Contents

1

namespace **Slang**

Este espaço nominativo envolve todas as ferramentas da biblioteca Slang++.

Names

Este espaço nominativo envolve todas as ferramentas da biblioteca Slang++.

A biblioteca Slang++ permite a utilização de algumas funções que actuam sobre o teclado e o ecrã e a criação de menus simples em modo texto. A biblioteca consiste de um pacote `Slang` (representado pelo espaço nominativo `Slang`) dividido em quatro módulos fisicos `teclado`, `ecra`, `menu` e `util`, cada um com o correspondente ficheiro de interface (`Slang/teclado.H`, `Slang/ecra.H`, `Slang/menu.H` e `Slang/util.H`).

Estão definidas nesta biblioteca as variáveis globais `teclado` (do tipo `Slang::Teclado`) e `ecra` (do tipo `Slang::Ecra`), não sendo por isso necessário criar quaisquer variáveis destes tipos.

Nesta documentação omite-se o prefixo `Slang::` sempre que conveniente para facilitar a leitura, embora em rigor este seja necessário.

Para construir um programa (neste caso `teste_slang.C`) que utilize estas biblioteca deve dar o seguinte comando:

```
c++ -Wall -ansi -pedantic -g -o teste_slang teste_slang.C -lSlang++ -lslang
```

Os ficheiros fonte devem incluir o ficheiro de interface `Slang/slang.H`, que por sua vez inclui os ficheiros de interface dos quatro módulos, ou incluir apenas o ficheiro de interface do módulo pretendido.

Sempre que se testar algum programa que use a biblioteca Slang++ deve-se usar uma consola `xterm` (outras consolas podem gerar alguns problemas). Para lançar uma consola `xterm` deve-se dar o comando:

```
xterm&
```

numa consola normal (`Konsole`).

Pode-se também fazer 'alt-F2' e escrever `xterm` na caixa de diálogo que surge no ecrã.

Caso se pretenda instalar esta biblioteca noutro computador deve-se fazer o seguinte:

1. Importar os seguintes ficheiros:

   - slang-1.4.0-2.i386.rpm (`http://www.iscte.pt/programacao/p2/trabalhos/slang-1.4.0-2.i38`
   - slang-devel-1.4.0-2.i386.rpm (`http://www.iscte.pt/programacao/p2/trabalhos/slang-devel-`
   - Slang++-0.1.tar.gz (`http://www.iscte.pt/programacao/p2/trabalhos/Slang++-0.1.tar.gz`)

   Para importar no netscape basta fazer 'shift-clique' no botão esquerdo do rato.

2. Dar os seguintes comandos:

```
tar -zxf Slang++-0.1.tar.gz
cd Slang++
make
su (inserir a senha [password] de administração da máquina)
cd ..
rpm -Uhv slang.*.rpm
exit
```

1.1 class **Coordenada**

Esta classe serve para representar as coordenadas de uma célula do ecrã.

Public Members

Esta classe serve para representar as coordenadas de uma célula do ecrã. Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.1.1

Construtores

Names

1.1.1.1

explicit **Coordenada** (int linha = 0, int coluna = 0)

Construtor da classe.

Construtor da classe. Por omissão a coordenada corresponde ao canto superior esquerdo do ecrã.

Parameters:

`linha`	Número da linha do ecrã onde se encontra a célula.
`coluna`	Número da coluna do ecrã onde se encontra a célula.

1.1.1.2

explicit **Coordenada** (std::istream& entrada)

Assume-se que o canal está ligado a um ficheiro cujo conteúdo foi criado usando o método `guarda()`.

Construtor da classe por carregamento a partir de um canal.

Parameters:

`entrada`	O canal de onde se carrega a coordenada.

Exceptions:

`ErroAoCarregar`	é lançada se o carregamento falhar.

1.1.2 Inspectores

Names

int	**linha** () const	*Devolve a linha do ecrã correspondente à coordenada.*
int	**coluna** () const	*Devolve a coluna do ecrã correspondente à coordenada.*

1.1.3 Modificadores

Names

void	**linha** (int linha)	*Modifica a linha do ecrã correspondente à coordenada.*
void	**coluna** (int coluna)	*Modifica a coluna do ecrã correspondente à coordenada.*

1.1.4 Serializadores

Names

1.1.4.1

void **carrega** (std::istream& entrada)

Carrega a coordenada a partir de um canal.

Carrega a coordenada a partir de um canal. Assume-se que o canal está ligado a um ficheiro cujo conteúdo foi criado usando o método `guarda()`.

Parameters:	`entrada`	O canal de onde se carrega a coordenada.
Exceptions:	`ErroAoCarregar`	é lançada se o carregamento falhar.Nesse caso a coordenada fica num estado inválido.

1.1.4.2

void **guarda** (std::ostream& saida) const

Guarda a coordenada escrevendo num canal.

Guarda a coordenada escrevendo num canal. O formato produzido é compatível com o que o método `carrega()` espera.

Parameters:	`saida`	O canal de onde se guarda a coordenada.
Exceptions:	`ErroAoGuardar`	é lançada se a operação falhar.

1.1.5

Operadores

Names

1.1.5.1

Coordenada& **operator +=** (Coordenada const& c)

Operador para soma de uma coordenada interpretada como um vector.

Operador para soma de uma coordenada interpretada como um vector.

1.1.5.2

Coordenada& **operator -=** (Coordenada const& c)

Operador para subtracção de uma coordenada interpretada como um vector.

Operador para subtracção de uma coordenada interpretada como um vector.

1.2

Operadores associados à classe `Coordenada`

Names

Definidos no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.2.1

Coordenada **operator +** (Coordenada a, Coordenada const& b)

Soma de duas coordenadas (o segundo operando é interpretado como um vector).

Soma de duas coordenadas (o segundo operando é interpretado como um vector).

1.2.2

Coordenada **operator -** (Coordenada a, Coordenada const& b)

Diferença de duas coordenadas (o resultado pode ser interpretado como um vector).

Diferença de duas coordenadas (o resultado pode ser interpretado como um vector).

1.2.3

Coordenada **operator +** (Coordenada a, Dimensao const& b)

Soma de uma coordenada com uma dimensão.

Soma de uma coordenada com uma dimensão.

1.2.4

Coordenada **operator -** (Coordenada a, Dimensao const& b)

Subtracção de uma dimensão a uma coordenada.

Subtracção de uma dimensão a uma coordenada.

1.3

class **Dimensao**

Esta classe serve para representar uma dimensão no ecrã.

Public Members

Esta classe serve para representar uma dimensão no ecrã. É tipicamente usada para representar a dimensão de caixas. Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.3.1 Construtores

Names

1.3.1.1 explicit **Dimensao** (int linhas = 0, int colunas = 0)

Construtor da classe.

Construtor da classe. Por omissão a dimensão é nula.

Parameters:

`linhas`	Número da linhas do ecrã ocupadas.
`colunas`	Número da colunas do ecrã ocupadas.

1.3.1.2 explicit **Dimensao** (std::istream& entrada)

Assume-se que o canal está ligado a um ficheiro cujo conteúdo foi criado usando o método `guarda()`.

Construtor da classe por carregamento a partir de um canal.

Parameters:

`entrada`	O canal de onde se carrega a dimensão.

Exceptions:

`ErroAoCarregar`	é lançada se o carregamento falhar.

1.3.2 Inspectores

Names

int	**linhas** () const	*Devolve o número de linhas do ecrã ocupadas.*
int	**colunas** () const	*Devolve o número de colunas do ecrã ocupadas.*

1.3.3 Modificadores

Names

void	**linhas** (int linhas)	*Modifica o número de linhas do ecrã ocupadas.*
void	**colunas** (int colunas)	*Modifica o número de colunas do ecrã ocupadas.*

1.3.4 Serializadores

Names

1.3.4.1	void	**carrega** (std::istream& entrada) *Carrega a dimensão a partir de um canal.*	15
1.3.4.2	void	**guarda** (std::ostream& saida) const *Guarda a dimensão escrevendo num canal.*	15

1.3.4.1

void **carrega** (std::istream& entrada)

Carrega a dimensão a partir de um canal.

Carrega a dimensão a partir de um canal. Assume-se que o canal está ligado a um ficheiro cujo conteúdo foi criado usando o método `guarda()`.

Parameters: `entrada` O canal de onde se carrega a dimensão.
Exceptions: `ErroAoCarregar` é lançada se o carregamento falhar.Nesse caso a dimensão fica num estado inválido.

1.3.4.2

void **guarda** (std::ostream& saida) const

Guarda a dimensão escrevendo num canal.

Guarda a dimensão escrevendo num canal. O formato produzido é compatível com o que o método `carrega()` espera.

Parameters: `saida` O canal de onde se guarda a dimensão.
Exceptions: `ErroAoGuardar` é lançada se a operação falhar.

1.3.5

Operadores

Names

1.3.5.1

Dimensao& **operator +=** (Dimensao const& c)

Operador para soma de uma dimensão.

Operador para soma de uma dimensão.

1.3.5.2

Dimensao& **operator -=** (Dimensao const& c)

Operador para subtracção de uma dimensão.

Operador para subtracção de uma dimensão.

1.4

Operadores associados à classe `Dimensao`

Names

Dimensao **operator +** (Dimensao a, Dimensao const& b)
Soma de duas dimensões.

Dimensao **operator -** (Dimensao a, Dimensao const& b)
Subtracção de duas dimensões.

Definidos no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.5

class **Caixa**

Esta classe serve para representar caixas no ecrã.

Public Members

Esta classe serve para representar caixas no ecrã. Uma caixa é um conjunto de células formando um rectângulo de lados não oblíquos. Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.5.1

Construtores

Names

1.5.1.1

explicit **Caixa** (Coordenada const& origem = Coordenada(), Dimensao const& dimensao = Dimensao())

Construtor da classe.

Construtor da classe. Por omissão a caixa está vazia.

Parameters:	`origem`	Coordenada da célula no canto superior esquerdo dacaixa.
	`dimensao`	Dimensão da caixa.

1.5.1.2

explicit **Caixa** (Coordenada const& a, Coordenada const& b)

Construtor da classe.

Construtor da classe.

Parameters:	`origem`	Coordenada da célula no canto superior esquerdo dacaixa.
	`destino`	Coordenada da célula no canto inferior direito dacaixa.

1.5.1.3

explicit **Caixa** (std::istream& entrada)

Assume-se que o canal está ligado a um ficheiro cujo conteúdo foi criado usando o método `guarda()`.

Construtor da classe por carregamento a partir de um canal.

Parameters:	`entrada`	O canal de onde se carrega a caixa.
Exceptions:	`ErroAoCarregar`	é lançada se o carregamento falhar.

1.5.2

Inspectores

Names

Coordenada	**origem** () const	*Devolve a coordenada da célula no canto superior esquerdo da caixa.*
Coordenada	**destino** () const	*Devolve a coordenada da célula no canto inferior direito da caixa.*
Dimensao	**dimensao** () const	*Devolve a dimensão da caixa.*

1.5.3

bool **vazia** () const

Devolve `true` *se a caixa estiver vazia, ou seja, se tiver dimensão nula.*

Devolve `true` se a caixa estiver vazia, ou seja, se tiver dimensão nula.

1.5.4

Relações geométricas

Names

bool	**sobre** (Coordenada const& c) const	*Devolve* `true` *se a coordenada* `c` *estiver sobre a caixa (na borda*
bool	**sobreABorda** (Coordenada const& c) const	*Devolve* `true` *se a coordenada* `c` *estiver sobre a borda da caixa.*

1.5.5

Modificadores

Names

1.5.5.1

void **origem** (Coordenada const& origem)

Modifica a coordenada do canto superior esquerdo da caixa (mas a sua dimensão mantém-se!).

Modifica a coordenada do canto superior esquerdo da caixa (mas a sua dimensão mantém-se!).

1.5.5.2

void **destino** (Coordenada const& destino)

Modifica a coordenada do canto inferior direito da caixa (não modifica a coordenada do canto superior esquerdo, pelo que a sua dimensão altera-se!).

Modifica a coordenada do canto inferior direito da caixa (não modifica a coordenada do canto superior esquerdo, pelo que a sua dimensão altera-se!).

1.5.6 Operadores

Names

Caixa& **operator +=** (Coordenada const& c)
Desloca a caixa de `c`*, que é interpretado como um vector.*

Caixa& **operator +=** (Caixa const& c)
"Soma"com uma caixa (resulta a caixa envolvente da união).

Caixa& **operator *=** (Caixa const& c)
"Multiplica"por uma caixa (resulta a caixa de intersecção).

1.5.7 Serializadores

Names

1.5.7.1 void **carrega** (std::istream& entrada)

Carrega a caixa a partir de um canal.

Carrega a caixa a partir de um canal. Assume-se que o canal está ligado a um ficheiro cujo conteúdo foi criado usando o método `guarda()`.

Parameters: `entrada` O canal de onde se carrega a caixa.
Exceptions: `ErroAoCarregar` é lançada se o carregamento falhar.Nesse caso a caixa fica num estado inválido.

1.5.7.2

void **guarda** (std::ostream& saida) const

Guarda a caixa escrevendo num canal.

Guarda a caixa escrevendo num canal. O formato produzido é compatível com o que o método `carrega()` espera.

Parameters: `saida` O canal de onde se guarda a caixa.
Exceptions: `ErroAoGuardar` é lançada se a operação falhar.

1.6

Operadores associados à classe `Caixa`

Names

Caixa **operator +** (Caixa a, Coordenada const& c)
Soma de caixa com coordenada: devolve caixa deslocada.

Caixa **operator +** (Coordenada const& c, Caixa a)
Soma de caixa com coordenada: devolve caixa deslocada.

1.6.1 Caixa **operator +** (Caixa a, Caixa const& b)
"Soma"de duas caixas: devolve caixa envolvente das duas caixas. 23

1.6.2 Caixa **operator *** (Caixa a, Caixa const& b)
"Produto"de duas caixas: devolve intersecção das duas caixas. 23

Definidos no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.6.1

Caixa **operator +** (Caixa a, Caixa const& b)

"Soma"de duas caixas: devolve caixa envolvente das duas caixas.

"Soma"de duas caixas: devolve caixa envolvente das duas caixas.

1.6.2

Caixa **operator *** (Caixa a, Caixa const& b)

"Produto"de duas caixas: devolve intersecção das duas caixas.

"Produto"de duas caixas: devolve intersecção das duas caixas.

1.7

Constantes associadas à classe `Caixa`

Names

Caixa const**caixa_vazia**	*Constante representando uma caixa vazia.*
Caixa const**caixa_universal**	*Constante representando a caixa universal (enche todo o espaço).*

Definidas no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.8

enum **Cor**

Este tipo enumerado representa as possíveis cores do texto e do fundo de cada célula do ecrã.

Members

preto	*Nome num canal:* `preto`.
cinza	*Nome num canal:* `cinza`.
vermelho	*Nome num canal:* `vermelho`.
vermelho_brilhante	

Nome num canal: `vermelho-brilhante`.

verde *Nome num canal:* `verde`.

verde_brilhante *Nome num canal:* `verde-brilhante`.

castanho *Nome num canal:* `castanho`.

amarelo *Nome num canal:* `amarelo`.

azul *Nome num canal:* `azul`.

azul_brilhante *Nome num canal:* `azul-brilhante`.

magenta *Nome num canal:* `magenta`.

magenta_brilhante *Nome num canal:* `magenta-brilhante`.

ciano *Nome num canal:* `ciano`.

ciano_brilhante *Nome num canal:* `ciano-brilhante`.

cinzento_claro *Nome num canal:* `cinzento-claro`.

branco *Nome num canal:* `branco`.

Este tipo enumerado representa as possíveis cores do texto e do fundo de cada célula do ecrã. Note-se que nem todas as cores são válidas para o fundo em todos os terminais! Experimente para verificar quais é que funcionam bem... Ver na documentação abaixo os nomes das cores usados para inserção e extração de canais. Definido no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.9 Operadores associados ao tipo enumerado `Cor`

Names

std::ostream&
operator << (std::ostream& saida, Cor cor)
Operador de inserção de cores num canal.

std::istream&

operator >> (std::istream& entrada, Cor& cor)
Operador de extracção de cores de um canal.

Definidos no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.10

int const **numero_de_cores**

Constante que guarda o número total de cores.

Constante que guarda o número total de cores. Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.11

class **Ecra**

Esta classe é um solitão: a sua única instância representa o ecrã.

Public Members

Esta classe é um solitão: a sua única instância representa o ecrã. Note-se que o ecrã é virtual. As alterações nele feitas só têm efeito no ecrã real depois de uma operação de refrescamento! Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

O ecrã permite várias operações:

- inspeccionar a dimensão do ecrã e a posição do cursor;
- verificar se o ecrã foi redimensionado;
- modificar a posição do cursor (de várias formas diferentes);
- copiar (*copy*) ou colar (*paste*) um troço de/para uma posição do ecrã;
- apagar várias partes do ecrã;
- refrescar o ecrã;
- desenhar uma caixa no ecrã;
- enviar para o ecrã um caractere, um inteiro, uma cadeia de caracteres, uma caixa, um troço ou um dos manipuladores definidos;
- etc.

Exemplo

```
// Cria uma caixa de dimensão 10x10 com origem na posição (0,0):
Caixa caixa(Coordenada(0, 0), Dimensao(10, 10));

// Copia o troço de ecrã que se encontra sob a caixa:
Ecra::Troco troco = ecra.copia(caixa);

// Cria um novo ObjectoCor:
Ecra::ObjectoCor cor(vermelho, verde);
// Move o cursor para a posição (30,30), cola o troço copiado acima
// nessa posição, muda a cor activa e refresca o ecrã de modo a que
// estas modificações se tornem visíveis:
        ecra << cursor(30, 30) << troco << cor  << refresca;
```

Atenção

As alterações só são visíveis depois de chamada uma das funções de refrescamento do ecrã. Quando há um redimensionamento do ecrã deve ser chamado o procedimento `refrescaTudo()`. Quando são feitas pequenas alterações deve ser chamado o procedimento `refresca()`. Também podem ser usados os manipuladores `refresca` e `refresca_tudo` para obter o mesmo efeito.

Exemplo de utilização

Este programa escreve no ecrã todas as letras do alfabeto (maiúsculas e minúsculas) e depois, a cada pressão de uma tecla, escreve os números de zero a nove com outra cor e com largura proporcional ao número escrito.

```
#include <Slang/slang.H>

using namespace Slang;

extern "C" {
#include <unistd.h>  // para sleep()
}

int main ()
{
    for(char c = 'a'; c != 'z' + 1; ++c) {
ecra << c << refresca;
sleep(1);
    }

    teclado.tecla();

    Ecra::ObjectoCor cor(amarelo, azul);

    ecra << cor << cursor(0, 0);
    for (int c = 0; c != 10; ++c) {
ecra.baixo();
ecra << parado << largura(c) << char('0' + c) << refresca;
teclado.tecla();
    }
}
```

1.11.1

class **ObjectoCor**

Esta classe é usada para indicar a cor das células do ecrã (as células têm uma cor de texto e outra de fundo).

Public Members

Esta classe é usada para indicar a cor das células do ecrã (as células têm uma cor de texto e outra de fundo). O sistema usado pela biblioteca C slang (base desta biblioteca C++ Slang++) é complicado pelo facto de se basear no conceito de palete. A ideia é que se deve usar um objecto cor para reservar uma posição da palete. Todas as células escritas usando este objecto cor guardarão na realidade uma referência para o respectivo objecto cor. Assim, se se alterar a cor do texto ou do fundo desse objecto cor, mudará a cor de todas as respectivas células! Note-se que, devido a limitações da biblioteca de base que está a ser usada (C slang), número de objectos cor definidos em cada momento não deverá ultrapassar os 255! Se isso acontecer as cores deixam de poder ser alteradas independentemente, com resultados muito estranhos... Note-se ainda que nem todas as cores são utilizáveis como cor de fundo em todas as consolas.

O ecrã tem sempre uma cor activa (nem que seja a cor do fundo do ecrã que por omissão é de texto branco sobre fundo preto).

Pode-se mudar a cor activa de várias maneiras, por exemplo usando o operador `Ecra& operator << (ObjectoCor const& objecto_cor)` do seguinte modo:

```
Ecra::ObjectoCor cor(vermelho / * texto * /, verde / * fundo * /);
ecra << cor;
```

A partir desta instrução tudo que fôr escrito passará a ser em vermelho sobre verde até que a cor seja mudada de novo. Note-se que é recomendável que a variável `cor` pressista enquanto existirem no ecrã células desenhadas com recurso a essa cor.

No entanto, após uma colagem de um troço de ecrã copiado previamente a cor activa volta a ser a do fundo do ecrã.

A escolha das cores utilizadas num programa deve ter sempre como objectivo tornar tão clara quanto possível a utilização do programa. Não se deve usar todas as cores disponíveis apenas "porque estão lá". Deve-se também também ter o cuidado de escolher cores de fundo e de texto que sejam minimamente contrastantes de modo a que o utilizador perceba bem o que está no ecrã.

1.11.1.1

ObjectoCor (Cor texto, Cor fundo)

Construtor da classe.

Construtor da classe.

Parameters:

`texto`	Cor inicial do texto.
`fundo`	Cor inicial do fundo.

1.11.1.2

Inspectores

Names

Cor	**texto** () const	*Devolve a cor do texto corrente deste objecto.*
Cor	**fundo** () const	*Devolve a cor do fundo corrente deste objecto.*

1.11.1.3

Modificadores

Names

1.11.1.3.1void	**texto** (Cor texto)	*Modifica a cor do texto de todas as células do ecrã desenhadas usando este objecto.* 30
1.11.1.3.2void	**fundo** (Cor fundo)	*Modifica a cor do fundo de todas as células do ecrã desenhadas usando este objecto.* 30

1.11.1.3.1

void **texto** (Cor texto)

Modifica a cor do texto de todas as células do ecrã desenhadas usando este objecto.

Modifica a cor do texto de todas as células do ecrã desenhadas usando este objecto.

1.11.1.3.2

void **fundo** (Cor fundo)

Modifica a cor do fundo de todas as células do ecrã desenhadas usando este objecto.

Modifica a cor do fundo de todas as células do ecrã desenhadas usando este objecto.

1.11.2

class **Troco**

Esta classe serve para representar um troço de ecrã.

Public Members

Esta classe serve para representar um troço de ecrã. Os troços de ecrã são úteis para guardar partes do ecrã, que mais tarde podem ser desenhados onde se entender.

Uma variável do tipo `Troco` (leia-se `Troço`) guarda uma parte do ecrã com todas as suas propriedades. Esta classe é especialmente útil em operações como cortar e colar (*copy & paste*) para guardar a parte do ecrã que se pretende reproduzir e depois pôr este mesmo troço de ecrã noutra ou na mesma posição.

1.11.2.1

explicit **Troco** (Dimensao const& dimensao)

Construtor da classe.

Construtor da classe. É pouco usual ser usado directamente. Normalmente é a própria classe ecrã que constrói o troço e o devolve (e portanto o construtor mais usado é o construtor por cópia).

Parameters:

`dimensao`	Dimensão do troço de ecrã a guardar.

1.11.3

explicit **Ecra** (Cor texto = branco, Cor fundo = preto, bool limita_cursor = false)

Construtor da classe.

Construtor da classe. Tal como a biblioteca se encontra definida, este construtor não pode ser usado directamente! No entanto, numa versão futura, onde não exista variável global para representar o ecrã, o programador utilizador pode vir a usá-lo. Encarrega-se de inicializar o "screen management"Slang. Se se construir com `limita_cursor true`, o ecrã garantirá que o cursor se encontra dentro dos seus limites em cada instante. Por omissão essa verificação não é feita.

Parameters:

`texto`	Cor do texto das chamadas células de fundo, para asquais não foi indicado qualquer objecto cor (cor por omissão).
`fundo`	Cor do fundo das chamadas células de fundo, para asquais não foi indicado qualquer objecto cor (cor por omissão).
`limita_cursor`	Indica se a posição do cursor deve serlimitada à dimensão do ecrã.

1.11.4

~Ecra ()

Destrutor da classe.

Destrutor da classe. Encarrega-se de finalizar o "screen management"Slang.

1.11.5

Inspectores

Names

Dimensao **dimensao** () const
Devolve a dimensão corrente do ecrã.

Coordenada
cursor () const *Devolve as coordenadas do ecrã.*

1.11.5.1bool **redimensionado** () const
Indica se o ecrã foi redimensionado ou não. 32

1.11.5.1

bool **redimensionado** () const

Indica se o ecrã foi redimensionado ou não.

Indica se o ecrã foi redimensionado ou não. Isto é particularmente relevante se o programa estiver a correr numa consola dentro de um ambiente de janelas. Nessas circunstâncias o utilizador pode alterar a dimensão da janela, sendo com isso alterada a dimensão do ecrã. Se isso acontecer o programa deverá redesenhar e refrescar o ecrã.

1.11.6

Modificadores

Names

1.11.6.1

void **atributos** (Cor texto, Cor fundo)

Modifica a cor por omissão, ie, a cor das células que constituem o fundo do ecrã.

Modifica a cor por omissão, ie, a cor das células que constituem o fundo do ecrã.

Parameters:

`texto`	Nova cor do texto para as células do fundo do ecrã.
`fundo`	Nova cor do fundo para as células do fundo do ecrã.

1.11.6.2

void **cursor** (Coordenada const& posicao)

Modifica a posição do cursor.

Modifica a posição do cursor.

Parameters: `posicao` Nova posição do cursor: limitada ou não à dimensãodo ecrã consoante o modo de construção do ecrã (por omissão não há limitação).

1.11.6.3

void **desloca** (Tecla const& tecla)

Desloca cursor na direcção indicada pela tecla.

Desloca cursor na direcção indicada pela tecla.

Parameters: `tecla` A tecla. Se for uma seta, o cursor é deslocado. Senão for, o cursor mantém-se.

1.11.7

Métodos para cópia e cola de troços de ecrã

Names

1.11.7.1

Troco **copia** (Caixa const& caixa) const

Devolve uma cópia de um troço do ecrã.

Devolve uma cópia de um troço do ecrã.

Parameters: `caixa` O troço do ecrã é o que se encontra sob esta caixa.

1.11.7.2

Troco **copia** (Dimensao const& dimensao) const

Devolve uma cópia do troço de ecrã com um dada dimensão e com origem na posição corrente do cursor.

Devolve uma cópia do troço de ecrã com um dada dimensão e com origem na posição corrente do cursor.

Parameters: `dimensao` A dimensão do troço.

1.11.7.3

void **cola** (Troco const& troco, Coordenada const& origem = Coordenada())

Cola no ecrã, na posição indicada, um troço de ecrã.

Cola no ecrã, na posição indicada, um troço de ecrã. O cursor é deslocado para a posição imediatamente após a última coluna da última linha onde o troço foi colocado, a não ser que se tenha pedido para manter o cursor parado. Atenção!

O objecto cor que estava a ser usado antes da colagem é perdido! As próximas escritas são feitas usando a cor do fundo.

Parameters:

`troco`	Troço a colar.
`origem`	Posição onde o troço deve ser colado (por omissãocola no canto superior esquerdo do ecrã).

See Also: parado

1.11.8

Métodos para desenhar no ecrã

Names

void	**campainha** () const	*Toca a "campaínha"(normalmente o ecrã pisca).*
1.11.8.1void	**apagaFimDaLinha** ()	*Apaga (ie, pinta com cor do fundo) até ao fim da linha do cursor.* 37
1.11.8.2void	**apagaFimDoEcra** ()	*Apaga (ie, pinta com cor do fundo) até ao fim do ecrã a partir do cursor.* 37
void	**apaga** ()	*Apaga (i.e., pinta com cor do fundo) todo o ecrã.*
1.11.8.3void	**refresca** () const	*Refresca o ecrã, ou seja, reflecte no ecrã real as últimas alterações realizadas no ecrã virtual.* 37
1.11.8.4void	**refrescaTudo** () const	*Refresca totalmente o ecrã, ou seja, reflecte no ecrã real todo o ecrã virtual.* 37
1.11.8.5void	**desenhaCaixa** (Caixa const& caixa)	*Desenha uma caixa no ecrã, sendo a borda desenhada com caracteres especiais.* 38

1.11.8.1

void **apagaFimDaLinha** ()

Apaga (ie, pinta com cor do fundo) até ao fim da linha do cursor.

Apaga (ie, pinta com cor do fundo) até ao fim da linha do cursor.

1.11.8.2

void **apagaFimDoEcra** ()

Apaga (ie, pinta com cor do fundo) até ao fim do ecrã a partir do cursor.

Apaga (ie, pinta com cor do fundo) até ao fim do ecrã a partir do cursor.

1.11.8.3

void **refresca** () const

Refresca o ecrã, ou seja, reflecte no ecrã real as últimas alterações realizadas no ecrã virtual.

Refresca o ecrã, ou seja, reflecte no ecrã real as últimas alterações realizadas no ecrã virtual.

1.11.8.4

void **refrescaTudo** () const

Refresca totalmente o ecrã, ou seja, reflecte no ecrã real todo o ecrã virtual.

Refresca totalmente o ecrã, ou seja, reflecte no ecrã real todo o ecrã virtual.

1.11.8.5

void **desenhaCaixa** (Caixa const& caixa)

Desenha uma caixa no ecrã, sendo a borda desenhada com caracteres especiais.

Desenha uma caixa no ecrã, sendo a borda desenhada com caracteres especiais.

Parameters: `caixa` Caixa a desenhar.

1.11.9

Operadores de inserção no ecrã de tipos usuais

Names

1.11.9.1

Ecra& **operator <<** (char c)

Insere um caracter na posição do cursor.

Insere um caracter na posição do cursor. O cursor é deslocado uma posição para a direita, a não ser que se tenha pedido para manter o cursor parado. Se se tiver especificado uma largura antes, são acrescentados espaços até perfazer a largura indicada.

Parameters: `c` O caractere a inserir.
See Also: parado
largura

1.11.9.2

Ecra& **operator <<** (int i)

Insere um número inteiro na base 10 na posição do cursor.

Insere um número inteiro na base 10 na posição do cursor. O cursor é deslocado para a posição à direita do último dígito escrito, a não ser que se tenha pedido para manter o cursor parado. Se se tiver especificado uma largura antes, são acrescentados espaços até perfazer a largura indicada. Se a largura for insuficiente para representar o inteiro, só são mostrados os primeiros dígitos.

Parameters: `i` O inteiro a inserir.
See Also: parado
largura

1.11.9.3

Ecra& **operator <<** (string const& c)

Insere uma cadeia de caracteres na posição do cursor.

Insere uma cadeia de caracteres na posição do cursor. O cursor é deslocado para a posição à direita do último caractere escrito, a não ser que se tenha pedido para manter o cursor parado. Se se tiver especificado uma largura antes, são acrescentados espaços até perfazer a largura indicada. Se a cadeia for maior que a largura especificada, só são mostrados os primeiros caracteres.

Parameters: `c` A cadeia a inserir.
See Also: parado
largura

1.11.10

Operadores de inserção de objectos especiais no ecrã

Names

1.11.10.1Ecra& **operator <<** (Caixa const& caixa)

1.11.10.1

Ecra& **operator <<** (Caixa const& caixa)

Desenha uma caixa no ecrã, sendo a borda desenhada com caracteres especiais.

Desenha uma caixa no ecrã, sendo a borda desenhada com caracteres especiais.

Parameters: `caixa` Caixa a desenhar.
See Also: desenhaCaixa (→1.11.8.5, *page 38*)

1.11.10.2

Ecra& **operator <<** (Troco const& troco)

Cola no ecrã, na posição do cursor, um troço de ecrã.

Cola no ecrã, na posição do cursor, um troço de ecrã. O cursor é deslocado para a posição imediatamente após a última coluna da última linha onde o troço foi colocado, a não ser que se tenha pedido para manter o cursor parado.

Parameters: `troco` Troço a colar.
See Also: parado
cola

1.11.10.3

Ecra& **operator <<** (Coordenada const& posicao)

Altera a posição do cursor para a posição dada.

Altera a posição do cursor para a posição dada.

Parameters: `posicao` Nova posição do cursor.
See Also: cursor

1.11.10.4

Ecra& **operator <<** (ObjectoCor const& objecto_cor)

Altera o objecto cor das próximas operações de escrita no ecrã (com excepção das colagens!).

Altera o objecto cor das próximas operações de escrita no ecrã (com excepção das colagens!).

Parameters: `objecto_cor` Objecto cor a usar nas próximas escritas noecrã.
See Also: cola

1.11.11

Ecra const& **operator >>** (Troco& troco) const

Operadores de extracção de um troço do ecrã.

Operadores de extracção de um troço do ecrã. Extrai um troço a partir da posição corrente do cursor.

Parameters: `troco` Troço a extrair (copiar).

1.12 extern Ecra **ecra**

Uma variável global representando o ecrã.

Uma variável global representando o ecrã. A ideia é que substitua a variável global `cout` usada normalmente para escrever no ecrã. Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.13 Manipuladores do ecrã

Names

Definidos no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.13.1 refresca

Manipulador usado para refrescar o ecrã. Usado como se segue:

```
// Coloca cursor no canto superior esquerdo do ecrã:
ecra << cursor(0, 0);
// Escreve "Olá mundo!" a partir dessa posição:
```

```
ecra << "Olá mundo";
// Refresca o ecrã:
ecra << refresca;
```

See Also: Ecra::refresca

1.13.2 refresca_tudo

Manipulador usado para fazer o refrescamento total do ecrã. Usado como se segue:

```
ecra << refresca_tudo;
```

See Also: Ecra::refrescaTudo

1.13.3 cursor

Manipulador usado para colocar o cursor numa dada posição. Usar como se segue:

```
// Coloca cursor na linha 10 coluna 20 do ecrã:
ecra << cursor(10, 20);
// Escreve "Olá mundo!" a partir dessa posição:
ecra << "Olá mundo";
// Refresca o ecrã:
ecra << refresca;
```

See Also: Ecra::cursor

1.13.4 parado

Manipulador usado para manter o cursor na posição corrente depois da **próxima** operação de escrita no ecrã. Usar como se segue:

```
// Coloca cursor no canto superior esquerdo do ecrã:
ecra << cursor(0, 0);
// Escreve "Olá mundo!" mantendo a posição do cursor no topo superior
// esquerdo do ecrã:
ecra << parado << "Olá mundo";
// Refresca o ecrã:
ecra << refresca;
```

1.13.5 largura

Manipulador usado para indicar o número de caracteres a escrever na **próxima** operação de escrita de tipos usuais (`char`, `int`, `std::string`). Usar como se segue:

```
// Coloca cursor no canto superior esquerdo do ecrã:
ecra << cursor(0, 0);
// Define objecto cor para a próxima escrita:
Ecra::ObjectoCor cor(amarelo, vermelho);
// Escreve "Olá mundo! " usando a cor indicada e ocupando 30 caracteres:
ecra << oor << largura(30) << "Olá mundo!";
```

1.13.6 fundo

Manipulador usado para indicar que as próximas escritas devem ser realizadas usando a cor do fundo do ecrã. Usar como se segue:

```
// Coloca cursor no canto superior esquerdo do ecrã:
ecra << cursor(0, 0);
// Define objecto cor para a próxima escrita:
Ecra::ObjectoCor cor(amarelo, vermelho);
// Escreve "Olá mundo! " usando a cor indicada:
ecra << oor << "Olá mundo! ";
// Escreve "Olá mundo!" mas usando a cor do fundo:
ecra << fundo << "Olá mundo!";
// Refresca o ecrã:
ecra << refresca;
```

1.13.7 caixa

Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição. Usar como se segue:

```
// Desenha caixa cor origem na linha 10 coluna 20 com 5 linhas e 15
// colunas:
ecra << caixa(10, 20, 5, 15);
// Refresca o ecrã:
ecra << refresca;
```

See Also: Ecra::desenhaCaixa

1.13.8 apaga

Manipulador usado para apagar o ecrã (ou seja, para o pintar com a cor do fundo). Usar como se segue:

```
ecra << apaga;
// Refresca o ecrã:
ecra << refresca;
```

See Also: Ecra::apaga

1.13.9 apaga_fim_da_linha

Manipulador usado para apagar até ao fim da linha do cursor (ou seja, para o pintar com a cor do fundo). Usar como se segue:

```
// Apaga linha 10 a partir da coluna 20:
ecra cursor(10, 20) << apaga_fim_da_linha;
// Refresca o ecrã:
ecra << refresca;
```

See Also: Ecra::apagaFimDaLinha

1.13.10 apaga_fim_do_ecra

Manipulador usado para apagar desde a posição do cursor até ao fim do ecrã (ou seja, para o pintar com a cor do fundo). Usar como se segue:

```
// Apaga ecrã a partir da coluna 20 da linha 10:
ecra cursor(10, 20) << apaga_fim_do_ecra;
// Refresca o ecrã:
ecra << refresca;
```

See Also: Ecra::apagaFimDoEcra

1.13.11 campainha

Manipulador usado para fazer soar a campainha. Usar como se segue:

```
ecra << campainha;
```

See Also: Ecra::campainha

1.14

extern string const **nomes_das_cores** [numero_de_cores]

Constante global com os nomes das cores.

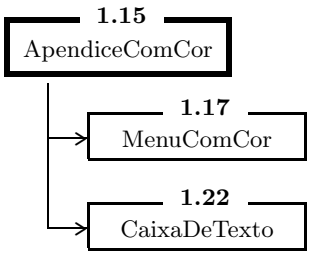
Constante global com os nomes das cores. Indexável com os valores do tipo enumerado `Cor`. Definida no módulo `ecra` (ficheiro de interface `Slang/ecra.H`).

1.15

class **ApendiceComCor**

Esta classe abstracta serve para definir as ferramentas relacionadas com cores para os menus que as possuirem.

Inheritance



Public Members

ApendiceComCor ()
Construtor da classe (atribui cores por omissão):

Esta classe abstracta serve para definir as ferramentas relacionadas com cores para os menus que as possuirem. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

São definidas cores para a borda, o título, os itens e o item corrente do menu. As cores usadas são (por omissão):

- borda: preto sobre branco (ou cinzento).

- título: vermelho sobre branco (ou cinzento).
- itens: preto sobre branco (ou cinzento).
- item corrente: amarelo sobre azul.

Um apêndice é uma classe que deve ser vista como abstracta, apesar de ser claramente uma classe concreta. Serva para equipar outras classes de ferramentas mais ou menos avulsas. Estas classes são conhecidas também por *mixin*.

1.15.1 Inspectores

Names

Ecra::ObjectoCor const&
o_borda () const — *Devolve a cor da borda.*

Ecra::ObjectoCor const&
o_titulo () const — *Devolve a cor do título.*

Ecra::ObjectoCor const&
o_itens () const — *Devolve a cor dos itens.*

Ecra::ObjectoCor const&
o_corrente () const
Devolve a cor do item corrente.

Inspectores

1.15.2 Modificadores

Names

Ecra::ObjectoCor&
o_borda () — *Devolve referência para a cor da borda.*

Ecra::ObjectoCor&
o_titulo () — *Devolve referência para a cor do título.*

Ecra::ObjectoCor&
o_itens () — *Devolve referência para a cor dos itens.*

Ecra::ObjectoCor&

o_corrente ()	*Devolve referência para a cor do item seleccionado.*

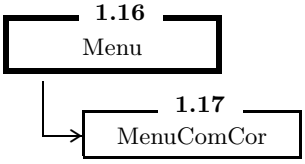
Modificadores

1.16

class **Menu**

Esta classe abstracta serve para definir a interface básica de todos os menus.

Inheritance



Public Members

1.16.1	explicit	**Menu** (std::string const& titulo)	*Construtor da classe.*	50
	virtual	**~Menu** ()	*Destrutor virtual para poder sofrer derivações...*	
1.16.2	virtual	int **executa** ()	*Função que executa o menu, devolvendo o número da opção escolhida pelo utilizador (a primeira opção ou item do menu tem número 0).*	50
	string	**titulo** () const	*Inspector do título do menu.*	

Esta classe abstracta serve para definir a interface básica de todos os menus. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.16.1

explicit **Menu** (std::string const& titulo)

Construtor da classe.

Construtor da classe.

Parameters: `titulo` Título do menu.

1.16.2

virtual int **executa** ()

Função que executa o menu, devolvendo o número da opção escolhida pelo utilizador (a primeira opção ou item do menu tem número 0).

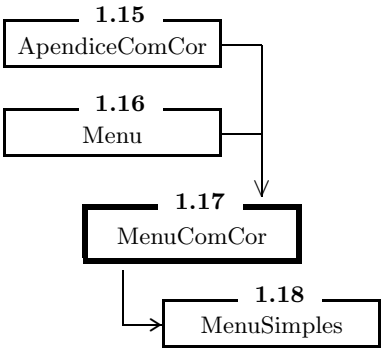
Função que executa o menu, devolvendo o número da opção escolhida pelo utilizador (a primeira opção ou item do menu tem número 0).

1.17

class **MenuComCor** : public Menu, public ApendiceComCor

Esta classe abstracta serve para definir a interface básica de todos os menus com cores.

Inheritance



Public Members

Esta classe abstracta serve para definir a interface básica de todos os menus com cores. A parte das cores é herdada do apêndice (*mixin*) `ApendiceComCor`. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.17.1

explicit **MenuComCor** (std::string const& titulo)

Construtor da classe.

Construtor da classe.

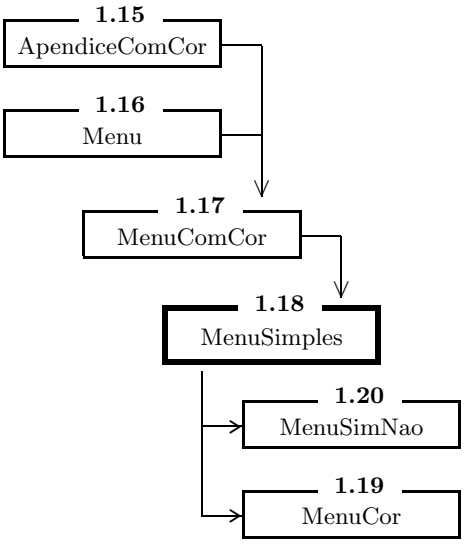
Parameters: `titulo` Título do menu.

1.18

class **MenuSimples** : public MenuComCor

Esta classe serve para representar menus simples, que consistem numa sequência de itens.

Inheritance



Public Members

Esta classe serve para representar menus simples, que consistem numa sequência de itens. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

Exemplo de utilização

Este programa mostra um menu simples no ecrã e escreve a opção que for pressionada até ser seleccionada a opção "Bazar".

```
#include <Slang/slang.H>

using namespace Slang;

int main ()
{
    string opcoes[] = {"Bazar",
       "Nao faz nada...",
       "Esta também não!",
       "Nem esta..."};
    int numero_de_opcoes = sizeof(opcoes) / sizeof(string);

    MenuSimples menu("Um menú que não faz nada!", opcoes,
                     numero_de_opcoes);

    while(int opcao = menu.executa())
ecra << parado << largura(20) << opcoes[opcao] << refresca;
}
```

1.18.1

MenuSimples (std::string const& titulo, std::string const itens[], int numero_de_itens)

Construtor da classe.

Construtor da classe.

Parameters:

`titulo`	Título do menu.
`itens`	Matriz de cadeias de caracteres com os nomes doitens.
`numero_de_itens`	Número de itens (e de elementos na matriz).

1.18.2

MenuSimples (std::string const& titulo, std::string const& itens)

Construtor da classe.

Construtor da classe.

Parameters:

`titulo`	Título do menu.
`itens`	Cadeia de caracteres contendo os itens separados porfind-de-linha (`'\n'`).

1.18.3

virtual int **executa** ()

Função que executa o menu, devolvendo o número da opção escolhida pelo utilizador (a primeira opção ou item do menu tem número 0).

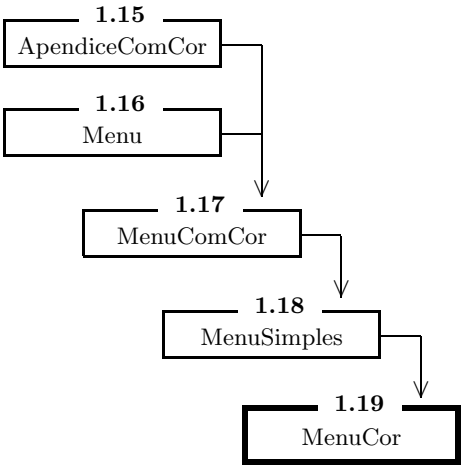
Função que executa o menu, devolvendo o número da opção escolhida pelo utilizador (a primeira opção ou item do menu tem número 0).

1.19

class **MenuCor** : public MenuSimples

Esta classe serve para representar menus de selecção das cores básicas usáveis no ecrã.

Inheritance



Public Members

Esta classe serve para representar menus de selecção das cores básicas usáveis no ecrã. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.19.1

explicit **MenuCor** (std::string const& titulo)

Construtor da classe.

Construtor da classe.

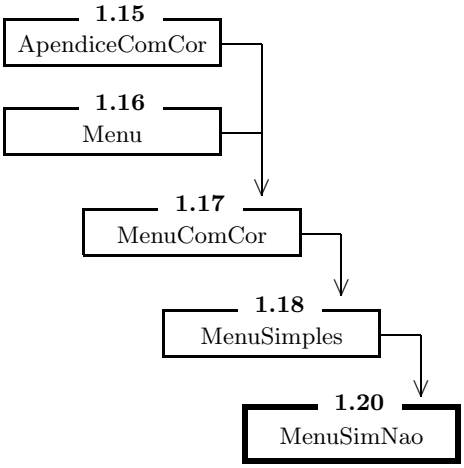
Parameters: `titulo` Título do menu.

1.20

class **MenuSimNao** : public MenuSimples

Esta classe serve para representar menus com apenas duas opções: sim e não.

Inheritance



Public Members

Esta classe serve para representar menus com apenas duas opções: sim e não. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.20.1

explicit **MenuSimNao** (std::string const& titulo)

Construtor da classe.

Construtor da classe.

Parameters: `titulo` Título do menu.

1.20.2

virtual int **executa** ()

Função que executa o menu, devolvendo a opção escolhida pelo utilizador (1 para sim, 0 para não).

Função que executa o menu, devolvendo a opção escolhida pelo utilizador (1 para sim, 0 para não).

1.21

class **Aviso**

Esta classe serve para representar caixas de aviso, que apenas mostram uma mensagem e das quais se sai pressionando 'enter'.

Public Members

Esta classe serve para representar caixas de aviso, que apenas mostram uma mensagem e das quais se sai pressionando 'enter'. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.21.1

explicit **Aviso** (std::string const& mensagem)

Construtor da classe.

Construtor da classe.

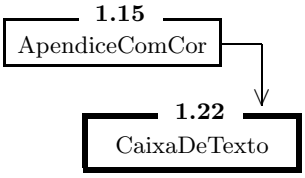
Parameters: `mensagem` Mensagem a mostrar.

1.22

class **CaixaDeTexto** : public ApendiceComCor

Esta classe serve para representar caixas de texto, que quando executadas permitem ao utilizador escrever uma cadeia de caracteres.

Inheritance



Public Members

1.22.1 explicit **CaixaDeTexto** (std::string const& titulo,
std::string const& valor = ,
std::string::size_type espacos = 0)
Construtor da classe. 58

virtual **~CaixaDeTexto** ()
Destrutor virtual para poder sofrer derivações...

1.22.2 virtual std::string
executa (bool impede_vazia = false)
Função que executa a caixa de texto, devolvendo a cadeia preenchida pelo utilizador. 58

std::string **titulo** () const *Inspector do título do menu.*

Esta classe serve para representar caixas de texto, que quando executadas permitem ao utilizador escrever uma cadeia de caracteres. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.22.1

explicit **CaixaDeTexto** (std::string const& titulo, std::string const& valor = , std::string::size_type espacos = 0)

Construtor da classe.

Construtor da classe.

Parameters:

`titulo`	Título da caixa.
`valor`	Valor inicial da caixa (por omissão a cadeia decaracteres vazia).
`espacos`	Número de espaços a usar para a edição (pode serespandido se o título for mais largo).

1.22.2

virtual std::string **executa** (bool impede_vazia = false)

Função que executa a caixa de texto, devolvendo a cadeia preenchida pelo utilizador.

Função que executa a caixa de texto, devolvendo a cadeia preenchida pelo utilizador.

1.23

class **Tecla**

Esta classe serve para representar teclas premidas.

Public Members

Esta classe serve para representar teclas premidas. As teclas podem corresponder a caracteres (e.g., 'a', 'x', '1', '.') ou a teclas de controlo (e.g., 'delete', 'home', etc.). Definida no módulo `teclado` (ficheiro de interface `Slang/teclado.H`).

1.23.1

enum **TeclaEnum**

Este tipo enumerado representa as várias teclas que podem ser premidas.

Members

refresca	*Tecla de refrescamento ('ctrl-l').*
cima	*Seta para cima.*
baixo	*Seta para baixo.*
esquerda	*Seta para a esquerda.*
direita	*Seta para a direita.*
pagina_anterior	*Tecla 'page up' ou 'página anterior'.*
pagina_seguinte	*Tecla 'page down' ou 'página seguinte'.*
casa	*Tecla 'home' ou 'casa'.*

fim	*Tecla 'end' ou 'fim'.*
refaz	*Tecla 'redo' ou 'refaz' (disponível só em alguns teclados).*
desfaz	*Tecla 'undo' ou 'desfaz' (disponível só em alguns teclados).*
apaga_para_tras	*Tecla 'backspace' ou 'apaga para trás'.*
entrada	*Tecla 'enter' ou 'entrada'.*
insere	*Tecla 'insert' ou 'insere'.*
apaga	*Tecla 'delete' ou 'apaga'.*
F1	*Tecla 'F1'.*
F2	*Tecla 'F2'.*
F3	*Tecla 'F3'.*
F4	*Tecla 'F4'.*
F5	*Tecla 'F5'.*
F6	*Tecla 'F6'.*
F7	*Tecla 'F7'.*
F8	*Tecla 'F8'.*
F9	*Tecla 'F9'.*
F10	*Tecla 'F10'.*
F11	*Tecla 'F11'.*
F12	*Tecla 'F12'.*
nulo	*Caractere nulo.*
erro	*Valor em caso de erro.*

Este tipo enumerado representa as várias teclas que podem ser premidas. Não são enumerados os caracteres normais, mas os seus valores podem ser representados neste tipo enumerado. O par de tipos `TeclaEnum` e `Tecla` permite usar uma classe como se de um tipo enumerado se tratasse. Há conversões implícitas entre os dois tipos. Além disso a classe proporciona algumas funções membro úteis.

1.23.2

char **comoChar** () const

Função para conversão explícita para `char` *(só se deve usar se* `caractere()` *devolver* `true`*).*

Função para conversão explícita para `char` (só se deve usar se `caractere()` devolver `true`).

1.23.3

bool **deslocamento** () const

Devolve `true` *se for uma tecla de deslocamento (seta para a esquerda, direita, cima ou baixo).*

Devolve `true` se for uma tecla de deslocamento (seta para a esquerda, direita, cima ou baixo).

1.24

class **Teclado**

Esta classe é um solitão: a sua única instância representa o teclado.

Public Members

Esta classe é um solitão: a sua única instância representa o teclado. Definida no módulo `teclado` (ficheiro de interface `Slang/teclado.H`).

A classe `Teclado` permite receber informação sobre a pressão de qualquer tecla através da função membro `Tecla Teclado::tecla()`, que devolve uma instância da classe `Tecla`. Esta função não deve ser utilizada sem que haja uma tecla disponivel (i.e. sem que o utilizador tenha de facto pressionado uma tecla), a não ser que se pretenda esperar que o utilizador pressione uma tecla.

Para verificar se o utilizador carregou numa tecla, existe a função `bool teclaDisponivel(int decimos_de_segundo) const`. Esta função tem como

argumento o número décimos de segundo que a função espera pela pressão de uma tecla. Não deve ser usado zero como argumento desta função, se ela fôr usada dentro de um ciclo, dado que pode tornar o sistema muito lento (10 décimos de segundo é um valor aceitável para o argumento desta função nessas circunstâncias).

A função membro `void Teclado::limpa()` permite descartar as teclas que foram pressionadas pelo utilizador anteriormente e ainda não foram lidas e passar a considerar apenas as teclas que forem pressionadas a partir desse instante.

É definida uma variável global chamada `teclado` da classe `Teclado` (declarada no ficheiro de interface `Slang/teclado.H`), de modo que não é necessário, nem permitido, criar nenhuma variável deste tipo nos programas que fazem uso desta biblioteca, podendo-se utilizar directamente a variável `teclado` (como acontece no caso do canal `cin`).

Uma instância da classe `Tecla` pode ser uma tecla especial (definidas no enumerado `TeclaEnum`) ou um caracter normal. É possível verificar se o valor armazenado numa instância da classe `Tecla` é um deslocamento ou um caracter normal através das funções membro `bool Tecla::deslocamento() const` e `bool Tecla::caractere() const`.

Exemplo de utilização

Este programa captura todas as teclas premidas até ser premido o caracter 's'. Se forem deslocamentos procede ao deslocamento respectivo do cursor. Se forem caracteres escreve-os no ecrã. Se a tecla pressionada corresponder a um caractere que não se pode imprimir no ecrã é mostrada uma mensagem adequada. Se for outra tecla aparece outra mensagem apropriada.

```
#include <Slang/slang.H>

using namespace Slang;

#include <cctype> // para isprint().

using namespace std;

int main ()
{
    Ecra::ObjectoCor cor_normal(amarelo, preto);
    Ecra::ObjectoCor cor_aviso(vermelho, preto);

    while(true) {
if(teclado.teclaDisponivel(10)) {
    Tecla a = teclado.tecla();

    if(a.deslocamento())
ecra.desloca(a);
    else if(a.caractere())
if(isprint(a.comoChar()))
    ecra << cor_normal << a.comoChar();
```

```
else
    ecra << cor_aviso << parado
 << "Este nao se pode imprimir!";
    else
ecra << cor_aviso << parado << "Tecla inválida!";
    if(a == 's')
break;
}
ecra.refresca();
    }
}
```

1.24.1

bool **teclaDisponivel** (int decimos_de_segundo = 0) const

Devolve `true` *se alguma tecla tiver sido pressionada e portanto estiver disponível para leitura.*

Devolve `true` se alguma tecla tiver sido pressionada e portanto estiver disponível para leitura. Desiste ao fim do tempo passado como argumento (em décimos de segundo).

1.25

extern Teclado **teclado**

Uma variável global representando o teclado.

Uma variável global representando o teclado. A ideia é que substitua a variável global `cin` usada normalmente para ler do teclado.

1.26

Manipuladores extra para `std::istream`

Names

Definidos no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.26.1

il

Manipulador que ignora todos os caracteres até ao próximo fim-de-linha (`'\n'`). O nome é uma abreviatura de "ignora linha".

Exemplo de utilização

Se num ficheiro estiverem guardados em linhas consecutivas um inteiro e o nome completo de uma passoa, pode-se tentar ler estes valores como se segue (admite-se que `entrada` é um canal ligado ao ficheiro):

```
int numero;
entrada >> numero;
string nome;
getline(entrada, nome);
```

Esta solução não funciona, pois o operador de extracção do inteiro deixa o fim-de-linha no canal, o que leva `getline` a ler uma cadeia vazia! A solução passa por ignorar todos os caracteres até ao fim-de-linha:

```
#include <Slang/util.H>
using namespace Slang;
...
int numero;
entrada >> numero >> il;
string nome;
getline(entrada, nome);
```

Neste caso o ficheiro de entrada até pode possuir um comentário depois do inteiro, que será ignorado. Por exemplo, o ficheiro poderia ser:

```
12345 Número do aluno
Xisto Ximenes
```

(Que se teria de fazer para que se pudessem colocar comentários após o nome?)

1.26.2

ill

Manipulador que ignora todos os caracteres até ao próximo fim-de-linha (`'\n'`) **mas primeiro limpa possíveis condições de erro do canal**. O nome é uma abreviatura de "ignora linha limpa".

Exemplo de utilização

Suponha que se pretende ler do teclado um inteiro que tem de ser não-negativo (o número de um aluno, por exemplo). A solução óbvia é:

```
cout << "Introduza um número não negativo: ";
int numero;
cin >> numero;
cout << "O número é: " << numero << endl;
```

Esta solução tem dois problemas:

- Se a extracção tiver sucesso, não garante que o valor lido é não-negativo.
- Se a extracção não tiver sucesso, o canal `cin` fica com uma condição de erro, o que faz com que todas as extracções subsequentes falhem.

É importante perceber que, neste caso, se assume que ao teclado está um humano, que reconhece e espera poder corrigir os seus erros! Assim, a solução passa por escrever um ciclo:

```
    int numero;
    while(true) {
        cout << "Introduza um número não negativo: ";
        cin >> numero >> il;
if(numero >= 0)
    break;
cout << "Tem de ser não negativo!" << endl;
    }
    cout << "O número é: " << numero << endl;
```

Note-se na utilização do manipulador `il`, que é usado para que a leitura seja orientada por linha.

Esta solução resolve o primeiro problema, mas não o segundo... A solução passa por verificar também se a extração teve sucesso. Caso não tenha tido sucesso é necessário limpar a condição de erro e ignorar toda a linha. Dessa forma o utilizador pode voltar a tentar introduzir um número:

```
int numero;
```

```
    while(true) {
        cout << "Introduza um número não negativo: ";
        if(cin >> numero >> il && numero >= 0)
    break;
if(cin)
    cout << "Tem de ser não negativo!" << endl;
else {
    cout << "Isso não é um número!" << endl;
    // Ignora resto da linha limpando condição de erro.
    cin >> ill;
}
    }
    cout << "O número é: " << numero << endl;
```

Idealmente esta solução seria encapsulada numa função que devolvesse o número e fosse parametrizada pela condição a verificar pelo valor (neste caso tem de ser não negativo) e pelas mensagens a escrever no ecrã.

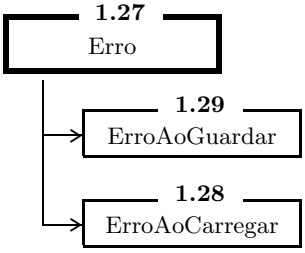
See Also: il

1.27

class **Erro**

Esta classe serve de base a uma pequena hierarquia de classes representando excepções.

Inheritance



Public Members

1.27.1 **Erro** (std::string const& mensagem)
Construtor da classe. 67

virtual **~Erro** () *Destrutor virtual para poder sofrer derivações...*

Esta classe serve de base a uma pequena hierarquia de classes representando excepções. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.27.1

Erro (std::string const& mensagem)

Construtor da classe.

Construtor da classe.

Parameters: `mensagem` Uma mensagem explicando a origem da excepção.

1.27.2

virtual **operator std::string** () const

Inspector da mensagem explicando a origem da excepção na forma de uma conversão implícita para `std::string`.

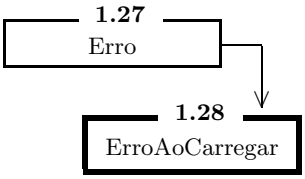
Inspector da mensagem explicando a origem da excepção na forma de uma conversão implícita para `std::string`.

1.28

class **ErroAoCarregar** : public Erro

Esta classe serve para representar excepções de carregamento de objectos a partir de canais.

Inheritance



Public Members

Esta classe serve para representar excepções de carregamento de objectos a partir de canais. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.28.1

ErroAoCarregar (std::string const& classe)

Construtor da classe.

Construtor da classe.

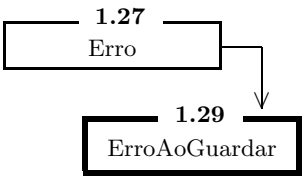
Parameters: `classe` O nome da classe que originou a excepção.

1.29

class **ErroAoGuardar** : public Erro

Esta classe serve para representar excepções ao guardar objectos usando canais.

Inheritance



Public Members

Esta classe serve para representar excepções ao guardar objectos usando canais. Definida no módulo `menu` (ficheiro de interface `Slang/menu.H`).

1.29.1

ErroAoGuardar (std::string const& classe)

Construtor da classe.

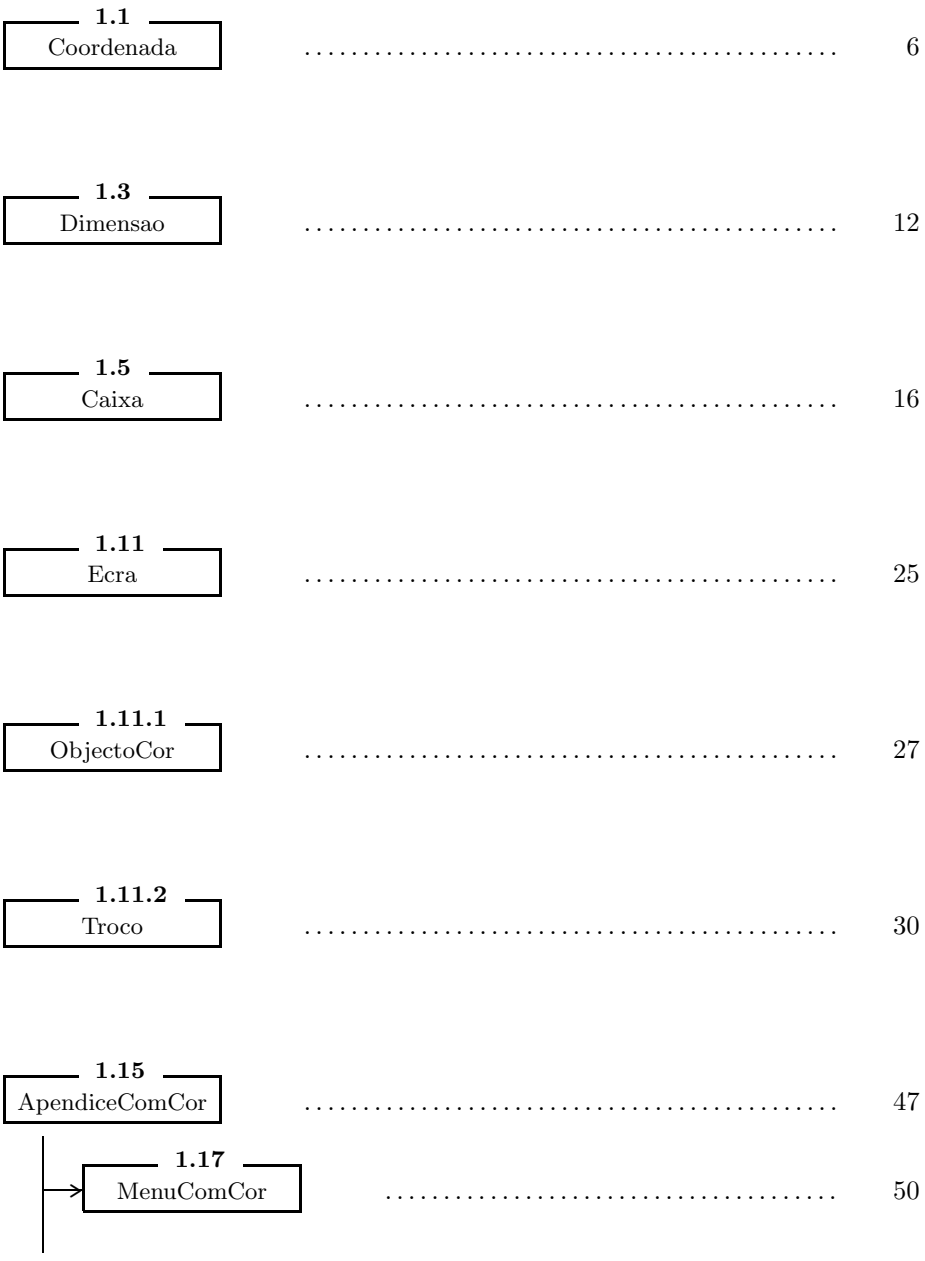
Construtor da classe.

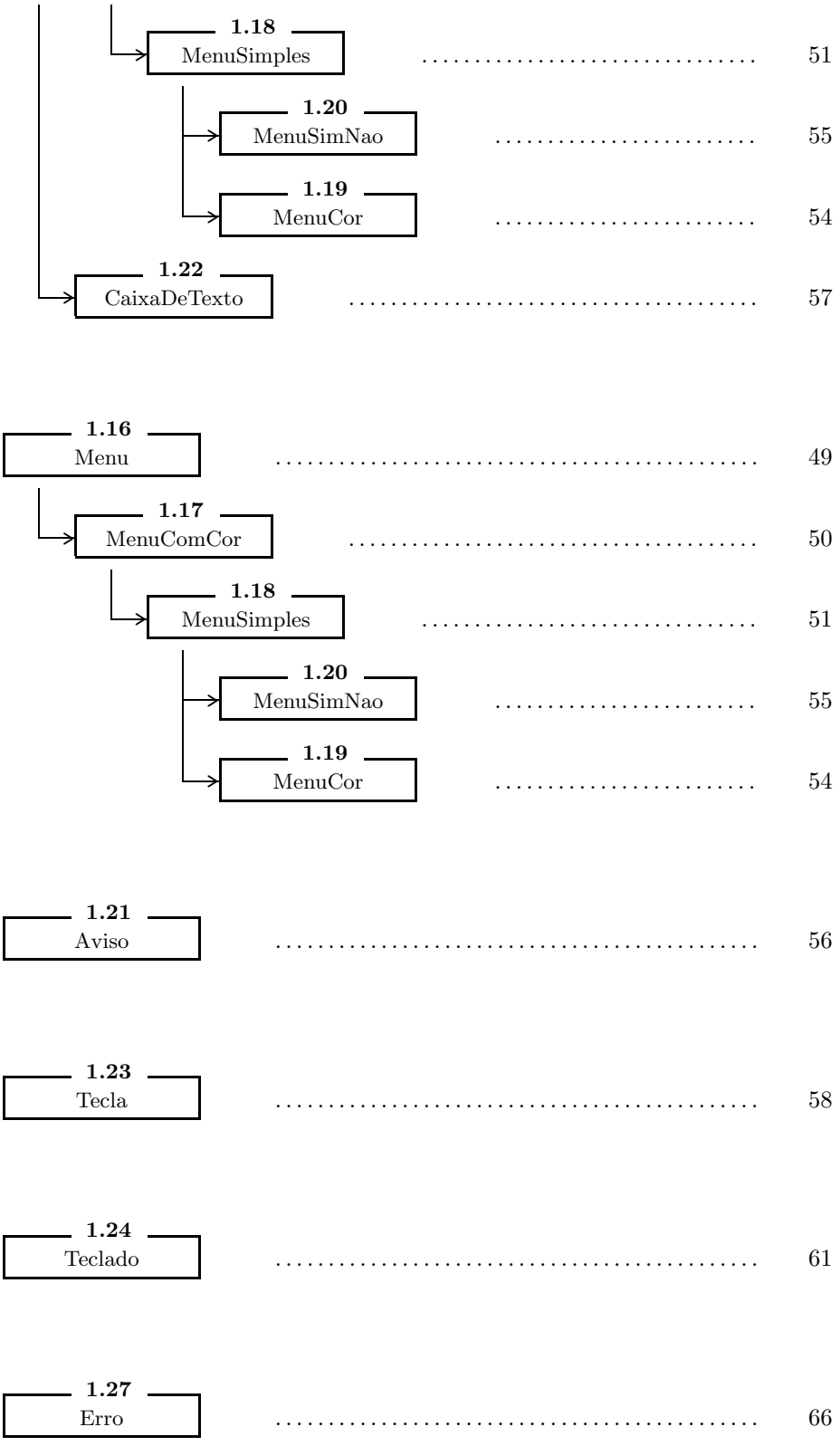
Parameters: `classe` O nome da classe que originou a excepção.

2

#define **SLANG_TECLADO_H**

Class Graph


1.1
Coordenada
6
1.3
Dimensao
12
1.5
Caixa
16
1.11
Ecra
25
1.11.1
ObjectoCor
27
1.11.2
Troco
30
1.15
ApendiceComCor
47
1.17
MenuComCor
50


1.18
MenuSimples
51
1.20
MenuSimNao
55
1.19
MenuCor
54
1.22
CaixaDeTexto
57
1.16
Menu
49
1.17
MenuComCor
50
1.18
MenuSimples
51
1.20
MenuSimNao
55
1.19
MenuCor
54
1.21
Aviso
56
1.23
Tecla
58
1.24
Teclado
61
1.27
Erro
66