

# Calculo Manual de referência

1

Gerado por Doxygen 1.2.6

Mon Apr 15 15:57:08 2002



---

# Conteúdo

<b>1</b>	<b>Calculo Índice dos namespaces</b>	<b>1</b>
1.1	Calculo Lista de namespaces . . . . .	1
<b>2</b>	<b>Calculo Índice da hierarquia</b>	<b>3</b>
2.1	Calculo Hierarquia de classes . . . . .	3
<b>3</b>	<b>Calculo Índice dos componentes</b>	<b>5</b>
3.1	Calculo Lista de componentes . . . . .	5
<b>4</b>	<b>Calculo Índice dos ficheiros</b>	<b>7</b>
4.1	Calculo Lista de ficheiros . . . . .	7
<b>5</b>	<b>Calculo Documentação dos namespaces</b>	<b>9</b>
5.1	Referencia ao namespace Calculo . . . . .	9
5.2	Referencia ao namespace std . . . . .	10
<b>6</b>	<b>Calculo Documentação da classe</b>	<b>11</b>
6.1	Referencia à classe Erros::Erro . . . . .	11
6.2	Referencia à classe Calculo::ErroDeAnalise . . . . .	12
6.3	Referencia à classe Calculo::ErroDeCalculo . . . . .	13
6.4	Referencia à classe Calculo::Formula . . . . .	14
6.5	Referencia à classe Calculo::Funcao . . . . .	17
6.6	Referencia à classe Calculo::Simbolo . . . . .	19
<b>7</b>	<b>Calculo Documentação do ficheiro</b>	<b>23</b>
7.1	Referencia ao ficheiro erros.H . . . . .	23
7.2	Referencia ao ficheiro erros_impl.H . . . . .	24
7.3	Referencia ao ficheiro formula.C . . . . .	25
7.4	Referencia ao ficheiro formula.H . . . . .	26
7.5	Referencia ao ficheiro formula_impl.H . . . . .	27

---

7.6	Referencia ao ficheiro funcao.C . . . . .	28
7.7	Referencia ao ficheiro funcao.H . . . . .	29
7.8	Referencia ao ficheiro funcao_impl.H . . . . .	30
7.9	Referencia ao ficheiro simbolo.C . . . . .	31
7.10	Referencia ao ficheiro simbolo.H . . . . .	32
7.11	Referencia ao ficheiro simbolo_impl.H . . . . .	33
7.12	Referencia ao ficheiro teste.C . . . . .	34

# Capítulo 1

## Calculo Índice dos namespaces

### 1.1 Calculo Lista de namespaces

Lista dos namespaces com uma breve descrição:

<b>Calculo</b>	.....	9
<b>std</b>	.....	10



---

# Capítulo 2

## Calculo Índice da hierarquia

### 2.1 Calculo Hierarquia de classes

Esta lista de heranças está organizada, dentro do possível, por ordem alfabética:

Erros::Erro . . . . .	11
Calculo::ErroDeAnalise . . . . .	12
Calculo::ErroDeCalculo . . . . .	13
Calculo::Formula . . . . .	14
Calculo::Funcao . . . . .	17
Calculo::Simbolo . . . . .	19





---

## Capítulo 3

# Calculo Índice dos componentes

### 3.1 Calculo Lista de componentes

Lista da classes, estruturas, uniões e interfaces com uma breve descrição:

<b>Erros::Erro</b> . . . . .	11
<b>Calculo::ErroDeAnalise</b> (Representa excepções ocorridas durante a análise de fórmulas) . . .	12
<b>Calculo::ErroDeCalculo</b> (Representa excepções ocorridas durante o cálculo de fórmulas) . . .	13
<b>Calculo::Formula</b> (Representa uma fórmula algébrica) . . . . .	14
<b>Calculo::Funcao</b> (Representa funções definidas pelo utilizador à custa de fórmulas) . . . . .	17
<b>Calculo::Simbolo</b> (Representa um símbolo da gramática usada nas fórmulas) . . . . .	19



---

## Capítulo 4

# Calculo Índice dos ficheiros

### 4.1 Calculo Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

<b>erros.H</b>	23
<b>erros_impl.H</b>	24
<b>formula.C</b>	25
<b>formula.H</b>	26
<b>formula_impl.H</b>	27
<b>funcao.C</b>	28
<b>funcao.H</b>	29
<b>funcao_impl.H</b>	30
<b>simbolo.C</b>	31
<b>simbolo.H</b>	32
<b>simbolo_impl.H</b>	33
<b>teste.C</b>	34



---

## Capítulo 5

# Calculo Documentação dos namespaces

### 5.1 Referencia ao namespace Calculo

#### Componentes

- struct **Calculo::Formula::Contexto**  
*Representa de forma compacta o contexto de um cálculo, i.e., guarda informação acerca das constantes e funções definidas pelo utilizador.*
  - class **Calculo::ErroDeAnalise**  
*Representa exceções ocorridas durante a análise de fórmulas.*
  - class **Calculo::ErroDeCalculo**  
*Representa exceções ocorridas durante o cálculo de fórmulas.*
  - class **Calculo::Formula**  
*Representa uma fórmula algébrica.*
  - class **Calculo::Funcao**  
*Representa funções definidas pelo utilizador à custa de fórmulas.*
  - class **Calculo::Simbolo**  
*Representa um símbolo da gramática usada nas fórmulas.*
-

## 5.2 Referencia ao namespace std

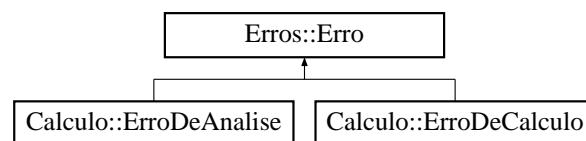
---

## Capítulo 6

# Calculo Documentação da classe

### 6.1 Referencia à classe Erros::Erro

Diagrama de heranças da classe Erros::Erro:



A documentação para esta classe foi gerada a partir do seguinte ficheiro:

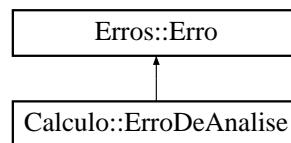
- [erros.H](#)

## 6.2 Referencia à classe Calculo::ErroDeAnalise

Representa exceções ocorridas durante a análise de fórmulas.

```
#include <erros.H>
```

Diagrama de heranças da classe Calculo::ErroDeAnalise:



### Membros públicos

- [ErroDeAnalise](#) (std::string const &mensagem)

*Constrói novo erro com mensagem de erro dada.*

### 6.2.1 Descrição detalhada

Representa exceções ocorridas durante a análise de fórmulas.

Definido na linha 9 do ficheiro erros.H.

### 6.2.2 Documentação do Construtor & Destrutor

#### 6.2.2.1 Calculo::ErroDeAnalise::ErroDeAnalise (std::string const & *mensagem*) [inline]

Constrói novo erro com mensagem de erro dada.

#### Parâmetros:

***mensagem*** Uma mensagem de erro.

Definido na linha 1 do ficheiro erros\_impl.H.

A documentação para esta classe foi gerada a partir dos seguintes ficheiros:

- [erros.H](#)
- [erros\\_impl.H](#)

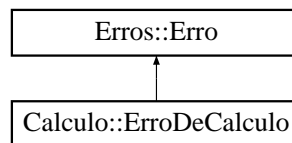


## 6.3 Referencia à classe `Calculo::ErroDeCalculo`

Representa exceções ocorridas durante o cálculo de fórmulas.

```
#include <erros.H>
```

Diagrama de heranças da classe `Calculo::ErroDeCalculo`:



### Membros públicos

- [ErroDeCalculo](#) (`std::string const &mensagem`)  
*Constrói novo erro com mensagem de erro dada.*

#### 6.3.1 Descrição detalhada

Representa exceções ocorridas durante o cálculo de fórmulas.

Definido na linha 18 do ficheiro `erros.H`.

#### 6.3.2 Documentação do Construtor & Destrutor

##### 6.3.2.1 `Calculo::ErroDeCalculo::ErroDeCalculo (std::string const & mensagem) [inline]`

Constrói novo erro com mensagem de erro dada.

##### Parâmetros:

*mensagem* Uma mensagem de erro.

Definido na linha 5 do ficheiro `erros_impl.H`.

A documentação para esta classe foi gerada a partir dos seguintes ficheiros:

- [erros.H](#)
- [erros\\_impl.H](#)

## 6.4 Referencia à classe Calculo::Formula

Representa uma fórmula algébrica.

```
#include <formula.H>
```

### Tipos Públicos

- typedef std::map<std::string, double> [MapaDeConstantes](#)  
*Tipo que representa mapas de constantes, onde a chave é o nome da constante e o valor é o seu valor:.*
- typedef std::map<std::string, [Funcao](#)> [MapaDeFuncoes](#)  
*Tipo que representa mapas de funções definidas pelo utilizador, i.e., funções definidas à custa da própria classe [Formula](#), onde a chave é o nome da função e o valor é uma instância da classe [Funcao](#):.:*

### Membros públicos

- [Formula](#) (std::string const &formula)  
*Constrói uma fórmula dada a sua representação na forma textual.*
- double [valor](#) ([MapaDeConstantes](#) const &constantes, [MapaDeFuncoes](#) const &funcoes) const  
*Calcula o valor de uma fórmula no contexto dado pelos mapas de constantes e funções passados como argumentos.*

#### 6.4.1 Descrição detalhada

Representa uma fórmula algébrica.

Uma fórmula algébrica corresponde a uma expressão recorrendo a valores literais, constantes e funções, com a seguinte gramática (regras apresentadas por ordem crescente de precedência):

Gramática das fórmulas:

expressao := expressao\_condicional

expressao\_condicional := expressao\_ou | expressao\_ou '?' expressao\_condicional ':' expressao\_condicional

expressao\_ou := expressao\_e | expressao\_e '|' expressao\_ou

expressao\_e := expressao\_igualdade | expressao\_igualdade '&' expressao\_e

expressao\_igualdade := expressao\_relacional | expressao\_relacional '==' expressao\_igualdade expressao\_relacional '!=' expressao\_igualdade

expressao\_relacional := expressao\_aditiva | expressao\_aditiva '<' expressao\_relacional expressao\_aditiva '<=' expressao\_relacional expressao\_aditiva '>' expressao\_relacional expressao\_aditiva '>=' expressao\_relacional

expressao\_aditiva := expressao\_multiplicativa | expressao\_multiplicativa '+' expressao\_aditiva expressao\_multiplicativa '-' expressao\_aditiva

`expressao_multiplicativa := expressao_potencial | expressao_potencial '*' expressao_multiplicativa expressao_potencial '/' expressao_multiplicativa`

`expressao_potencial := expressao_unaria | expressao_unaria '^' expressao_potencial`

`expressao_unaria := expressao_funcional | '-' expressao_unaria '+' expressao_unaria '!' expressao_unaria`

`expressao_funcional := expressao_primaria | identificador '(' ')' identificador '(' lista_de_expressoes ')'`

`lista_de_expressoes := expressao | expressao ',' lista_de_expressoes`

`expressao_primaria := numero | constante | '(' expressao ')'`

`numero := como os double`

`constante := nome, com nome começando por letra (sequência de letras, dígitos, e ..`

`identificador := sequência de letras, dígitos, e _ começando por letra`

#### Invariante:

`true.`

Definido na linha 94 do ficheiro `formula.H`.

## 6.4.2 Documentação de tipos definidos

### 6.4.2.1 `typedef std::map< std::string, double > Calculo::Formula::MapaDeConstantes`

Tipo que representa mapas de constantes, onde a chave é o nome da constante e o valor é o seu valor..

Definido na linha 98 do ficheiro `formula.H`.

### 6.4.2.2 `typedef std::map< std::string, Funcao > Calculo::Formula::MapaDeFuncoes`

Tipo que representa mapas de funções definidas pelo utilizador, i.e., funções definidas à custa da própria classe `Formula`, onde a chave é o nome da função e o valor é uma instância da classe `Funcao`..

Definido na linha 104 do ficheiro `formula.H`.

## 6.4.3 Documentação do Construtor & Destrutor

### 6.4.3.1 `Calculo::Formula::Formula (std::string const & formula) [inline]`

Constrói uma fórmula dada a sua representação na forma textual.

#### Excepções:

*`Calculo::ErroDeAnalise`* erro durante a análise da fórmula.

Definido na linha 10 do ficheiro `formula_impl.H`.

## 6.4.4 Documentação dos métodos

### 6.4.4.1 `double Calculo::Formula::valor (MapaDeConstantes const & constantes, MapaDeFuncoes const & funcoes) const`

Calcula o valor de uma fórmula no contexto dado pelos mapas de constantes e funções passados como argumentos.

**Parâmetros:**

*constantes* Mapa com os valores das constantes.

*funcoes* Mapa com as funções definidas pelo utilizador.

**Excepções:**

*Calculo::ErroDeAnalise* erro durante a análise da fórmula.

A documentação para esta classe foi gerada a partir dos seguintes ficheiros:

- [formula.H](#)
- [formula\\_impl.H](#)

## 6.5 Referencia à classe `Calculo::Funcao`

Representa funções definidas pelo utilizador à custa de fórmulas.

```
#include <funcao.H>
```

### Tipos Públicos

- `typedef std::map<std::string, double>` [MapaDeConstantes](#)  
*Tipo que representa mapas de constantes, onde a chave é o nome da constante e o valor é o seu valor.*
- `typedef std::map<std::string, Funcao>` [MapaDeFuncoes](#)  
*Tipo que representa mapas de funções definidas pelo utilizador, i.e., funções definidas à custa da própria classe [Formula](#), onde a chave é o nome da função e o valor é uma instância da classe [Funcao](#).*
- `typedef std::vector<std::string>` [Parametros](#)  
*Tipo que representa os parâmetros de uma função.*

### Membros públicos

- [Funcao](#) ([Parametros](#) const &parametros=[Parametros](#)(), std::string const &formula="")  
*Constrói uma nova função correspondente à fórmula dada com os parâmetros indicados.*
- `double operator()` ([MapaDeConstantes](#) const &constantes, [MapaDeFuncoes](#) const &funcoes, std::vector< double > const &argumentos) const  
*Devolve a valor da função no contexto dado pelo mapa de constantes |constantes| e pelo mapa de funções |funcoes|, usando |argumentos| como os valores a usar para cada parâmetro.*

#### 6.5.1 Descrição detalhada

Representa funções definidas pelo utilizador à custa de fórmulas.

Uma função corresponde a uma fórmula com os parâmetros representados por constantes que são automaticamente injectadas durante o seu cálculo.

**Veja também:**

[Formula](#).

Definido na linha 17 do ficheiro `funcao.H`.

#### 6.5.2 Documentação de tipos definidos

#### 6.5.2.1 `typedef std::map< std::string, double > Calculo::Funcao::MapaDeConstantes`

Tipo que representa mapas de constantes, onde a chave é o nome da constante e o valor é o seu valor:.

Definido na linha 21 do ficheiro funcao.H.

#### 6.5.2.2 `typedef std::map< std::string, Funcao > Calculo::Funcao::MapaDeFuncoes`

Tipo que representa mapas de funções definidas pelo utilizador, i.e., funções definidas à custa da própria classe [Formula](#), onde a chave é o nome da função e o valor é uma instância da classe [Funcao](#)..

Definido na linha 27 do ficheiro funcao.H.

#### 6.5.2.3 `typedef std::vector< std::string > Calculo::Funcao::Parametros`

Tipo que representa os parâmetros de uma função.

Definido na linha 30 do ficheiro funcao.H.

### 6.5.3 Documentação do Construtor & Destrutor

#### 6.5.3.1 `Calculo::Funcao::Funcao (Parametros const & parametros = Parametros(), std::string const & formula = "") [inline]`

Constrói uma nova função correspondente à fórmula dada com os parâmetros indicados.

Definido na linha 1 do ficheiro funcao\_impl.H.

### 6.5.4 Documentação dos métodos

#### 6.5.4.1 `double Calculo::Funcao::operator() (MapaDeConstantes const & constantes, MapaDeFuncoes const & funcoes, std::vector< double > const & argumentos) const`

Devolve a valor da função no contexto dado pelo mapa de constantes |constantes| e pelo mapa de funções |funcoes|, usando |argmumentos| como os valores a usar para cada parâmetro.

A documentação para esta classe foi gerada a partir dos seguintes ficheiros:

- [funcao.H](#)
- [funcao\\_impl.H](#)

## 6.6 Referencia à classe Calculo::Simbolo

Representa um símbolo da gramática usada nas fórmulas.

```
#include <simbolo.H>
```

### Tipos Públicos

- enum **Tipo** { teste\_de\_condicao, primeiro = teste\_de\_condicao, separador\_de\_condicao, ou, e, negacao, igual, diferente, menor, menor\_ou\_igual, maior, maior\_ou\_igual, adicao, subtracao, produto, divisao, potencia, parenteses\_esquerdo, parenteses\_direito, virgula, identificador, numero, constante, fim, ultimo = fim }

*Representa os diversos tipos de símbolos existentes na gramática.*

### Membros públicos

- **Simbolo** (std::istream &entrada)

*Constrói um símbolo de forma gulosa (quantos mais caracteres melhor, lendo caracteres do canal |entrada|).*

- **operator Tipo** () const

*Conversor implícito para o tipo enumerado Tipo, permite simplificar muito as expressões envolvendo instâncias desta classe.*

- std::string const& **valorTextual** () const

*Devolve o valor textual correspondente ao símbolo.*

- double **valorNumerico** () const

*Devolve o valor numérico correspondente ao símbolo.*

- std::string const& **nomeCurto** () const

*Devolve o nome curto do símbolo (e.g., se o símbolo for |igual|, o nome curto é "=").*

- std::string const& **nomeLongo** () const

*Devolve o nome longo do símbolo (e.g., se o símbolo for |menor\_ou\_igual|, o nome longo é "menor ou igual").*

- std::string **nomeCompleto** () const

*Devolve o nome completo do símbolo (e.g., se o símbolo for |menor\_ou\_igual|, o nome longo é "'<=' (menor ou igual)").*

- std::string **descricao** () const

*Devolve a descrição completa do símbolo (e.g., se o símbolo for |constante|, com o nome "pi", a descrição é "'const' (constante) [valor = pi]").*

### 6.6.1 Descrição detalhada

Representa um símbolo da gramática usada nas fórmulas.

**Veja também:**

[Calculo::Formula](#)

**Invariante:**

true.

Definido na linha 15 do ficheiro simbolo.H.

### 6.6.2 Documentação da enumerações

#### 6.6.2.1 enum Calculo::Simbolo::Tipo

Representa os diversos tipos de símbolos existentes na gramática.

**Valores da enumeração:**

*teste\_de\_condicao* símbolo '?'.  
*primeiro*

*separador\_de\_condicao* Símbolo ':'.

*ou* Símbolo '|'.  
*e* Símbolo '&'.

*negacao* Símbolo '!'.  
*igual* Símbolo '=='.

*diferente* Símbolo '!='.  
*menor* Símbolo '<'.

*menor\_ou\_igual* Símbolo '<='.

*maior* Símbolo '>'.

*maior\_ou\_igual* Símbolo '>='.

*adicao* Símbolo '+'.  
*subtracao* Símbolo '-'.

*produto* Símbolo '\*'.  
*divisao* Símbolo '/'.  
*potencia* Símbolo '^'.

*parenteses\_esquerdo* Símbolo '('.

*parenteses\_direito* Símbolo ')'.  
*virgula* Símbolo ','.

*identificador* Um identificador (nome).

*numero* Um número em notação científica.

*constante* Uma constante representando um número.

*fim* Símbolo especial, marca o fim da análise.

*ultimo*

Definido na linha 18 do ficheiro simbolo.H.



### 6.6.3 Documentação do Construtor & Destrutor

#### 6.6.3.1 `Calculo::Simbolo::Simbolo (std::istream & entrada) [explicit]`

Constrói um símbolo de forma gulosa (quantos mais caracteres melhor, lendo caracteres do canal `|entrada|`).

### 6.6.4 Documentação dos métodos

#### 6.6.4.1 `std::string Calculo::Simbolo::descricao () const`

Devolve a descrição completa do símbolo (e.g., se o símbolo for `|constante|`, com o nome `"pi"`, a descrição é `""const' (constante) [valor = pi]"`).

#### 6.6.4.2 `std::string Calculo::Simbolo::nomeCompleto () const [inline]`

Devolve o nome completo do símbolo (e.g., se o símbolo for `|menor_ou_igual|`, o nome longo é `""<=' (menor ou igual)"`).

Definido na linha 42 do ficheiro `simbolo_impl.H`.

#### 6.6.4.3 `std::string const & Calculo::Simbolo::nomeCurto () const [inline]`

Devolve o nome curto do símbolo (e.g., se o símbolo for `|igual|`, o nome curto é `""='"`).

Definido na linha 28 do ficheiro `simbolo_impl.H`.

Referenciado por `nomeCompleto()`.

#### 6.6.4.4 `std::string const & Calculo::Simbolo::nomeLongo () const [inline]`

Devolve o nome longo do símbolo (e.g., se o símbolo for `|menor_ou_igual|`, o nome longo é `""menor ou igual"`).

Definido na linha 35 do ficheiro `simbolo_impl.H`.

Referenciado por `nomeCompleto()`.

#### 6.6.4.5 `Calculo::Simbolo::operator Tipo () const`

Conversor implícito para o tipo enumerado `Tipo`, permite simplificar muito as expressões envolvendo instâncias desta classe.

É um pouco como se esta classe funcionasse como uma versão sofisticada de um simples tipo enumerado. Se o canal não tiver nada disponível, o símbolo construído corresponde ao valor enumerado `|fim|`, usado para assinalar o fim de um processo de análise.

**Veja também:**

[Tipo](#)

#### **6.6.4.6 double Calculo::Simbolo::valorNumerico() const [inline]**

Devolve o valor numérico correspondente ao símbolo.

Só faz sentido invocar esta operação se o símbolo for do tipo |numero|.

**Precondição:**

\*this == numero.

Definido na linha 20 do ficheiro simbolo\_impl.H.

#### **6.6.4.7 std::string const & Calculo::Simbolo::valorTextual() const [inline]**

Devolve o valor textual correspondente ao símbolo.

Só faz sentido invocar esta operação se o símbolo for dos tipos |constante| ou |identificador|.

**Precondição:**

\*this == constante or \*this == constante.

Definido na linha 12 do ficheiro simbolo\_impl.H.

A documentação para esta classe foi gerada a partir dos seguintes ficheiros:

- [simbolo.H](#)
- [simbolo\\_impl.H](#)

---

## Capítulo 7

# Calculo Documentação do ficheiro

### 7.1 Referencia ao ficheiro erros.H

```
#include <Erros/erros.H>
#include "erros_impl.H"
```

#### Namespaces

- namespace [Calculo](#)

## 7.2 Referencia ao ficheiro erros\_impl.H

## 7.3 Referencia ao ficheiro formula.C

```
#include "formula.H"  
#include <cmath>  
#include <Extra/string.H>  
#include "erros.H"  
#include "funcao.H"
```

### Namespaces

- namespace [std](#)

## 7.4 Referencia ao ficheiro formula.H

```
#include <string>
#include <vector>
#include <map>
#include "simbolo.H"
#include "formula_impl.H"
```

### Namespaces

- namespace [Calculo](#)

## 7.5 Referencia ao ficheiro formula\_impl.H

```
#include <sstream>
```

## 7.6 Referencia ao ficheiro funcao.C

```
#include "funcao.H"  
#include "formula.H"  
#include "erros.H"  
#include <Extra/string.H>
```



## 7.7 Referencia ao ficheiro funcao.H

```
#include <vector>
#include <string>
#include <map>
#include "formula.H"
#include "funcao_impl.H"
```

### Namespaces

- namespace [Calculo](#)

## **7.8 Referencia ao ficheiro funcao\_impl.H**

## 7.9 Referencia ao ficheiro simbolo.C

```
#include "simbolo.H"  
#include <sstream>
```

## 7.10 Referencia ao ficheiro simbolo.H

```
#include <iosfwd>
#include <string>
#include "erros.H"
#include "simbolo_impl.H"
```

### Namespaces

- namespace [Calculo](#)

## 7.11 Referencia ao ficheiro simbolo\_impl.H

```
#include <iostream>
#include <cassert>
```

## 7.12 Referencia ao ficheiro teste.C

```
#include <cmath>
#include <map>
#include <string>
#include <vector>
#include <iostream>
#include "funcao.H"
#include "formula.H"
```

### Funções

- `int main ()`

*Programa de teste interactivo da classe Formula.*

### 7.12.1 Documentação da função

#### 7.12.1.1 `int main ()`

Programa de teste interactivo da classe Formula.

Lê fórmulas pedidas ao utilizador e calcula o seu valor.

Definido na linha 16 do ficheiro teste.C.