);
15     strcpy(prod->nome, nome);
16     prod->volume = volume;
17     strcpy(prod->unidade_de_medida, unidade_de_medida);
18     prod->preco = preco;
19 }
```

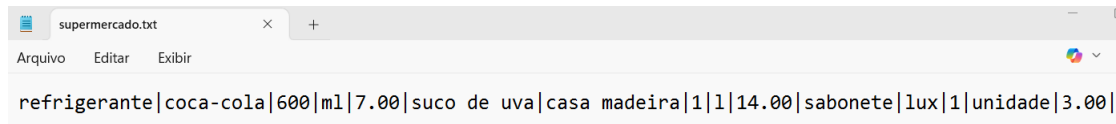


```

1 void main() {
2     FILE *f;
3     int i;
4     struct produto produtos[3];
5
6     preencher_produto(&produtos[0], "refrigerante", "coca-cola", 600, "ml", 7.00);
7     preencher_produto(&produtos[1], "suco de uva", "casa madeira", 1, "l", 14.00);
8     preencher_produto(&produtos[2], "sabonete", "lux", 1, "unidade", 3.00);
9     f = fopen("supermercado.txt", "w");
10    for (i = 0; i < 3; i++) {
11        fputs(produtos[i].tipo, f);
12        fputc('|', f);
13        fputs(produtos[i].nome, f);
14        fputc('|', f);
15        fprintf(f, "%d", produtos[i].volume);
16        fputc('|', f);
17        fputs(produtos[i].unidade_de_medida, f);
18        fputc('|', f);
19        fprintf(f, "%.2f", produtos[i].preco);
20        fputc('|', f);
21    }
22    fclose(f);
23 }

```

# Exercício 1 - Resultado



A screenshot of a text editor window titled "supermercado.txt". The window has a menu bar with "Arquivo", "Editar", and "Exibir". The main text area contains a single line of data separated by vertical bars (pipe characters). The data is: "refrigerante|coca-cola|600|ml|7.00|suco de uva|casa madeira|1|1|14.00|sabonete|lux|1|unidade|3.00|".

```
refrigerante|coca-cola|600|ml|7.00|suco de uva|casa madeira|1|1|14.00|sabonete|lux|1|unidade|3.00|
```

## Exercício 2

- Considere os seguintes produtos de um supermercado:
  - Refrigerante - Coca-cola - 600ml - R\$7,00
  - Suco de uva - Casa Madeira - 1L - R\$14,00
  - Sabonete - Lux - 1un - R\$3,00
- Codifique um programa em C que armazene as informações acima em um arquivo, utilizando a estratégia de registros separados pelo delimitador '#', com campos separados pelo delimitador '|'

```

1 #include <stdio.h>
2 #include <string.h>
3
4 struct produto
5 {
6     char tipo[50];
7     char nome[50];
8     int volume;
9     char unidade_de_medida[50];
10    float preco;
11 };
12
13 void preencher_produto(struct produto *prod, char tipo[], char nome[], int volume,
14                        char unidade_de_medida[], float preco)
15 {
16     strcpy(prod->tipo, tipo);
17     strcpy(prod->nome, nome);
18     prod->volume = volume;
19     strcpy(prod->unidade_de_medida, unidade_de_medida);
20     prod->preco = preco;
21 }

```

```

1 void main() {
2     FILE *f;
3     int i;
4     struct produto produtos[3];
5     preencher_produto(&produtos[0], "refrigerante", "coca-cola", 600, "ml", 7.00);
6     preencher_produto(&produtos[1], "suco de uva", "casa madeira", 1, "l", 14.00);
7     preencher_produto(&produtos[2], "sabonete", "lux", 1, "unidade", 3.00);
8     f = fopen("supermercado.txt", "w");
9     for (i = 0; i < 3; i++) {
10         fputc('#', f);
11         fputs(produtos[i].tipo, f);
12         fputc('|', f);
13         fputs(produtos[i].nome, f);
14         fputc('|', f);
15         fprintf(f, "%d", produtos[i].volume);
16         fputc('|', f);
17         fputs(produtos[i].unidade_de_medida, f);
18         fputc('|', f);
19         fprintf(f, "%.2f", produtos[i].preco);
20         fputc('|', f);
21     }
22     fclose(f);
23 }

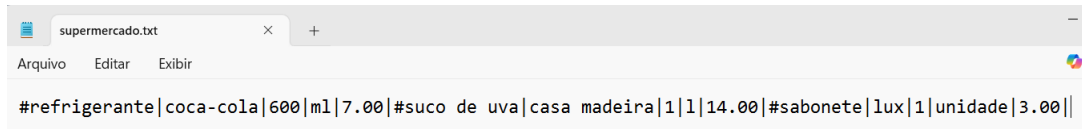
```

```

1 void main() {
2     FILE *f;
3     int i;
4     struct produto produtos[3];
5     preencher_produto(&produtos[0], "refrigerante", "coca-cola", 600, "ml", 7.00);
6     preencher_produto(&produtos[1], "suco de uva", "casa madeira", 1, "l", 14.00);
7     preencher_produto(&produtos[2], "sabonete", "lux", 1, "unidade", 3.00);
8     f = fopen("supermercado.txt", "w");
9     for (i = 0; i < 3; i++) {
10         fputc('#', f);
11         fputs(produtos[i].tipo, f);
12         fputc('|', f);
13         fputs(produtos[i].nome, f);
14         fputc('|', f);
15         fprintf(f, "%d", produtos[i].volume);
16         fputc('|', f);
17         fputs(produtos[i].unidade_de_medida, f);
18         fputc('|', f);
19         fprintf(f, "%.2f", produtos[i].preco);
20         fputc('|', f);
21     }
22     fclose(f);
23 }

```

## Exercício 2 - Resultado



The screenshot shows a text editor window with the title bar 'supermercado.txt'. The menu bar contains 'Arquivo', 'Editar', and 'Exibir'. The text area contains a single line of CSV data: '#refrigerante|coca-cola|600|ml|7.00|#suco de uva|casa madeira|1|1|14.00|#sabonete|lux|1|unidade|3.00|'.

```
#refrigerante|coca-cola|600|ml|7.00|#suco de uva|casa madeira|1|1|14.00|#sabonete|lux|1|unidade|3.00|
```

## Exercício 3

- Considere os seguintes produtos de um supermercado:
  - Refrigerante - Coca-cola - 600ml - R\$7,00
  - Suco de uva - Casa Madeira - 1L - R\$14,00
  - Sabonete - Lux - 1un - R\$3,00
- Codifique um programa em C que armazene as informações acima em um arquivo, utilizando a estratégia de registros de tamanho fixo, com campos de tamanho variável.



```

1 #include <stdio.h>
2 #include <string.h>
3
4 struct produto
5 {
6     char tipo[50];
7     char nome[50];
8     int volume;
9     char unidade_de_medida[50];
10    float preco;
11 };
12
13 void preencher_produto(struct produto *prod, char tipo[], char nome[], int volume,
14                        char unidade_de_medida[], float preco)
15 {
16     strcpy(prod->tipo, tipo);
17     strcpy(prod->nome, nome);
18     prod->volume = volume;
19     strcpy(prod->unidade_de_medida, unidade_de_medida);
20     prod->preco = preco;
21 }

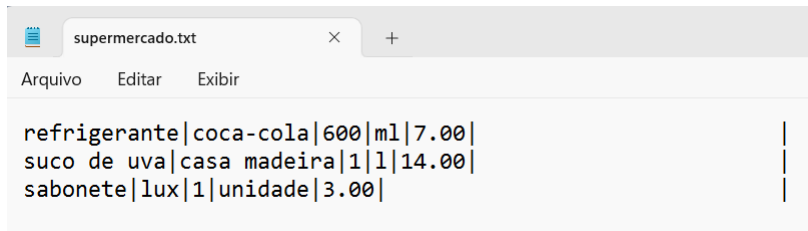
```

```
1 void main() {  
2  
3     FILE *f;  
4     int i;  
5     struct produto produtos[3];  
6  
7     preencher_produto(&produtos[0], "refrigerante", "coca-cola", 600, "ml", 7.00);  
8     preencher_produto(&produtos[1], "suco de uva", "casa madeira", 1, "l", 14.00);  
9     preencher_produto(&produtos[2], "sabonete", "lux", 1, "unidade", 3.00);  
10  
11     f = fopen("supermercado.txt", "w");
```

```
1  for (i = 0; i < 3; i++) {
2      fputs(produtos[i].tipo, f);
3      total += strlen(produtos[i].tipo);
4
5      fputc('|', f);
6      fputs(produtos[i].nome, f);
7      total += strlen(produtos[i].nome);
8
9      fputc('|', f);
10     fprintf(f, "%d", produtos[i].volume);
11     sprintf(aux, "%d", produtos[i].volume);
12     total += strlen(aux);
13
14     fputc('|', f);
15     fputs(produtos[i].unidade_de_medida, f);
16     total += strlen(produtos[i].unidade_de_medida);
17
18     fputc('|', f);
19     fprintf(f, "%.2f", produtos[i].preco);
20     sprintf(aux, "%g", produtos[i].preco);
21     total += strlen(aux);
```

```
1      fputc('|', f);
2      fprintf(f, "%.2f", produtos[i].preco);
3      sprintf(aux, "%g", produtos[i].preco);
4      total += strlen(aux);
5
6      fputc('|', f);
7      restante = tamanho_fixo - total;
8
9      for(j=0; j<restante; j++){
10         fputc(' ', f);
11     }
12
13     fputc('|', f);
14     fputc('\n', f);
15     total = 0;
16 }
17 fclose(f);
18 }
```

## Exercício 3 - Resultado



The screenshot shows a text editor window with the title 'supermercado.txt'. The editor contains a table of supermarket items with three columns: item name, quantity, and price. The items are: refrigerante (coca-cola, 600ml, 7.00), suco de uva (casa madeira, 1l, 14.00), and sabonete (lux, 1 unidade, 3.00). The table is displayed as a simple text-based representation with vertical bars separating the columns.

refrigerante	coca-cola	600ml	7.00
suco de uva	casa madeira	1l	14.00
sabonete	lux	1 unidade	3.00

## Faculdade de Computação - FACOM

### Bacharelado em Sistemas de Informação

**Prof. Thiago Pirola Ribeiro**