

Projecto de LI3

Generated by Doxygen 1.8.15



<b>1 LI3</b>	<b>1</b>
<b>2 Data Structure Index</b>	<b>3</b>
2.1 Data Structures	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Data Structure Documentation</b>	<b>7</b>
4.1 query Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Field Documentation	7
4.1.2.1 precototal	7
4.1.2.2 unidcompradas	7
4.2 vendas Struct Reference	8
4.2.1 Detailed Description	8
4.2.2 Field Documentation	8
4.2.2.1 cliente	8
4.2.2.2 filial	8
4.2.2.3 mes	8
4.2.2.4 preco	8
4.2.2.5 prod	9
4.2.2.6 tcompria	9
4.2.2.7 unidades	9
<b>5 File Documentation</b>	<b>11</b>
5.1 init.c File Reference	11
5.1.1 Detailed Description	11
5.1.2 Macro Definition Documentation	11
5.1.2.1 _GNU_SOURCE	12
5.1.3 Function Documentation	12
5.1.3.1 initt()	12
5.2 init.h File Reference	12
5.2.1 Detailed Description	12
5.2.2 Macro Definition Documentation	13
5.2.2.1 _GNU_SOURCE	13
5.2.3 Function Documentation	13
5.2.3.1 initt()	13
5.3 main.c File Reference	13
5.3.1 Detailed Description	14
5.3.2 Macro Definition Documentation	14
5.3.2.1 _GNU_SOURCE	14
5.3.3 Function Documentation	14
5.3.3.1 main()	14

5.4 queries.c File Reference . . . . .	14
5.4.1 Detailed Description . . . . .	15
5.4.2 Macro Definition Documentation . . . . .	15
5.4.2.1 _GNU_SOURCE . . . . .	15
5.4.3 Function Documentation . . . . .	15
5.4.3.1 imprime_ultimo() . . . . .	15
5.4.3.2 linha_mais_longa() . . . . .	16
5.4.3.3 testa_brp() . . . . .	16
5.5 queries.h File Reference . . . . .	16
5.5.1 Detailed Description . . . . .	17
5.5.2 Macro Definition Documentation . . . . .	17
5.5.2.1 _GNU_SOURCE . . . . .	17
5.5.3 Function Documentation . . . . .	17
5.5.3.1 imprime_ultimo() . . . . .	17
5.5.3.2 linha_mais_longa() . . . . .	17
5.5.3.3 testa_brp() . . . . .	18
5.6 README.md File Reference . . . . .	18
5.7 valida.c File Reference . . . . .	18
5.7.1 Detailed Description . . . . .	20
5.7.2 Macro Definition Documentation . . . . .	20
5.7.2.1 _GNU_SOURCE . . . . .	20
5.7.2.2 CAMPOSVENDA . . . . .	20
5.7.2.3 staAux . . . . .	20
5.7.2.4 TAMCLIENTES . . . . .	20
5.7.2.5 TAMPROD . . . . .	20
5.7.2.6 TAMVENDAS . . . . .	20
5.7.3 Typedef Documentation . . . . .	21
5.7.3.1 Query . . . . .	21
5.7.3.2 Vendas . . . . .	21
5.7.4 Function Documentation . . . . .	21
5.7.4.1 clienttoArray() . . . . .	21
5.7.4.2 escreveArray() . . . . .	21
5.7.4.3 fazStruct() . . . . .	22
5.7.4.4 initArrayTree() . . . . .	22
5.7.4.5 printelements() . . . . .	22
5.7.4.6 prodtoArray() . . . . .	22
5.7.4.7 prodToTree() . . . . .	23
5.7.4.8 validclient() . . . . .	23
5.7.4.9 validProd() . . . . .	23
5.7.4.10 validvendas() . . . . .	23
5.7.4.11 verclien() . . . . .	24
5.7.4.12 verfilial() . . . . .	24

5.7.4.13 vermes()	24
5.7.4.14 verpreco()	25
5.7.4.15 verprod()	25
5.7.4.16 vertcompra()	25
5.7.4.17 verunidadesvend()	26
5.7.5 Variable Documentation	26
5.7.5.1 arrayprod	26
5.7.5.2 clientes	26
5.7.5.3 produtos	26
5.7.5.4 teste	27
5.7.5.5 validadas	27
5.7.5.6 ven	27
5.7.5.7 venda	27
5.8 valida.h File Reference	27
5.8.1 Detailed Description	28
5.8.2 Macro Definition Documentation	28
5.8.2.1 _GNU_SOURCE	29
5.8.2.2 CAMPOSVENDA	29
5.8.2.3 staAux	29
5.8.2.4 TAMCLIENTES	29
5.8.2.5 TAMPROD	29
5.8.2.6 TAMVENDAS	29
5.8.3 Function Documentation	29
5.8.3.1 clienttoArray()	29
5.8.3.2 escreveArray()	30
5.8.3.3 fazStruct()	30
5.8.3.4 prodtoArray()	30
5.8.3.5 validclient()	31
5.8.3.6 validProd()	31
5.8.3.7 validvendas()	31
5.8.3.8 verclien()	31
5.8.3.9 verfilial()	32
5.8.3.10 vermes()	32
5.8.3.11 verprod()	32
5.8.3.12 vertcompra()	33
5.8.3.13 verunidadesvend()	33
5.8.3.14 verunidec()	33
5.8.4 Variable Documentation	34
5.8.4.1 clientes	34
5.8.4.2 produtos	34
5.8.4.3 venda	34



# Chapter 1

## LI3

Projeto de LI3 - 2018/2019

Sistema de Gestão das Vendas de uma Distribuidora com 3 Filiais

Introdução e Objectivos.

O projecto de C da disciplina de LI3 de MIEI tem por objectivo fundamental ajudar à consolidação experimental dos conhecimentos teóricos e práticos adquiridos nas UCs anteriores e a introdução de novos conceitos. Os objectivos de LI3, e deste trabalho, não se restringem apenas a aumentar os conhecimentos dos alunos na linguagem C, o que seria até questionável, mas, e talvez fundamentalmente, apresentar aos alunos de forma pragmática, os desafios que se colocam a quem concebe e programa aplicações software (em qualquer linguagem), quando passamos a realizar a designada programação em larga escala, ou seja, aplicações com grandes volumes de dados e com mais elevada complexidade algorítmica e estrutural. De facto, quando passamos para tais patamares de complexidade, torna-se imperioso conhecer e usar os melhores princípios da Engenharia de Software, de modo a que tais projectos de software, em geral realizados por equipas, possam ser concebidos com melhor estrutura, de modo a que sejam mais facilmente modificáveis, e sejam, apesar da complexidade, o mais optimizados possível a todos os níveis. Para que tal seja possível teremos que introduzir novos princípios de programação, mais adequados à programação em grande escala, designadamente: -> Modularidade e encapsulamento de dados usando construções da linguagem; -> Criação de código reutilizável; -> Escolha optimizada das estruturas de dados e reutilização; -> Testes de performance e até profiling. Este projecto, a desenvolver em trabalho de grupo (de no máximo 3 alunos), visa a experimentação e aplicação destas práticas de desenvolvimento de software usando a linguagem C, práticas que são extensíveis a outras linguagens e paradigmas.





## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">query</a>	Struct usada numa querie . . . . .	<a href="#">7</a>
<a href="#">vendas</a>	Struct de Venda . . . . .	<a href="#">8</a>



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">init.c</a>	Ficheio init . . . . .	11
<a href="#">init.h</a>	Head de init . . . . .	12
<a href="#">main.c</a>	Ficheio main . . . . .	13
<a href="#">queries.c</a>	Ficheio queries . . . . .	14
<a href="#">queries.h</a>	Head de queries . . . . .	16
<a href="#">valida.c</a>	Ficheio valida . . . . .	18
<a href="#">valida.h</a>	Head de valida . . . . .	27



## Chapter 4

# Data Structure Documentation

### 4.1 query Struct Reference

Struct usada numa querie.

#### Data Fields

- int [unidcompradas](#)
- double [precototal](#)

#### 4.1.1 Detailed Description

Struct usada numa querie.

#### 4.1.2 Field Documentation

##### 4.1.2.1 precototal

```
double query::precototal
```

##### 4.1.2.2 unidcompradas

```
int query::unidcompradas
```

The documentation for this struct was generated from the following file:

- [valida.c](#)

## 4.2 vendas Struct Reference

Struct de Venda.

### Data Fields

- char \* [prod](#)
- double [preco](#)
- int [unidades](#)
- char \* [tcompra](#)
- char \* [cliente](#)
- int [mes](#)
- int [filial](#)

### 4.2.1 Detailed Description

Struct de Venda.

### 4.2.2 Field Documentation

#### 4.2.2.1 cliente

```
char* vendas::cliente
```

#### 4.2.2.2 filial

```
int vendas::filial
```

#### 4.2.2.3 mes

```
int vendas::mes
```

#### 4.2.2.4 preco

```
double vendas::preco
```

#### 4.2.2.5 prod

```
char* vendas::prod
```

#### 4.2.2.6 tcompra

```
char* vendas::tcompra
```

#### 4.2.2.7 unidades

```
int vendas::unidades
```

The documentation for this struct was generated from the following file:

- [valida.c](#)





## Chapter 5

# File Documentation

### 5.1 init.c File Reference

Ficheio init.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "valida.h"
#include <glib.h>
```

#### Macros

- `#define \_GNU\_SOURCE`

#### Functions

- `int initt(char *argv[])`  
*Função initt, chama todas as outras funções.*

#### 5.1.1 Detailed Description

Ficheio init.

Ficheiro onde é chamada todas as outras funções.

#### 5.1.2 Macro Definition Documentation

### 5.1.2.1 `_GNU_SOURCE`

```
#define _GNU_SOURCE
```

## 5.1.3 Function Documentation

### 5.1.3.1 `initt()`

```
int initt (
    char * argv[ ] )
```

Função `initt`, chama todas as outras funções.

#### Parameters

<code>argv</code>	Nomes do ficheiros de Produtos, Clientes e Vendas. por esta ordem.
-------------------	--

#### Returns

`int`

## 5.2 `init.h` File Reference

head de `init`

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <glib.h>
```

#### Macros

- `#define \_GNU\_SOURCE`

#### Functions

- `int initt (char *argv[ ])`  
*Função `initt`, chama todas as outras funções.*

### 5.2.1 Detailed Description

head de `init`

Contem as definições de todas as funcoes unicas de [init.c](#)

## 5.2.2 Macro Definition Documentation

### 5.2.2.1 \_GNU\_SOURCE

```
#define _GNU_SOURCE
```

## 5.2.3 Function Documentation

### 5.2.3.1 initt()

```
int initt (
    char * argv[] )
```

Função initt, chama todas as outras funções.

#### Parameters

<i>argv</i>	Nomes do ficheiros de Produtos, Clientes e Vendas. por esta ordem.
-------------	--

#### Returns

int

## 5.3 main.c File Reference

Ficheio main.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "init.h"
#include <glib.h>
```

### Macros

- `#define \_GNU\_SOURCE`

### Functions

- int `main` (int argc, char \*argv[])

*Função Main, onde a função initt é chamada; também é feita a contagem do tempo de execução do programa.*

### 5.3.1 Detailed Description

Ficheio main.

Ficheiro principal do programa.

### 5.3.2 Macro Definition Documentation

#### 5.3.2.1 \_GNU\_SOURCE

```
#define _GNU_SOURCE
```

### 5.3.3 Function Documentation

#### 5.3.3.1 main()

```
int main (
    int argc,
    char * argv[ ] )
```

Função Main, onde a função initt é chamada; também é feita a contagem do tempo de execução do programa.

#### Parameters

<i>argc</i>	
<i>argv</i>	Nomes do ficheiros de Produtos, Clientes e Vendas. por esta ordem.

#### Returns

int

## 5.4 queries.c File Reference

Ficheio queries.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <glib.h>
#include "valida.h"
```

## Macros

- `#define _GNU_SOURCE`

## Functions

- `int linha_mais_longa (char *array[])`  
*Função que descobre o tamanho da linha mais longa das vendas.*
- `void imprime_ultimo (char *array[])`  
*Função que imprime ultimo cliente.*
- `void testa_brp ()`  
*Função de testes.*

### 5.4.1 Detailed Description

Ficheio queries.

Ficheiro onde é feitos as queries do trabalho.

### 5.4.2 Macro Definition Documentation

#### 5.4.2.1 \_GNU\_SOURCE

```
#define _GNU_SOURCE
```

### 5.4.3 Function Documentation

#### 5.4.3.1 imprime\_ultimo()

```
void imprime_ultimo (  
    char * array[] )
```

Função que imprime ultimo cliente.

#### Parameters

<code>array</code>	
--------------------	--

#### 5.4.3.2 linha\_mais\_longa()

```
int linha_mais_longa (
    char * array[] )
```

Função que descobre o tamanho da linha mais longa das vendas.

##### Parameters

array	
-------	--

##### Returns

int

#### 5.4.3.3 testa\_brp()

```
void testa_brp ( )
```

Função de testes.

## 5.5 queries.h File Reference

head de queries

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <glib.h>
```

### Macros

- `#define \_GNU\_SOURCE`

### Functions

- void [testa\\_brp](#) ()  
*Função de testes.*
- int [linha\\_mais\\_longa](#) (char \*array[])  
*Função que descobre o tamanho da linha mais longa das vendas.*
- void [imprime\\_ultimo](#) (char \*array[])  
*Função que imprime ultimo cliente.*

### 5.5.1 Detailed Description

head de queries

Contem as definições de todas as funcoes unicas de [queries.c](#)

### 5.5.2 Macro Definition Documentation

#### 5.5.2.1 \_GNU\_SOURCE

```
#define _GNU_SOURCE
```

### 5.5.3 Function Documentation

#### 5.5.3.1 imprime\_ultimo()

```
void imprime_ultimo (
    char * array[] )
```

Função que imprime ultimo cliente.

##### Parameters

<i>array</i>	
--------------	--

#### 5.5.3.2 linha\_mais\_longa()

```
int linha_mais_longa (
    char * array[] )
```

Função que descobre o tamanho da linha mais longa das vendas.

##### Parameters

<i>array</i>	
--------------	--

##### Returns

int

### 5.5.3.3 testa\_brp()

```
void testa_brp ( )
```

Função de testes.

## 5.6 README.md File Reference

## 5.7 valida.c File Reference

Ficheio valida.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <glib.h>
```

### Data Structures

- struct [vendas](#)  
*Struct de Venda.*
- struct [query](#)  
*Struct usada numa querie.*

### Macros

- #define [\\_GNU\\_SOURCE](#)
- #define [CAMPOSVENDA](#) 7  
*Defines para tamanhos de arrays.*
- #define [TAMPROD](#) 200000
- #define [TAMCLIENTES](#) 20000
- #define [TAMVENDAS](#) 1000000
- #define [staAux](#) 50

### Typedefs

- typedef struct [vendas](#) \* [Vendas](#)  
*Struct de Venda.*
- typedef struct [query](#) [Query](#)  
*Struct usada numa querie.*



## Functions

- void [printelements](#) (gpointer key, gpointer value, gpointer user\_data)  
*Função que imprime um elemento de um nodo de arvore. Será usada com g\_tree\_foreach(), logo irá imprimir todos os nodos da árvore.*
- void [initArrayTree](#) (GTree \*arrayprod[90])
- void [prodToTree](#) (char \*campos, GTree \*\*arrayprod)
- int [verprod](#) (char \*campos)  
*Função que verifica se o Produto é válido. Procura a venda dada como input no array de Produtos validos.*
- int [verclien](#) (char \*campos)  
*Função que verifica se o Cliente é válido.*
- int [verpreco](#) (double unidec)  
*Função que verifica se Preço do Produto é válido.*
- int [verunidadesvend](#) (int unidades)  
*Função que verifica se o número de unidades compradas é válido.*
- int [vertcompra](#) (char \*compra)  
*Função que verifica se o tipo de compra é válido.*
- int [vermes](#) (int mes)  
*Função que verifica se o mês da compra é válido.*
- int [verfilial](#) (int filial)  
*Função que verifica se a filial é válida.*
- int [fazStruct](#) (char \*linhaVendaOk)  
*Função que escreve cada linha do array de vendas num array de struct; também verifica se cada elemento da venda é valido.*
- void [escreveArray](#) (FILE \*fp, char \*array[])  
*Função que dado um apontador para um ficheiro e um array de "strings" (vaziu), preenche o array com o que esta no ficheiro.*
- void [validProd](#) (char produtos[])  
*Função que valida Produtos antes de os inserir num array.*
- void [prodtoArray](#) (char \*fich)
- void [validclient](#) (char clientes[])  
*Função que valida Clientes antes de os inserir num array.*
- void [clienttoArray](#) (char \*fich)  
*Função que lê os Cleintes do ficheiro e os poe num array de strings. Tambem faz a validação usando a função validclient.*
- void [validvendas](#) (char \*fich)  
*Função que lê as vendas do ficheiro e as poe num array de strings. Tambem faz a validação.*

## Variables

- char \* [produtos](#) [[TAMPROD](#)]  
*Arrays e variaveis definidos globalmente para todas as funcoes.*
- char \* [clientes](#) [[TAMCLIENTES](#)]
- char \* [venda](#) [[TAMVENDAS](#)]
- [Vendas ven](#) [[TAMVENDAS](#)]
- int [teste](#) = 0
- int [validadas](#) = 0
- GTree \* [arrayprod](#) [90]

### 5.7.1 Detailed Description

Ficheio valida.

Ficheiro onde é feito todas as validaações de dados. tambem é onde se mexe nas estruturas.

### 5.7.2 Macro Definition Documentation

#### 5.7.2.1 \_GNU\_SOURCE

```
#define _GNU_SOURCE
```

#### 5.7.2.2 CAMPOSVENDA

```
#define CAMPOSVENDA 7
```

Defines para tamanhos de arrays.

#### 5.7.2.3 staAux

```
#define staAux 50
```

#### 5.7.2.4 TAMCLIENTES

```
#define TAMCLIENTES 20000
```

#### 5.7.2.5 TAMPROD

```
#define TAMPROD 200000
```

#### 5.7.2.6 TAMVENDAS

```
#define TAMVENDAS 1000000
```

### 5.7.3 Typedef Documentation

#### 5.7.3.1 Query

```
typedef struct query Query
```

Struct usada numa querie.

#### 5.7.3.2 Vendas

```
typedef struct vendas* Vendas
```

Struct de Venda.

### 5.7.4 Function Documentation

#### 5.7.4.1 clienttoArray()

```
void clienttoArray (  
    char * fich )
```

Função que lê os Cleintes do ficheiro e os poes num array de strings. Tambem faz a validação usando a função validclient.

##### Parameters

<i>fich</i>	
-------------	--

#### 5.7.4.2 escreveArray()

```
void escreveArray (  
    FILE * fp,  
    char * array[] )
```

Função que dado um apontador para um ficheiro e um array de "strings" (vaziu), preenche o array com o que esta no ficheiro.

##### Parameters

<i>fp</i>	
<i>array</i>	

#### 5.7.4.3 fazStruct()

```
int fazStruct (
    char * linhaVendaOk )
```

Função que escreve cada linha do array de vendas num array de struct; também verifica se cada elemento da venda é valido.

##### Parameters

<i>linhaVendaOk</i>	- uma linha do array de vendas.
---------------------	---------------------------------

##### Returns

int

#### 5.7.4.4 initArrayTree()

```
void initArrayTree (
    GTree * arrayprod[90] )
```

#### 5.7.4.5 printelements()

```
void printelements (
    gpointer key,
    gpointer value,
    gpointer user_data )
```

Função que imprime um elemento de um nodo de arvore. Será usada com `g_tree_foreach()`, logo irá imprimir todos os nodos da árvore.

##### Parameters

<i>key</i>	(conteudo de cada nodo da árvore)
<i>value</i>	(valor associado a chave)
<i>user_data</i>	(contador - conta o numero de nodos que passou pela função)

#### 5.7.4.6 prodtoArray()

```
void prodtoArray (
    char * fich )
```

## Parameters

<i>fich</i>	
-------------	--

## 5.7.4.7 prodToTree()

```
void prodToTree (
    char * campos,
    GTree ** arrayprod )
```

## Parameters

<i>campos</i>	
<i>arrayprod</i>	

## 5.7.4.8 validclient()

```
void validclient (
    char clientes[] )
```

Função que valida Clientes antes de os inserir num array.

## Parameters

<i>clientes</i>	
-----------------	--

## 5.7.4.9 validProd()

```
void validProd (
    char produtos[] )
```

Função que valida Produtos antes de os inserir num array.

## Parameters

<i>produtos</i>	
-----------------	--

## 5.7.4.10 validvendas()

```
void validvendas (
    char * fich )
```

Função que lê as vendas do ficheiro e as põe num array de strings. Também faz a validação.

**Parameters**

<i>fich</i>	
-------------	--

**5.7.4.11 verclien()**

```
int verclien (
    char * campos )
```

Função que verifica se o Cliente é válido.

**Parameters**

<i>campos</i>	
---------------	--

**Returns**

int

**5.7.4.12 verfilial()**

```
int verfilial (
    int filial )
```

Função que verifica se a filial é válida.

**Parameters**

<i>filial</i>	
---------------	--

**Returns**

int

**5.7.4.13 vermes()**

```
int vermes (
    int mes )
```

Função que verifica se o mês da compra é válido.

**Parameters**

<i>mes</i>	
------------	--

**Returns**

int

**5.7.4.14 verpreco()**

```
int verpreco (
    double unidec )
```

Função que verifica se Preço do Produto é válido.

**Parameters**

<i>unidec</i>	
---------------	--

**Returns**

int

**5.7.4.15 verprod()**

```
int verprod (
    char * campos )
```

Função que verifica se o Produto é válido. Procura a venda dada como input no array de Produtos validos.

**Parameters**

<i>campos</i>	
---------------	--

**Returns**

int

**5.7.4.16 vertcompra()**

```
int vertcompra (
    char * compra )
```

Função que verifica se o tipo de compra é válido.

**Parameters**

<i>compra</i>	
---------------	--

**Returns**

int

**5.7.4.17 verunidadesvend()**

```
int verunidadesvend (  
    int unidades )
```

Função que verifica se o número de unidades compradas é válido.

**Parameters**

<i>unidades</i>	
-----------------	--

**Returns**

int

**5.7.5 Variable Documentation****5.7.5.1 arrayprod**

```
GTree* arrayprod[90]
```

**5.7.5.2 clientes**

```
char* clientes[TAMCLIENTES]
```

**5.7.5.3 produtos**

```
char* produtos[TAMPROD]
```

Arrays e variáveis definidos globalmente para todas as funções.



#### 5.7.5.4 teste

```
int teste = 0
```

#### 5.7.5.5 validadas

```
int validadas = 0
```

#### 5.7.5.6 ven

```
Vendas ven[TAMVENDAS]
```

#### 5.7.5.7 venda

```
char* venda[TAMVENDAS]
```

## 5.8 valida.h File Reference

head de valida

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <glib.h>
```

### Macros

- #define [\\_GNU\\_SOURCE](#)
- #define [CAMPOSVENDA](#) 7
- #define [TAMPROD](#) 200000
- #define [TAMCLIENTES](#) 20000
- #define [TAMVENDAS](#) 1000000
- #define [staAux](#) 50

## Functions

- void [validvendas](#) (char \*fich)  
*Função que lê as vendas do ficheiro e as põe num array de strings. Também faz a validação.*
- void [clienteToArray](#) (char \*fich)  
*Função que lê os Clientes do ficheiro e os põe num array de strings. Também faz a validação usando a função [validclient](#).*
- void [prodtoArray](#) (char \*fich)
- int [fazStruct](#) (char \*linhaVendaOk)  
*Função que escreve cada linha do array de vendas num array de struct; também verifica se cada elemento da venda é válido.*
- void [escreveArray](#) (FILE \*fp, char \*array[])  
*Função que dado um apontador para um ficheiro e um array de "strings" (vazio), preenche o array com o que está no ficheiro.*
- int [verprod](#) (char \*campos)  
*Função que verifica se o Produto é válido. Procura a venda dada como input no array de Produtos válidos.*
- int [verclien](#) (char \*campos)  
*Função que verifica se o Cliente é válido.*
- int [verunidec](#) (double unidec)
- int [verunidadesvend](#) (int unidades)  
*Função que verifica se o número de unidades compradas é válido.*
- int [vertcompra](#) (char \*compra)  
*Função que verifica se o tipo de compra é válido.*
- int [vermes](#) (int mes)  
*Função que verifica se o mês da compra é válido.*
- int [verfilial](#) (int filial)  
*Função que verifica se a filial é válida.*
- void [validProd](#) (char [produtos](#)[])  
*Função que valida Produtos antes de os inserir num array.*
- void [validclient](#) (char [clientes](#)[])  
*Função que valida Clientes antes de os inserir num array.*

## Variables

- char \* [produtos](#) [[TAMPROD](#)]
- char \* [clientes](#) [[TAMCLIENTES](#)]
- char \* [venda](#) [[TAMVENDAS](#)]

### 5.8.1 Detailed Description

head de valida

Contem as definições de todas as funcoes unicas de [valida.c](#)

### 5.8.2 Macro Definition Documentation

### 5.8.2.1 \_GNU\_SOURCE

```
#define _GNU_SOURCE
```

### 5.8.2.2 CAMPOSVENDA

```
#define CAMPOSVENDA 7
```

### 5.8.2.3 staAux

```
#define staAux 50
```

### 5.8.2.4 TAMCLIENTES

```
#define TAMCLIENTES 20000
```

### 5.8.2.5 TAMPROD

```
#define TAMPROD 200000
```

### 5.8.2.6 TAMVENDAS

```
#define TAMVENDAS 1000000
```

## 5.8.3 Function Documentation

### 5.8.3.1 clienttoArray()

```
void clienttoArray (  
    char * fich )
```

Função que lê os Cleintes do ficheiro e os poe num array de strings. Tambem faz a validação usando a função validclient.

**Parameters**

<i>fich</i>	
-------------	--

**5.8.3.2 escreveArray()**

```
void escreveArray (
    FILE * fp,
    char * array[ ] )
```

Função que dado um apontador para um ficheiro e um array de "strings" (vaziu), preenche o array com o que esta no ficheiro.

**Parameters**

<i>fp</i>	
<i>array</i>	

**5.8.3.3 fazStruct()**

```
int fazStruct (
    char * linhaVendaOk )
```

Função que escreve cada linha do array de vendas num array de struct; também verifica se cada elemento da venda é valido.

**Parameters**

<i>linhaVendaOk</i>	- uma linha do array de vendas.
---------------------	---------------------------------

**Returns**

int

**5.8.3.4 prodtoArray()**

```
void prodtoArray (
    char * fich )
```

**Parameters**

<i>fich</i>	
-------------	--

#### 5.8.3.5 validclient()

```
void validclient (
    char clientes [ ] )
```

Função que valida Clientes antes de os inserir num array.

##### Parameters

<i>clientes</i>	
-----------------	--

#### 5.8.3.6 validProd()

```
void validProd (
    char produtos [ ] )
```

Função que valida Produtos antes de os inserir num array.

##### Parameters

<i>produtos</i>	
-----------------	--

#### 5.8.3.7 validvendas()

```
void validvendas (
    char * fich )
```

Função que lê as vendas do ficheiro e as põe num array de strings. Também faz a validação.

##### Parameters

<i>fich</i>	
-------------	--

#### 5.8.3.8 verclien()

```
int verclien (
    char * campos )
```

Função que verifica se o Cliente é válido.

**Parameters**

<i>campos</i>	
---------------	--

**Returns**

int

**5.8.3.9 verfilial()**

```
int verfilial (
    int filial )
```

Função que verifica se a filial é válida.

**Parameters**

<i>filial</i>	
---------------	--

**Returns**

int

**5.8.3.10 vermes()**

```
int vermes (
    int mes )
```

Função que verifica se o mês da compra é válido.

**Parameters**

<i>mes</i>	
------------	--

**Returns**

int

**5.8.3.11 verprod()**

```
int verprod (
    char * campos )
```

Função que verifica se o Produto é válido. Procura a venda dada como input no array de Produtos validos.

## Parameters

<i>campos</i>	
---------------	--

## Returns

int

## 5.8.3.12 vertcompra()

```
int vertcompra (
    char * compra )
```

Função que verifica se o tipo de compra é válido.

## Parameters

<i>compra</i>	
---------------	--

## Returns

int

## 5.8.3.13 verunidadesvend()

```
int verunidadesvend (
    int unidades )
```

Função que verifica se o número de unidades compradas é válido.

## Parameters

<i>unidades</i>	
-----------------	--

## Returns

int

## 5.8.3.14 verunidec()

```
int verunidec (
    double unidec )
```

## 5.8.4 Variable Documentation

### 5.8.4.1 clientes

```
char* clientes[TAMCLIENTES]
```

### 5.8.4.2 produtos

```
char* produtos[TAMPROD]
```

### 5.8.4.3 venda

```
char* venda[TAMVENDAS]
```



# Index

## `_GNU_SOURCE`

- `init.c`, 11
- `init.h`, 13
- `main.c`, 14
- `queries.c`, 15
- `queries.h`, 17
- `valida.c`, 20
- `valida.h`, 28

## `arrayprod`

- `valida.c`, 26

## `CAMPOSVENDA`

- `valida.c`, 20
- `valida.h`, 29

## `cliente`

- `vendas`, 8

## `clientes`

- `valida.c`, 26
- `valida.h`, 34

## `clienttoArray`

- `valida.c`, 21
- `valida.h`, 29

## `escreveArray`

- `valida.c`, 21
- `valida.h`, 30

## `fazStruct`

- `valida.c`, 22
- `valida.h`, 30

## `filial`

- `vendas`, 8

## `imprime_ultimo`

- `queries.c`, 15
- `queries.h`, 17

## `init.c`, 11

- `_GNU_SOURCE`, 11
- `initt`, 12

## `init.h`, 12

- `_GNU_SOURCE`, 13
- `initt`, 13

## `initArrayTree`

- `valida.c`, 22

## `initt`

- `init.c`, 12
- `init.h`, 13

## `linha_mais_longa`

- `queries.c`, 15

- `queries.h`, 17

## `main`

- `main.c`, 14

## `main.c`, 13

- `_GNU_SOURCE`, 14
- `main`, 14

## `mes`

- `vendas`, 8

## `preco`

- `vendas`, 8

## `precototal`

- `query`, 7

## `printelemnts`

- `valida.c`, 22

## `prod`

- `vendas`, 8

## `prodtoArray`

- `valida.c`, 22
- `valida.h`, 30

## `prodToTree`

- `valida.c`, 23

## `produtos`

- `valida.c`, 26
- `valida.h`, 34

## `queries.c`, 14

- `_GNU_SOURCE`, 15
- `imprime_ultimo`, 15
- `linha_mais_longa`, 15
- `testa_brp`, 16

## `queries.h`, 16

- `_GNU_SOURCE`, 17
- `imprime_ultimo`, 17
- `linha_mais_longa`, 17
- `testa_brp`, 17

## `Query`

- `valida.c`, 21

## `query`, 7

- `precototal`, 7
- `unidcompradas`, 7

## `README.md`, 18

## `staAux`

- `valida.c`, 20
- `valida.h`, 29

## `TAMCLIENTES`

- `valida.c`, 20

- valida.h, 29
- TAMPROD
  - valida.c, 20
  - valida.h, 29
- TAMVENDAS
  - valida.c, 20
  - valida.h, 29
- tcompra
  - vendas, 9
- testa\_brp
  - queries.c, 16
  - queries.h, 17
- teste
  - valida.c, 26
- unidades
  - vendas, 9
- unidcompradas
  - query, 7
- valida.c, 18
  - \_GNU\_SOURCE, 20
  - arrayprod, 26
  - CAMPOSVENDA, 20
  - clientes, 26
  - clienttoArray, 21
  - escreveArray, 21
  - fazStruct, 22
  - initArrayTree, 22
  - printelements, 22
  - prodtoArray, 22
  - prodToTree, 23
  - produtos, 26
  - Query, 21
  - staAux, 20
  - TAMCLIENTES, 20
  - TAMPROD, 20
  - TAMVENDAS, 20
  - teste, 26
  - validadas, 27
  - validclient, 23
  - validProd, 23
  - validvendas, 23
  - ven, 27
  - venda, 27
  - Vendas, 21
  - verclien, 24
  - verfilial, 24
  - vermes, 24
  - verpreco, 25
  - verprod, 25
  - vertcompra, 25
  - verunidadesvend, 26
- valida.h, 27
  - \_GNU\_SOURCE, 28
  - CAMPOSVENDA, 29
  - clientes, 34
  - clienttoArray, 29
  - escreveArray, 30
  - fazStruct, 30
  - prodtoArray, 30
  - produtos, 34
  - staAux, 29
  - TAMCLIENTES, 29
  - TAMPROD, 29
  - TAMVENDAS, 29
  - validclient, 31
  - validProd, 31
  - validvendas, 31
  - venda, 34
  - verclien, 31
  - verfilial, 32
  - vermes, 32
  - verprod, 32
  - vertcompra, 33
  - verunidadesvend, 33
  - verunidec, 33
- validadas
  - valida.c, 27
- validclient
  - valida.c, 23
  - valida.h, 31
- validProd
  - valida.c, 23
  - valida.h, 31
- validvendas
  - valida.c, 23
  - valida.h, 31
- ven
  - valida.c, 27
- venda
  - valida.c, 27
  - valida.h, 34
- Vendas
  - valida.c, 21
- vendas, 8
  - cliente, 8
  - filial, 8
  - mes, 8
  - preco, 8
  - prod, 8
  - tcompra, 9
  - unidades, 9
- verclien
  - valida.c, 24
  - valida.h, 31
- verfilial
  - valida.c, 24
  - valida.h, 32
- vermes
  - valida.c, 24
  - valida.h, 32
- verpreco
  - valida.c, 25
- verprod
  - valida.c, 25
  - valida.h, 32

vertcompra  
    valida.c, [25](#)  
    valida.h, [33](#)  
verunidadesvend  
    valida.c, [26](#)  
    valida.h, [33](#)  
verunidec  
    valida.h, [33](#)