Pacotes Manual de referência

Gerado por Doxygen 1.3-rc1

Tue Dec 3 15:19:34 2002

Conteúdo

Capítulo 1

O pacote Pacotes

Esta documentação pode ser encontrada em HTML em `http://iscte.pt/programacao/p1/pacotes/Pacotes.doc/html` e em PDF em `http://iscte.pt/programacao/p1/pacotes/Pacotes.doc/latex/refman.pdf`.

1.0.1 Introdução

Este pacote aglutina um conjunto dispar de bibliotecas, nomeadamente a biblioteca `Slang++`, para escrita de programas com interface alfanumérica, a biblioteca `IPC++`, para escrita de programas com comunicações entre processos, a biblioteca **Utilitarios**, que possui um conjunto diversificado de farramentas, lidando com datas, manipuladores e cadeias de caracteres, e a biblioteca **Erros**, que define algumas classes úteis para representar erros associados ao carregamento e guarda de dados em canais.

1.0.2 Instalação das bibliotecas do pacote Pacotes

Se pretender instalar as bibliotecas neste pacote em outro computador deve começar por instalar em primeiro lugar a biblioteca S-Lang (ver `http://www.s-lang.org/`). Em seguida, deve:

1. Importar o ficheiro `http://iscte.pt/programacao/p1/pacotes/Pacotes-1.0.tar.gz` (para importar no netscape basta fazer <shift-botão esquerdo do rato>).

2. Dar os seguintes comandos:

```
tar -zxf Pacotes-1.0.tar.gz
cd Pacotes
make
make test
```

   (Este último comando não deverá produzir qualquer mensagem de erro.)

```
su (inserir a senha [password] de administração da máquina)
make install
exit
```

1.0.3 Executando os exemplos

A distribuição desta biblioteca inclui um directório de exemplos. É recomendável que os construa e execute enquanto lê esta documentação.

Os exemplos podem ser encontrados nos sub-directório Exemplos das várias bibliotecas. Para construir os respectivos executáveis, dê os seguintes comandos (no directório `Slang++`):

```
make examples
```

Experimente já executar um dos exemplos. Dê o comando (no directório `Pacotes`):

```
xterm -e Slang++/Exemplos/exemplo2 &
```

1.0.4 Lendo a documentação

A melhor forma de aprender a usar as bibliotecas neste pacote é explorar esta documentação e os exemplos fornecidos. Recomenda-se que se comece a explorar a documentação na secção de Módulos, nomeadamente na sub-secção de Ferramentas de ecrã, que é parte da biblioteca `Slang++`.

Autor:
Manuel Menezes de Sequeira

Versão:
1.0

Data:
2002/11/30

Capítulo 2

Pacotes Índice dos módulos

2.1 Pacotes Módulos

Lista de todos os módulos:

Capítulo 3

Pacotes Índice dos namespaces

3.1 Pacotes Lista de namespaces

Lista dos namespaces com uma breve descrição:

Capítulo 4

Pacotes Índice da hierarquia

4.1 Pacotes Hierarquia de classes

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

Capítulo 5

Pacotes Índice dos componentes

5.1 Pacotes Lista de componentes

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

Capítulo 6

Pacotes Índice dos ficheiros

6.1 Pacotes Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

Capítulo 7

Pacotes Índice da página

7.1 Pacotes Páginas relacionadas

Lista de documentação relacionada:

Capítulo 8

Pacotes Documentação do módulo

8.1 Erros genéricos

Módulo correspondente ao ficheiro Erros/erros.H.

Ficheiros

- file **erros.H**

  Ficheiro de interface da biblioteca **Erros** *e do módulo erros.*

- file **erros_impl.H**

  Ficheiro auxiliar de implementação do módulo erros.

Componentes

- class **Erro**

  Base de uma pequena hierarquia de classes representando excepções.

- class **ErroAoCarregar**

  Representa excepções ocorridas durante o carregamento de objectos a partir de canais.

- class **ErroAoGuardar**

  Representa excepções ocorridas durante o armazenamento de objectos usando canais.

8.1.1 Descrição detalhada

Módulo correspondente ao ficheiro Erros/erros.H.

Contém ferramentas para lidar com erros genéricos.

8.2 Mensageiros entre processos

Módulo correspondente ao ficheiro IPC++/mensageiro.H.

Ficheiros

- file **mensageiro1.C**

  Programa que demostra a utilização de mensageiros para comunicar entre processos.

- file **mensageiro.H**

  Ficheiro de interface do módulo mensageiros.

- file **mensageiro_impl.H**

  Ficheiro auxiliar de implementação do módulo mensageiros.

- file **mensageiro.C**

  Ficheiro de implementação do módulo mensageiros.

Componentes

- class **Erro**

  Representa as excepções da biblioteca IPC++.

- class **Mensageiro**

  Classe cuja única instância (em cada processo) representa um mensageiro entre o processo em causa e um outro na mesma máquina.

8.2.1 Descrição detalhada

Módulo correspondente ao ficheiro IPC++/mensageiro.H.

Contém ferramentas para comunicar entre processos via mensageiros.

8.3 Ferramentas de ecrã

Módulo correspondente ao ficheiro Slang++/ecra.H.

Ficheiros

- file **ecra.C**

  Ficheiro de implementação do módulo ecras.

- file **caixa1.C**

  Programa que demostra o desenho de uma caixa no ecrã.

- file **campainha1.C**

  Este programa toca a campainha do ecrã a cada tecla premida, saindo quando se pressiona 's'.

- file **ecra1.C**

  Programa que demostra a utilização do ecrã de forma simples.

- file **ecra2.C**

  Programa que demostra a utilização do troços de ecrã e respectiva cópia e colagem.

- file **ecra3.C**

  Programa que permite ao utilizador deslocar o cursor através das setas do teclado, indicando em cada instante se o cursor está ou não visível.

- file **ecra4.C**

  Programa que demonstra que a alteração das cores do fundo do ecrã altera todas as células do ecrã que com ele tenham sido desenhadas.

- file **exemplo3.C**

  Programa que demostra a utilização do teclado e do ecrã de forma simples.

- file **justificacao1.C**

  Programa que demostra os vários tipos de justificação e o seu efeito.

- file **manipulador1.C**

  Programa que define um manipulador que move o cursor para uma posição arbitrária mas visível do ecrã.

- file **objectocor1.C**

  Programa que demostra o efeito dos objectos cor e da alterção das suas cores já depois de feitas inserções por seu intermédio.

- file **objectocor2.C**

  Programa que demonstra o efeito do tempo de vida sobre os objectos cor.

- file **objectocor3.C**

  Programa que demonstra que a alteração de um objecto cor altera todas as células do ecrã que com ele tenham sido desenhadas.

- file **segmentos1.C**

  Programa que demonstra a utilização de segmentos de recta horizontais e verticais.

- file **simbolo1.C**

  Programa que mostra os símbolos possíveis de inserir no ecrã.

- file **ecra.H**

  Ficheiro de interface do módulo ecras.

Componentes

- class **Posicao**

  Representa a posição de uma célula do ecrã.

- class **Dimensao**

  Representa uma dimensão em células do ecrã.

- class **Caixa**

  Representa caixas no ecrã.

- class **Ecra**

  Representa o ecrã.

- 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).

- class **Troco**

  Esta classe serve para representar um troço de ecrã.

Operadores associados ao tipo enumerado Cor

- std::ostream & **operator**<< (std::ostream &saida, **Cor** const cor)

  Insere uma cor num canal.

- std::istream & **operator**>> (std::istream &entrada, **Cor** &cor)

  Extrai uma cor de um canal.

- int const **numero_de_cores**

  Constante que guarda o número total de cores.

Manipuladores do ecrã

- Posicao const **cursor** (int const linha, int const coluna)

  Manipulador usado para colocar o cursor numa dada posição.

- Posicao const & **cursor** (Posicao const &posicao)

  Manipulador usado para colocar o cursor numa dada posição.

- Caixa const **caixa** (int const linha, int const coluna, int const numero_de_linhas, int const numero_de_colunas)

  Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição e com uma dada dimensão.

- Caixa const **caixa** (Posicao const &origem=Posicao(), Dimensao const &dimensao=Dimensao())

  Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição e com uma dada dimensão.

- void **fundo** (Ecra &**ecra**)

  Manipulador usado para indicar que as próximas escritas devem ser realizadas usando a cor do fundo do ecrã.

- void **parado** (Ecra &**ecra**)

  Manipulador usado para manter o cursor na posição corrente durante a próxima operação de inserção no ecrã.

- void **sobeCursor** (Ecra &**ecra**)

  Manipulador usado para subir o cursor no ecrã.

- void **baixaCursor** (Ecra &**ecra**)

  Manipulador usado para baixar o cursor no ecrã.

- void **recuaCursor** (Ecra &**ecra**)

  Manipulador usado para recuar o cursor no ecrã.

- void **avancaCursor** (Ecra &**ecra**)

  Manipulador usado para avançar o cursor no ecrã.

- ManipuladorDaLargura **largura** (int const largura)

  Manipulador usado para indicar o número de células do ecrã a ocupar na próxima operação de inserção no ecrã.

- ManipuladorDeDesenhoDeSegmento **segmento_horizontal** (int const largura)

  Manipulador usado para desenhar um segmento de recta horizontal.

- ManipuladorDeDesenhoDeSegmento **segmento_vertical** (int const altura)

  Manipulador usado para desenhar um segmento de recta vertical.

- void **refresca** (Ecra &**ecra**)

  Manipulador usado para refrescar o ecrã.

- void **refresca_tudo** (Ecra &**ecra**)

  Manipulador usado para fazer o refrescamento total do ecrã.

- void **apaga** (Ecra &**ecra**)

  Manipulador usado para apagar o ecrã (ou seja, para o pintar com a cor do fundo).

- void **apaga_fim_da_linha** (Ecra &**ecra**)

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

- void **apaga_fim_do_ecra** (Ecra &**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).

- void **campainha** (Ecra &**ecra**)

  Manipulador usado para fazer soar a campainha.

Constantes associadas à classe Caixa

- Caixa const **caixa_vazia**

  Constante representando uma caixa vazia, i.e., com dimensão nula.

- Caixa const **caixa_universal**

  Constante representando a caixa universal (enche todo o espaço).

Enumerações

- enum **Cor** { **preto**, **primeira_cor** = preto, **cinza**, **vermelho**, **vermelho_brilhante**, **verde**, **verde_brilhante**, **castanho**, **amarelo**, **azul**, **azul_brilhante**, **magenta**, **magenta_brilhante**, **ciano**, **ciano_brilhante**, **cinzento_claro**, **branco**, **ultima_cor** = branco }

  Representa as possíveis cores do texto e do fundo de cada célula do ecrã.

- enum **Simbolo** { **diamante** = '`', **tijolo**, **ht**, **ff**, **cr**, **lf**, **grau**, **mais_menos**, **nl**, **vt**, **canto_inferior_direito**, **canto_superior_direito**, **canto_superior_esquerdo**, **canto_inferior_esquerdo**, **cruzamento**, **traco_topo**, **traco_superior**, **traco_horizontal**, **traco_inferior**, **traco_base**, **encontro_direita**, **encontro_esquerda**, **encontro_cima**, **encontro_baixo**, **traco_vertical**, **menor_ou_igual**, **maior_ou_igual**, **pi**, **diferente** }

  Representa caracteres especiais úteis para desenhar no ecrã caixas e outros símbolos pouco usuais.

- enum **Justificacao** { **ao_centro**, **a_esquerda**, **a_direita** }

  Representa os vários tipos de justificação possíveis ao escrever texto no ecrã.

Variáveis

- Ecra **ecra**

  Uma variável global representando o ecrã.

- std::string const **nomes_das_cores** [**numero_de_cores**]

  Constante global com os nomes das cores.

8.3.1 Descrição detalhada

Módulo correspondente ao ficheiro Slang++/ecra.H.

Contém ferramentas para escrever e manipular o ecrã.

8.3.2 Documentação dos valores da enumeração

8.3.2.1 enum Slang::Cor

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 funcionam bem... Ver na documentação abaixo os nomes das cores usados para inserção e extração de canais.

Valores da enumeração:

preto Num canal: `preto`.

primeira_cor

cinza Num canal: `cinza`.

vermelho Num canal: `vermelho`.

vermelho_brilhante Num canal: `vermelho-brilhante`.

verde Num canal: `verde`.

verde_brilhante Num canal: `verde-brilhante`.

castanho Num canal: `castanho`.

amarelo Num canal: `amarelo`.

azul Num canal: `azul`.

azul_brilhante Num canal: `azul-brilhante`.

magenta Num canal: `magenta`.

magenta_brilhante Num canal: `magenta-brilhante`.

ciano Num canal: `ciano`.

ciano_brilhante Num canal: `ciano-brilhante`.

cinzento_claro Num canal: `cinzento-claro`.

branco Num canal: `branco`.

ultima_cor

Definido na linha 921 do ficheiro ecra.H.

Referenciado por Slang::Ecra::Ecra(), main(), Slang::Ecra::ObjectoCor::mudaCorDoFundoPara(), Slang::Ecra::ObjectoCor::mudaCorDoTextoPara(), Slang::Ecra::mudaCoresDasCelulasDoFundoPara(), Slang::Ecra::ObjectoCor::ObjectoCor() e Slang::operator>>().

8.3.2.2 enum Slang::Simbolo

Representa caracteres especiais úteis para desenhar no ecrã caixas e outros símbolos pouco usuais.

Pode ver o aspecto destes símbolos executando o seguinte programa de exemplo:

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

using namespace Slang;

int main()
{
    for(int s = int(diamante); s != int(diferente) + 1; ++s)
        ecra << largura(2) << Simbolo(s);

    ecra << refresca;

    teclado.leProximaTeclaDisponivel();
}
```

Valores da enumeração:

- **diamante**
- **tijolo**
- **ht**
- **ff**
- **cr**
- **lf**
- **grau**
- **mais_menos**
- **nl**
- **vt**
- **canto_inferior_direito**
- **canto_superior_direito**
- **canto_superior_esquerdo**
- **canto_inferior_esquerdo**
- **cruzamento**
- **traco_topo**
- **traco_superior**
- **traco_horizontal**
- **traco_inferior**
- **traco_base**
- **encontro_direita**
- **encontro_esquerda**
- **encontro_cima**
- **encontro_baixo**
- **traco_vertical**
- **menor_ou_igual**
- **maior_ou_igual**
- **pi**
- **diferente**

Definido na linha 979 do ficheiro ecra.H.

Referenciado por main() e Slang::Ecra::operator<<().

8.3.2.3 enum Slang::Justificacao

Representa os vários tipos de justificação possíveis ao escrever texto no ecrã.

O valor a_esquerda (que é a justificação por omissão do ecrã) faz com que as entidades inseridas no ecrã fiquem à esquerda do espaço para eles reservado. O valor a_direita alinha as entidades à direita. O valor ao_centro leva as entidades inseridas a serem colocadas ao centro do espaço para elas reservado através, por exemplo, do manipulador **largura()**. Se o espaço reservado for insuficiente, a entidade inserida será truncada à direita, de ambos os lados, ou à esquerda, consoante a justificação escolhida seja à esquerda, ao centro ou à direita.

O programa que se segue ilustra o funcionamento destes valores enumerados, que ao serem inseridos no ecrã alteram a justificação de forma permanente:

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

using namespace std;
using namespace Slang;

int main()
{
    char caractere = '*';
    string cadeia("abcde");

    Ecra::ObjectoCor cor(amarelo, azul);

    ecra << cor;

    ecra << parado << largura(10) << a_esquerda << caractere << baixaCursor;
    ecra << parado << largura(10) << ao_centro << caractere << baixaCursor;
    ecra << parado << largura(10) << a_direita << caractere << baixaCursor;

    ecra << parado << largura(10) << a_esquerda << cadeia << baixaCursor;
    ecra << parado << largura(10) << ao_centro << cadeia << baixaCursor;
    ecra << parado << largura(10) << a_direita << cadeia << baixaCursor;

    ecra << parado << largura(3) << a_esquerda << cadeia << baixaCursor;
    ecra << parado << largura(3) << ao_centro << cadeia << baixaCursor;
    ecra << parado << largura(3) << a_direita << cadeia << baixaCursor;

    ecra << refresca;

    teclado.leProximaTeclaDisponivel();
}
```

Veja também:
largura()

Valores da enumeração:

ao_centro Manipulador usado para colocar justificação centrada.

a_esquerda Manipulador usado para colocar justificação à esquerda.

a_direita Manipulador usado para colocar justificação à direita.

Definido na linha 1034 do ficheiro ecra.H.

Referenciado por Slang::Ecra::mudaJustificacaoPara() e Slang::Ecra::operator<<().

8.3.3 Documentação das funções

8.3.3.1 std::ostream & Slang::operator<< (std::ostream & *saida*, Cor const *cor*) [inline]

Insere uma cor num canal.

Precondição:
V.

Poscondição:
não saida.good() ou (saida.good() e canal contém a representação textual da cor).

Definido na linha 531 do ficheiro ecra_impl.H.

8.3.3.2 Slang::Posicao const Slang::cursor (int const *linha*, int const *coluna*) [inline]

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;
```

Veja também:
Ecra::moveCursorPara()

Definido na linha 1122 do ficheiro ecra_impl.H.

8.3.3.3 istream & Slang::operator>> (std::istream & *entrada*, Cor & *cor*)

Extrai uma cor de um canal.

Precondição:
cor = c.

Poscondição:
(não entrada.good() e cor = c) ou (entrada.good() e canal já não contém a representação textual da cor e cor = cor cuja representação textual estava no canal).

Definido na linha 74 do ficheiro ecra.C.

Referências Slang::Cor.

8.3.3.4 Slang::Posicao const & Slang::cursor (Posicao const & *posicao*) [inline]

Manipulador usado para colocar o cursor numa dada posição.

Usar como se segue:

```
// Definição de uma posicão:
Posicao posicao(10, 20);

// Coloca cursor na posição do ecrã definida:
ecra << cursor(posicao);

// Escreve "Olá mundo!" a partir dessa posição:
ecra << "Olá mundo";

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

embora fosse mais simples omitir a utilização do manipulador, inserindo directamente a posição no ecrã:

```
// Definição de uma posicão:
Posicao posicao(10, 20);

// Coloca cursor na posição do ecrã definida:
ecra << posicao;

// Escreve "Olá mundo!" a partir dessa posição:
ecra << "Olá mundo";

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

que ainda pode ser simplificado para:

```
// Coloca cursor na posição do ecrã definida:
ecra << Posicao(10, 20);

// Escreve "Olá mundo!" a partir dessa posição:
ecra << "Olá mundo";

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

Veja também:
Ecra::moveCursorPara()

Definido na linha 1127 do ficheiro ecra_impl.H.

Referenciado por cursor_aleatorio() e main().

8.3.3.5 Slang::Caixa const Slang::caixa (int const *linha*, int const *coluna*, int const *numero_de_linhas*, int const *numero_de_colunas*) [inline]

Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição e com uma dada dimensão.

Usar como se segue:

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

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

Veja também:
Ecra::desenha()

Definido na linha 1132 do ficheiro ecra_impl.H.

Referenciado por Slang::Ecra::operator<<().

8.3.3.6 Slang::Caixa const Slang::caixa (Posicao const & *origem* = Posicao(), Dimensao const & *dimensao* = Dimensao()) [inline]

Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição e com uma dada dimensão.

Usar como se segue:

```
// Definição de uma posição e uma dimensão para a caixa:
Posicao posicao(10, 20);
Dimensao dimensao(5, 15);

// Desenha caixa com a posição e dimensão definidas:
ecra << caixa(posicao, dimensao);

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

Neste caso poder-se-ia ter omitido o manipulador, usando-se em vez dele o construtor da classe **Caixa**, que se pode inserir num ecrã:

```
// Definição de uma posição e uma dimensão para a caixa:
Posicao posicao(10, 20);
Dimensao dimensao(5, 15);

// Desenha caixa com a posição e dimensão definidas:
ecra << Caixa(posicao, dimensao);

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

Veja também:
Ecra::desenha()

Definido na linha 1141 do ficheiro ecra_impl.H.

Referenciado por main().

8.3.3.7 void Slang::fundo (Ecra & *ecra*) [inline]

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;
```

Veja também:
Ecra::mudaObjectoCorEmUsoParaFundo().

Definido na linha 1146 do ficheiro ecra_impl.H.

Referências Slang::ecra.

Referenciado por Slang::Ecra::cola(), Slang::CaixaDeTexto::interage(), Slang::MenuSimples::interage() e main().

8.3.3.8 void Slang::parado (Ecra & *ecra*) [inline]

Manipulador usado para manter o cursor na posição corrente durante a próxima operação de inserção 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;
```

Veja também:
Ecra::impoeImobilidadeDoCursorNaProximaInsercao()

Definido na linha 1151 do ficheiro ecra_impl.H.

Referências Slang::ecra.

Referenciado por main(), procedimento1() e procedimento2().

8.3.3.9 void Slang::sobeCursor (Ecra & *ecra*) [inline]

Manipulador usado para subir o cursor no ecrã.

Usar como se segue:

```
// Coloca cursor a meio do ecrã, à esquerda:
ecra << cursor(ecra.dimensao().numeroDeLinhas() / 2, 0);

// Escreve "mundo!" mantendo a posição do cursor:
ecra << parado << "mundo!";

// Sobe cursor para a linha acima:
ecra << sobeCursor;

// Escreve "Olá" mantendo a posição do cursor:
ecra << parado << "Olá";

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

Veja também:
Ecra::sobeCursor()

Definido na linha 1156 do ficheiro ecra_impl.H.

Referências Slang::ecra.

8.3.3.10 void Slang::baixaCursor (Ecra & *ecra*) [inline]

Manipulador usado para baixar o cursor no ecrã.

Usar como se segue:

```
// Coloca cursor a meio do ecrã, à esquerda:
ecra << cursor(ecra.dimensao().numeroDeLinhas() / 2, 0);

// Escreve "Olá" mantendo a posição do cursor:
ecra << parado << "Olá";

// Baixa o cursor:
ecra << sobeCursor;

// Escreve "mundo!" mantendo a posição do cursor:
ecra << parado << "mundo!";

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

Veja também:
Ecra::baixaCursor()

Definido na linha 1161 do ficheiro ecra_impl.H.

Referências Slang::ecra.

Referenciado por main(), procedimento1() e procedimento2().

8.3.3.11 void Slang::recuaCursor (Ecra & *ecra*) [inline]

Manipulador usado para recuar o cursor no ecrã.

Usa-se como os manipuladores **sobeCursor()** e **baixaCursor()**.

Veja também:
Ecra::recuaCursor()

Definido na linha 1166 do ficheiro ecra_impl.H.

Referências Slang::ecra.

8.3.3.12 void Slang::avancaCursor (Ecra & *ecra*) [inline]

Manipulador usado para avançar o cursor no ecrã.

Usa-se como os manipuladores **sobeCursor()** e **baixaCursor()**.

Veja também:
Ecra::avancaCursor()

Definido na linha 1171 do ficheiro ecra_impl.H.

Referências Slang::ecra.

8.3.3.13 Slang::ManipuladorDaLargura Slang::largura (int const *largura*) [inline]

Manipulador usado para indicar o número de células do ecrã a ocupar na próxima operação de inserção no 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 e centrados em 30
// caracteres:
ecra << oor << ao_centro << largura(30) << "Olá mundo!";
```

Precondição:
0 <= largura

Veja também:
Ecra::mudaLarguraDaProximaOperacaoDeEscritaPara()

Definido na linha 1176 do ficheiro ecra_impl.H.

Referências Slang::largura().

Referenciado por Slang::Ecra::desenhaSegmentoHorizontalCom(), Slang::largura(), main(), Slang::ManipuladorDaLargura::ManipuladorDaLargura() e Slang::segmento_horizontal().

8.3.3.14 Slang::ManipuladorDeDesenhoDeSegmento Slang::segmento_horizontal (int const *largura*) [inline]

Manipulador usado para desenhar um segmento de recta horizontal.

Veja-se um exemplo de utilização no programa abaixo:

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

using namespace Slang;

int main()
{
    // Guardar dimensão do ecrã:
    int const altura = ecra.dimensao().numeroDeLinhas();
    int const largura = ecra.dimensao().numeroDeColunas();

    // Desenhar segmento de recta horizontal a meio do ecrã e com meia largura
    // do ecrã:
    ecra << cursor(altura / 2, largura / 4)
         << segmento_horizontal(largura / 2);

    // Desenhar segmento de recta vertical a meio do ecrã e com meia altura do
    // ecrã:
    ecra << cursor(altura / 4, largura / 2)
         << segmento_vertical(altura / 2);

    // Refescar e esperar por tecla:
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();

    ecra << apaga << refresca;
    teclado.leProximaTeclaDisponivel();

    // O mesmo, mas usando operações da classe Ecra:

    ecra << cursor(altura / 2, largura / 4);
    ecra.desenhaSegmentoHorizontalCom(largura / 2);

    ecra << cursor(altura / 4, largura / 2);
    ecra.desenhaSegmentoVerticalCom(altura / 2);

    // Refescar e esperar por tecla:
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
}
```

Precondição:
0 <= largura

Veja também:
Ecra::desenhaSegmentoHorizontalCom()

Definido na linha 1184 do ficheiro ecra_impl.H.

Referências Slang::largura().

Referenciado por main().

8.3.3.15 Slang::ManipuladorDeDesenhoDeSegmento Slang::segmento_vertical (int const *altura*) [inline]

Manipulador usado para desenhar um segmento de recta vertical.

Veja-se uma exemplo de utilização na documentação do manipulador **segmento_horizontal()**.

Precondição:
0 <= largura

Veja também:
Ecra::desenhaSegmentoVerticalCom()

Definido na linha 1192 do ficheiro ecra_impl.H.

Referenciado por main().

8.3.3.16 void Slang::refresca (Ecra & *ecra*) [inline]

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;
```

Veja também:
Ecra::refresca()

Definido na linha 1199 do ficheiro ecra_impl.H.

Referências Slang::ecra.

Referenciado por Slang::CaixaDeTexto::interage(), Slang::MenuSimples::interage(), main(), procedimento1() e procedimento2().

8.3.3.17 void Slang::refresca_tudo (Ecra & *ecra*) [inline]

Manipulador usado para fazer o refrescamento total do ecrã.

Usado como se segue:

```
ecra << refresca_tudo;
```

Veja também:
Ecra::refrescaTudo()

Definido na linha 1204 do ficheiro ecra_impl.H.

Referências Slang::ecra.

8.3.3.18 void Slang::apaga (Ecra & *ecra*) [inline]

Manipulador usado para apagar o ecrã (ou seja, para o pintar com a cor do fundo).

Usar como se segue:

```
// Apaga o ecrã:
ecra << apaga;

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

Veja também:
Ecra::apaga()

Definido na linha 1209 do ficheiro ecra_impl.H.

Referências Slang::ecra.

Referenciado por main().

8.3.3.19 void Slang::apaga_fim_da_linha (Ecra & *ecra*) [inline]

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;
```

Veja também:
Ecra::apagaDoCursorAteOFimDaLinha()

Definido na linha 1214 do ficheiro ecra_impl.H.

Referências Slang::ecra.

8.3.3.20 void Slang::apaga_fim_do_ecra (Ecra & *ecra*) [inline]

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;
```

Veja também:
Ecra::apagaDoCursorAteOFim()

Definido na linha 1219 do ficheiro ecra_impl.H.

Referências Slang::ecra.

8.3.3.21 void Slang::campainha (Ecra & *ecra*) [inline]

Manipulador usado para fazer soar a campainha.

Usar como se segue:

```
ecra << campainha;
```

Veja também:
Ecra::tocaCampainha()

Definido na linha 1224 do ficheiro ecra_impl.H.

Referências Slang::ecra.

Referenciado por Slang::CaixaDeTexto::interage() e main().

8.3.4 Documentação das variáveis

8.3.4.1 Caixa const Slang::caixa_vazia

Constante representando uma caixa vazia, i.e., com dimensão nula.

Esta caixa tem como origem o canto superior esquerdo do ecrã.

Definido na linha 897 do ficheiro ecra.H.

8.3.4.2 Slang::Ecra Slang::ecra

Uma variável global representando o ecrã.

A ideia é que substitua a variável global cout usada normalmente para escrever no ecrã. É através desta variável que se procede a todas as inserções no ecrã.

Definido na linha 515 do ficheiro ecra.C.

Referenciado por Slang::apaga(), Slang::apaga_fim_da_linha(), Slang::apaga_fim_do_ecra(), Slang::avancaCursor(), Slang::baixaCursor(), Slang::campainha(), cursor_aleatorio(), Slang::fundo(), Slang::CaixaDeTexto::interage(), Slang::MenuSimples::interage(), main(), Slang::Ecra::ObjectoCor::mudaCorDoFundoPara(), Slang::Ecra::ObjectoCor::mudaCorDoTextoPara(), Slang::Ecra::ObjectoCor::ObjectoCor(), Slang::operator<<(), Slang::Ecra::operator<<(), Slang::parado(), procedimento1(), procedimento2(), Slang::recuaCursor(), Slang::refresca(), Slang::refresca_tudo(), Slang::sobeCursor() e Slang::Ecra::ObjectoCor::~ObjectoCor().

8.3.4.3 string const Slang::nomes_das_cores

Valor inicial:

```
{
   "preto",
   "cinza",
   "vermelho",
   "vermelho-brilhante",
   "verde",
   "verde-brilhante",
```

```
    "castanho",
    "amarelo",
    "azul",
    "azul-brilhante",
    "magenta",
    "magenta-brilhante",
    "ciano",
    "ciano-brilhante",
    "cinzento-claro",
    "branco"
}
```

Constante global com os nomes das cores.

Indexável com os valores do tipo enumerado Cor.

Definido na linha 517 do ficheiro ecra.C.

Referenciado por Slang::MenuDeCores::MenuDeCores().

8.3.4.4 Caixa const Slang::caixa_universal

Constante representando a caixa universal (enche todo o espaço).

Definido na linha 905 do ficheiro ecra.H.

8.3.4.5 int const Slang::numero_de_cores

Constante que guarda o número total de cores.

Definido na linha 966 do ficheiro ecra.H.

Referenciado por Slang::MenuDeCores::MenuDeCores().

8.4 Ferramentas de menus

Módulo correspondente ao ficheiro Slang++/menu.H.

Ficheiros

- file **exemplo2.C**

  Programa que demostra a utilização do ecrã e dos menus de forma simples.

- file **menu1.C**

  Programa que demostra a utilização de menus simples.

- file **menu.C**

  Ficheiro de implementação do módulo menus.

- file **menu.H**

  Ficheiro de interface do módulo menus.

- file **menu_impl.H**

  Ficheiro auxiliar de implementação do módulo menus.

Componentes

- class **ApendiceComCor**

  Define ferramentas relacionadas com cores para os menus que as possuirem.

- class **Menu**

  Classe abstracta que representa a interface básica de todos os menus.

- class **MenuComCor**

  Classe abstracta que define a interface básica de todos os menus com cores.

- class **MenuSimples**

  Representa menus simples, que consistem numa sequência de itens.

- class **MenuDeCores**

  Representa menus de selecção das cores básicas usáveis no ecrã.

- class **MenuDeSimOuNao**

  Representa menus com apenas duas opções: sim e não.

- class **Aviso**

  Representa caixas de aviso, que apenas mostram uma mensagem e das quais se sai pressionando 'enter'.

- class **CaixaDeTexto**

  Representa caixas de texto, que quando executadas permitem ao utilizador introduzir uma cadeia de caracteres.

8.4.1 Descrição detalhada

Módulo correspondente ao ficheiro Slang++/menu.H.

Contém ferramentas para criar e executar vários tipos de menus e caixas de diálogo simples.

Os vários tipos de menus são exemplificados no programa abaixo:

```
#include <cstdlib> // para atoi().
#include <string>  // para string.

#include <Slang++/slang.H>

using namespace std;

using namespace Slang;

int main()
{
    // A definição dos menus e caixas de texto fora do ciclo leva a que estes
    // tenham memória, preservando as escolhas do utilizador entre interacções
    // indenpendentes:
    MenuSimples menu("Um menu que não faz nada...",
                     "Perguntar se sim ou não\n"
                     "Pedir uma cor\n"
                     "Abrir caixa para inserir texto\n"
                     "Abrir caixa para inserir número\n"
                     "Sair");
    MenuDeSimOuNao menu_de_sim_ou_nao("Responda sim ou não:");
    MenuDeCores menu_de_cores("Escolha uma cor");
    CaixaDeTexto caixa_para_texto("Escreva um texto:");
    CaixaDeTexto caixa_para_numeros("Escreva um número:", "", "0123456789",
                                    true);

    do {
        menu.interage();

        Posicao const posicao_original_do_cursor = ecra.posicaoDoCursor();

        switch(menu.opcaoActual()) {
          case 0: {
              menu_de_sim_ou_nao.interage();

              ecra << "Escolheu: ";
              if(menu_de_sim_ou_nao.opcaoActual() == true)
                  ecra << "Sim";
              else
                  ecra << "Não";

              break;
          }
          case 1: {
              menu_de_cores.interage();

              ecra << "Escolheu: " << Cor(menu_de_cores.opcaoActual());

              break;
          }
          case 2: {
              caixa_para_texto.interage();

              ecra << "Texto: " << caixa_para_texto.textoActual();

              break;
          }
```

```
          case 3: {
              caixa_para_numeros.interage();

              ecra << "Número: "
                   << atoi(caixa_para_numeros.textoActual().c_str());

              break;
          }
        }
        ecra << posicao_original_do_cursor << baixaCursor << refresca;
    } while(menu.opcaoActual() != 4);

    Aviso("Vou terminando...").interage();
}
```

8.5 Ferramentas de teclado

Módulo correspondente ao ficheiro Slang++/teclado.H.

Ficheiros

- file **exemplo1.C**

  Programa que demostra a utilização simultânea do teclado e do ecrã de forma simples.

- file **teclado1.C**

  Programa que demostra a utilização do teclado de forma simples.

- file **teclado.H**

  Ficheiro de interface do módulo teclados.

- file **teclado_impl.H**

  Ficheiro auxiliar de implementação do módulo teclados.

- file **teclado.C**

  Ficheiro de implementação do módulo teclados.

Componentes

- class **Tecla**

  Representa teclas premidas.

- class **Teclado**

  Representa o teclado.

Variáveis

- Teclado **teclado**

  Representa o teclado.

8.5.1 Descrição detalhada

Módulo correspondente ao ficheiro Slang++/teclado.H.

Contém ferramentas para lidar com o teclado.

8.5.2 Documentação das variáveis

8.5.2.1 Slang::Teclado Slang::teclado

Representa o teclado.

Substitui a variável global #cin usada normalmente para ler do teclado.

Definido na linha 32 do ficheiro teclado.C.

Referenciado por Slang::CaixaDeTexto::interage(), Slang::MenuSimples::interage(), main(), procedimento1() e procedimento2().

8.6 Ferramentas complementares para cadeias de caracteres

Módulo correspondente ao ficheiro Utilitarios/cadeia.H.

Ficheiros

- file **cadeia.H**

  Ficheiro de interface do módulo cadeias.

- file **cadeia_impl.H**

  Ficheiro auxiliar de implementação do módulo cadeias.

8.6.1 Descrição detalhada

Módulo correspondente ao ficheiro Utilitarios/cadeia.H.

Contém ferramentas para somar valores inteiros a cadeias de caracteres.

8.7 Ferramentas de data e tempo

Módulo correspondente ao ficheiro Utilitarios/data.H.

Ficheiros

- file **data.C**

  Ficheiro de implementação do módulo datas.

- file **data1.C**

  Programa que demostra a utilização de datas e tempos de forma simples.

- file **data.H**

  Ficheiro de interface do módulo datas.

- file **data_impl.H**

  Ficheiro auxiliar de implementação do módulo datas.

Componentes

- class **Data**

  Representa datas posteriores a 1582, i.e., datas no calendário Gregoriano, adoptado por Portugal e outros países católicos em 1582.

Tipos básicos para datas e tempos

- typedef long **Duracao**

  Representa durações em dias.

- typedef int **Dia**

  Representa o dia do mês de uma data.

- typedef int **Ano**

  Representa o ano de uma data.

- enum **Mes** { **janeiro** = 1, **fevereiro**, **marco**, **abril**, **maio**, **junho**, **julho**, **agosto**, **setembro**, **outubro**, **novembro**, **dezembro** }

  Representa os meses do ano.

- enum **DiaDaSemana** { **domingo** = 0, **segunda_feira**, **terca_feira**, **quarta_feira**, **quinta_feira**, **sexta_feira**, **sabado** }

  Representa os dias da semana.

- int const **numero_total_de_meses** = 12

  A número de meses no ano.

- std::string **nomes_dos_meses** [**numero_total_de_meses**+1]

Os nomes dos meses do ano.

- int const **numero_total_de_dias_da_semana** = 7

  O número de dias na semana.

- std::string **nomes_dos_dias_da_semana** [**numero_total_de_dias_da_semana**]

  Os nomes dos dias da semana.

Operações associadas ao tipo Mes

- **Mes** & **operator++** (**Mes** &mes)

  Incrementação prefixa de meses.

- **Mes** & **operator–** (**Mes** &mes)

  Decrementação prefixa de meses.

- **Mes operator++** (**Mes** &mes, int)

  Incrementação sufixa de meses.

- **Mes operator–** (**Mes** &mes, int)

  Decrementação sufixa de meses.

- **Mes** & **operator+=** (**Mes** &mes, int const numero_de_meses)

  Avança mês de um número de meses.

- **Mes** & **operator-=** (**Mes** &mes, int const numero_de_meses)

  Recua mês de um número de meses.

- **Mes operator+** (**Mes** const mes, int const numero_de_meses)

  Devolve a adição de um mês com um dado número de meses.

- **Mes operator+** (int const numero_de_meses, **Mes** const mes)

  Devolve a adição de um mês com um dado número de meses.

- int **operator-** (**Mes** const um_mes, **Mes** const outro_mes)

  Devolve a distância em número de meses entre dois meses.

- **Mes operator-** (**Mes** const mes, int const numero_de_meses)

  Devolve a subtracção de um mês de um dado número de meses.

- std::ostream & **operator<<** (std::ostream &saida, **Mes** const mes)

  Insere um mês num canal.

- std::istream & **operator>>** (std::istream &entrada, **Mes** &mes)

  Extrai um mês de um canal.

Operações associadas ao tipo DiaDaSemana

- **DiaDaSemana** & **operator++** (**DiaDaSemana** &dia_da_semana)

  Incrementação prefixa de dias da semana.

- **DiaDaSemana** & **operator–** (**DiaDaSemana** &dia_da_semana)

  Decrementação prefixa de dias da semana.

- **DiaDaSemana operator++** (**DiaDaSemana** &dia_da_semana, int)

  Incrementação sufixa de dias da semana.

- **DiaDaSemana operator–** (**DiaDaSemana** &dia_da_semana, int)

  Incrementação sufixa de dias da semana.

- **DiaDaSemana** & **operator+=** (**DiaDaSemana** &dia_da_semana, int const numero_de_dias)

  Avança dia da semana de um número de dias da semana.

- **DiaDaSemana** & **operator-=** (**DiaDaSemana** &dia_da_semana, int const numero_de_dias)

  Recua dia da semana de um número de dias da semana.

- **DiaDaSemana operator+** (**DiaDaSemana** const dia_da_semana, int const numero_de_dias)

  Devolve a adição de um dia da semana com um dado número de dias da semana.

- **DiaDaSemana operator+** (int const numero_de_dias, **DiaDaSemana** const dia_da_semana)

  Devolve a adição de um dia da semana com um dado número de dias da semana.

- int **operator-** (**DiaDaSemana** const um_dia_da_semana, **DiaDaSemana** const outro_dia_da_semana)

  Devolve a distância em número de dias da semana entre dois dias da semana.

- **DiaDaSemana operator-** (**DiaDaSemana** const dia_da_semana, int const numero_de_dias)

  Devolve a subtracção de um dia da semana de um dado número de dias da semana.

- std::ostream & **operator<<** (std::ostream &saida, **DiaDaSemana** const dia_da_semana)

  Insere um dia da semana num canal.

- std::istream & **operator>>** (std::istream &entrada, **DiaDaSemana** &dia_da_semana)

  Extrai um dia da semana de um canal.

Operações associadas à classe Data

- bool **operator==** (Data const &uma_data, Data const &outra_data)

  Indica se duas datas são iguais.

- bool **operator!=** (Data const &uma_data, Data const &outra_data)

  Indica se duas datas são diferentes.

- bool **operator<** (Data const &uma_data, Data const &outra_data)

  Indica se uma data é menor que outra.

- bool **operator>** (Data const &uma_data, Data const &outra_data)

  Indica se uma data é maior que outra.

- bool **operator<=** (Data const &uma_data, Data const &outra_data)

  Indica se uma data é menor ou igual a outra.

- bool **operator>=** (Data const &uma_data, Data const &outra_data)

  Indica se uma data é maior ou igual a outra..

- **Duracao operator-** (Data const &uma_data, Data const &outra_data)

  Devolve a distância entre duas datas.

- Data **operator-** (Data const &data, **Duracao** const &duracao)

  Devolve a subtracção de uma duração de uma data.

- Data **operator+** (Data const &data, **Duracao** const &duracao)

  Devolve a adição de uma duração a uma data.

- Data **operator+** (**Duracao** const &duracao, Data const &data)

  Devolve a adição de uma duração a uma data.

- std::ostream & **operator<<** (std::ostream &saida, Data const &data)

  Insere uma data num canal.

- std::istream & **operator>>** (std::istream &entrada, Data &data)

  Extrai uma data de um canal.

8.7.1 Descrição detalhada

Módulo correspondente ao ficheiro Utilitarios/data.H.

Contém ferramentas para manipular data e tempo.

8.7.2 Documentação dos tipos

8.7.2.1 typedef long Utilitarios::Duracao

Representa durações em dias.

As durações podem ser positivas ou negativas. Não são representáveis durações superiores ao maior dos long int nem inferiores ao menor dos long int. Para resolver este problema (se ele for relevante), tem de se mudar o tipo básico para outro com maior gama ou para uma classe desenvolvida para o efeito.

Definido na linha 31 do ficheiro data.H.

Referenciado por Utilitarios::operator+(), Utilitarios::Data::operator+=(), Utilitarios::operator-() e Utilitarios::Data::operator-=().

8.7.2.2 typedef int Utilitarios::Dia

Representa o dia do mês de uma data.

Definido na linha 36 do ficheiro data.H.

Referenciado por Utilitarios::Data::Data() e Utilitarios::operator>>().

8.7.2.3 typedef int Utilitarios::Ano

Representa o ano de uma data.

Definido na linha 41 do ficheiro data.H.

Referenciado por Utilitarios::Data::Data(), Utilitarios::eBissexto(), Utilitarios::numeroDeDiasEm() e Utilitarios::operator>>().

8.7.3 Documentação dos valores da enumeração

8.7.3.1 enum Utilitarios::Mes

Representa os meses do ano.

Valores da enumeração:

janeiro

fevereiro

marco

abril

maio

junho

julho

agosto

setembro

outubro

novembro

dezembro

Definido na linha 46 do ficheiro data.H.

Referenciado por Utilitarios::Data::actual(), Utilitarios::Data::Data(), Utilitarios::numeroDeDiasEm(), Utilitarios::operator+(), Utilitarios::operator++(), Utilitarios::operator+=(), Utilitarios::operator-(), Utilitarios::operator–(), Utilitarios::operator-=(), Utilitarios::operator<<() e Utilitarios::operator>>().

8.7.3.2 enum Utilitarios::DiaDaSemana

Representa os dias da semana.

Valores da enumeração:

domingo

segunda_feira

terca_feira

quarta_feira

quinta_feira

sexta_feira

sabado

Definido na linha 75 do ficheiro data.H.

Referenciado por Utilitarios::Data::diaDaSemana(), Utilitarios::operator+(), Utilitarios::operator++(), Utilitarios::operator+=(), Utilitarios::operator-(), Utilitarios::operator–(), Utilitarios::operator-=(), Utilitarios::operator<<() e Utilitarios::operator>>().

8.7.4 Documentação das funções

8.7.4.1 Utilitarios::Mes & Utilitarios::operator++ (Mes & *mes*) [inline]

Incrementação prefixa de meses.

Precondição:
mes = m.

Poscondição:
operator++ := mes = mês após m, passando-se de Dezembro a Janeiro.

Definido na linha 46 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.2 Utilitarios::Mes & Utilitarios::operator– (Mes & *mes*) [inline]

Decrementação prefixa de meses.

Precondição:
mes = m.

Poscondição:
operator– := mes = mês antes de m, passando-se de Janeiro a Dezembro.

Definido na linha 58 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.3 Utilitarios::Mes Utilitarios::operator++ (Mes & *mes*, int) [inline]

Incrementação sufixa de meses.

Precondição:
mes = m.

Poscondição:
operator++ = m e mes = mês após m, passando-se de Dezembro a Janeiro.

Definido na linha 70 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.4 Utilitarios::Mes Utilitarios::operator– (Mes & *mes*, int) [inline]

Decrementação sufixa de meses.

Precondição:
mes = m.

Poscondição:
operator– = m e mes = mês antes de m, passando-se de Janeiro a Dezembro.

Definido na linha 80 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.5 Utilitarios::Mes & Utilitarios::operator+= (Mes & *mes*, int const *numero_de_meses*) [inline]

Avança mês de um número de meses.

Precondição:
mes = m.

Poscondição:
operator+= := mes = mês numero_de_meses após m, passando-se de Dezembro a Janeiro.

Definido na linha 90 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro, Utilitarios::Mes e Utilitarios::numero_total_de_meses.

8.7.4.6 Utilitarios::Mes & Utilitarios::operator-= (Mes & *mes*, int const *numero_de_meses*) [inline]

Recua mês de um número de meses.

Precondição:
mes = m.

Poscondição:
operator-= := mes = mês numero_de_meses antes de m, passando-se de Janeiro a Dezembro.

Definido na linha 104 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.7 Utilitarios::Mes Utilitarios::operator+ (Mes const *mes*, int const *numero_de_meses*) [inline]

Devolve a adição de um mês com um dado número de meses.

Precondição:
V.

Poscondição:
operator+ = mês numero_de_meses após mes, passando-se de Dezembro a Janeiro.

Definido na linha 114 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.8 Utilitarios::Mes Utilitarios::operator+ (int const *numero_de_meses*, Mes const *mes*) [inline]

Devolve a adição de um mês com um dado número de meses.

Precondição:
V.

Poscondição:
operator+ = mês numero_de_meses após mes, passando-se de Dezembro a Janeiro.

Definido na linha 124 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.9 int Utilitarios::operator- (Mes const *um_mes*, Mes const *outro_mes*) [inline]

Devolve a distância em número de meses entre dois meses.

Precondição:
V.

Poscondição:
operator- = numero_de_meses entre outro_mes e um_mes.

Definido na linha 132 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro, Utilitarios::Mes e Utilitarios::numero_total_de_meses.

8.7.4.10 Utilitarios::Mes Utilitarios::operator- (Mes const *mes*, int const *numero_de_meses*) [inline]

Devolve a subtracção de um mês de um dado número de meses.

Precondição:
V.

Poscondição:
operator- = mês numero_de_meses antes de mes, passando-se de Janeiro a Dezembro.

Definido na linha 143 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro e Utilitarios::Mes.

8.7.4.11 std::ostream & Utilitarios::operator<< (std::ostream & *saida*, Mes const *mes*) [inline]

Insere um mês num canal.

Precondição:
V.

Poscondição:
saida contém o mês (formato textual) ou não saida.good().

Definido na linha 151 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::janeiro, Utilitarios::Mes e Utilitarios::nomes_dos_meses.

8.7.4.12 istream & Utilitarios::operator>> (std::istream & *entrada*, Mes & *mes*)

Extrai um mês de um canal.

Precondição:
V.

Poscondição:
entrada.good() e já não contém o mês, que está em *mes*. Alternativamente, não entrada.good()..

Definido na linha 41 do ficheiro data.C.

Referências Utilitarios::Mes, Utilitarios::nomes_dos_meses e Utilitarios::numero_total_de_meses.

8.7.4.13 Utilitarios::DiaDaSemana & Utilitarios::operator++ (DiaDaSemana & *dia_da_semana*) [inline]

Incrementação prefixa de dias da semana.

Precondição:
dia_de_semana = d.

Poscondição:
operator++ := dia_de_semana = dia da semana após d, passando-se de sabado a domingo.

Definido na linha 163 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.14 Utilitarios::DiaDaSemana & Utilitarios::operator– (DiaDaSemana & *dia_da_semana*) [inline]

Decrementação prefixa de dias da semana.

Precondição:
dia_de_semana = d.

Poscondição:
operator– := dia_de_semana = dia da semana antes de d, passando-se de domingo a sabado.

Definido na linha 176 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.15 Utilitarios::DiaDaSemana Utilitarios::operator++ (DiaDaSemana & *dia_da_semana*, int) [inline]

Incrementação sufixa de dias da semana.

Precondição:
dia_de_semana = d.

Poscondição:
operator++ = d e dia_de_semana = dia da semana após d, passando-se de sabado a domingo.

Definido na linha 189 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.16 Utilitarios::DiaDaSemana Utilitarios::operator– (DiaDaSemana & *dia_da_semana*, int) [inline]

Incrementação sufixa de dias da semana.

Precondição:
dia_de_semana = d.

Poscondição:
operator++ = d e dia_de_semana = dia da semana após d, passando-se de sabado a domingo.

Definido na linha 201 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.17 Utilitarios::DiaDaSemana & Utilitarios::operator+= (DiaDaSemana & *dia_da_semana*, int const *numero_de_dias*) [inline]

Avança dia da semana de um número de dias da semana.

Precondição:
dia_de_semana = d.

Poscondição:
operator+= := dia_de_semana = dia da semana numero_de_dias_de_semana após d, passando-se de sabado a domingo.

Definido na linha 213 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo, Utilitarios::numero_total_de_dias_da_semana e Utilitarios::sabado.

8.7.4.18 Utilitarios::DiaDaSemana & Utilitarios::operator-= (DiaDaSemana & *dia_da_semana*, int const *numero_de_dias*) [inline]

Recua dia da semana de um número de dias da semana.

Precondição:
dia_de_semana = d.

Poscondição:
operator-= := dia_de_semana = dia da semana numero_de_dias_de_semana antes de d, passando-se de domingo a sabado.

Definido na linha 229 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.19 Utilitarios::DiaDaSemana Utilitarios::operator+ (DiaDaSemana const *dia_da_semana*, int const *numero_de_dias*) [inline]

Devolve a adição de um dia da semana com um dado número de dias da semana.

Precondição:
V.

Poscondição:
operator+ = dia da semana numero_de_dias_de_semana após dia_de_semana, passando-se de sabado a domingo.

Definido na linha 238 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.20 Utilitarios::DiaDaSemana Utilitarios::operator+ (int const *numero_de_dias*, DiaDaSemana const *dia_da_semana*) [inline]

Devolve a adição de um dia da semana com um dado número de dias da semana.

Precondição:
V.

Poscondição:
operator+ = dia da semana numero_de_dias_de_semana após dia_de_semana, passando-se de sabado a domingo.

Definido na linha 249 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.21 int Utilitarios::operator- (DiaDaSemana const *um_dia_da_semana*, DiaDaSemana const *outro_dia_da_semana*) [inline]

Devolve a distância em número de dias da semana entre dois dias da semana.

Precondição:
V.

Poscondição:
operator- = numero_de_dias_de_semana entre outro_dia_de_semana e um_dia_de_semana.

Definido na linha 257 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo, Utilitarios::numero_total_de_dias_da_semana e Utilitarios::sabado.

8.7.4.22 Utilitarios::DiaDaSemana Utilitarios::operator- (DiaDaSemana const *dia_da_semana*, int const *numero_de_dias*) [inline]

Devolve a subtracção de um dia da semana de um dado número de dias da semana.

Precondição:
V.

Poscondição:
operator- = dia da semana numero_de_dias_de_semana antes de dia_de_semana, passando-se de domingo a sabado.

Definido na linha 271 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo e Utilitarios::sabado.

8.7.4.23 std::ostream & Utilitarios::operator<< (std::ostream & *saida*, DiaDaSemana const *dia_da_semana*) [inline]

Insere um dia da semana num canal.

Precondição:
V.

Poscondição:
saida contém o dia da semana (formato textual) ou não saida.good().

Definido na linha 279 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, Utilitarios::domingo, Utilitarios::nomes_dos_dias_da_semana e Utilitarios::sabado.

8.7.4.24 std::istream & Utilitarios::operator>> (std::istream & *entrada*, DiaDaSemana & *dia_da_semana*)

Extrai um dia da semana de um canal.

Precondição:
V.

Poscondição:
entrada.good() e já não contém o dia da semana, que está em *dia_de_semana.* Alternativamente, não entrada.good()..

Definido na linha 58 do ficheiro data.C.

Referências Utilitarios::DiaDaSemana, Utilitarios::nomes_dos_dias_da_semana e Utilitarios::numero_total_de_dias_da_semana.

8.7.4.25 bool Utilitarios::operator== (Data const & *uma_data*, Data const & *outra_data*) [inline]

Indica se duas datas são iguais.

Precondição:
V.

Poscondição:
operador== = *uma_data* é igual a *outra_data.*

Definido na linha 518 do ficheiro data_impl.H.

8.7.4.26 bool Utilitarios::operator!= (Data const & *uma_data*, Data const & *outra_data*) [inline]

Indica se duas datas são diferentes.

Precondição:
V.

Poscondição:
operador!= = *uma_data* é diferente de *outra_data.*

Definido na linha 526 do ficheiro data_impl.H.

8.7.4.27 bool Utilitarios::operator< (Data const & *uma_data*, Data const & *outra_data*) [inline]

Indica se uma data é menor que outra.

Precondição:
V.

Poscondição:
operador< = *uma_data* é menor que *outra_data.*

Definido na linha 532 do ficheiro data_impl.H.

8.7.4.28 bool Utilitarios::operator> (Data const & *uma_data*, Data const & *outra_data*) [inline]

Indica se uma data é maior que outra.

Precondição:
V.

Poscondição:
operador>= = *uma_data* é maior uqe *outra_data.*

Definido na linha 538 do ficheiro data_impl.H.

8.7.4.29 bool Utilitarios::operator<= (Data const & *uma_data*, Data const & *outra_data*) [inline]

Indica se uma data é menor ou igual a outra.

Precondição:
V.

Poscondição:
operador<= = *uma_data* é menor ou igual a *outra_data.*

Definido na linha 544 do ficheiro data_impl.H.

8.7.4.30 bool Utilitarios::operator>= (Data const & *uma_data*, Data const & *outra_data*) [inline]

Indica se uma data é maior ou igual a outra..

Precondição:
V.

Poscondição:
operador>= = *uma_data* é maior ou igual igual a *outra_data.*

Definido na linha 550 do ficheiro data_impl.H.

8.7.4.31 Utilitarios::Duracao Utilitarios::operator- (Data const & *uma_data*, Data const & *outra_data*) [inline]

Devolve a distância entre duas datas.

Precondição:
V.

Poscondição:
operador- = distância entre *outra_data* e *uma_data,* em dias.

Definido na linha 556 do ficheiro data_impl.H.

8.7.4.32 Utilitarios::Data Utilitarios::operator- (Data const & *data*, Duracao const & *duracao*) [inline]

Devolve a subtracção de uma duração de uma data.

Precondição:
duracao <= 0 ou Data(ano_minimo, 1, 1) + d <= data..

Poscondição:
operador- + duracao = *data.*

Definido na linha 562 do ficheiro data_impl.H.

Referências Utilitarios::Duracao e Utilitarios::janeiro.

8.7.4.33 Utilitarios::Data Utilitarios::operator+ (Data const & *data*, Duracao const & *duracao*) [inline]

Devolve a adição de uma duração a uma data.

Precondição:
0 <= duracao ou Data(ano_minimo, 1, 1) - d <= data.

Poscondição:
operador- = *data* acrescentada de *duracao.*

Definido na linha 571 do ficheiro data_impl.H.

Referências Utilitarios::Duracao e Utilitarios::janeiro.

8.7.4.34 Utilitarios::Data Utilitarios::operator+ (Duracao const & *duracao*, Data const & *data*) [inline]

Devolve a adição de uma duração a uma data.

Precondição:
0 <= duracao ou Data(ano_minimo, 1, 1) - d <= data.

Poscondição:
operador+ = *data* acrescentada de *duracao.*

Definido na linha 580 do ficheiro data_impl.H.

Referências Utilitarios::Duracao e Utilitarios::janeiro.

8.7.4.35 std::ostream & Utilitarios::operator<< (std::ostream & *saida*, Data const & *data*) [inline]

Insere uma data num canal.

Precondição:
V.

Poscondição:
saida contém a data (a/m/d) ou não saida.good().

Definido na linha 589 do ficheiro data_impl.H.

8.7.4.36 std::istream & Utilitarios::operator>> (std::istream & *entrada*, Data & *data*)

Extrai uma data de um canal.

Precondição:
V.

Poscondição:
entrada.good() e já não contém a data, que está em *data.* Alternativamente, não entrada.good()..

Definido na linha 115 do ficheiro data.C.

Referências Utilitarios::Ano, Utilitarios::Dia, Utilitarios::Mes, Utilitarios::numero_total_de_meses e Utilitarios::numeroDeDiasEm().

8.7.5 Documentação das variáveis

8.7.5.1 int const Utilitarios::numero_total_de_meses = 12

A número de meses no ano.

Definido na linha 64 do ficheiro data.H.

Referenciado por Utilitarios::numeroDeDiasEm(), Utilitarios::operator+=(), Utilitarios::operator-() e Utilitarios::operator>>().

8.7.5.2 string Utilitarios::nomes_dos_meses

Valor inicial:

```
{
    "",
    "Janeiro",
    "Fevereiro",
    "Março",
    "Abril",
    "Maio",
    "Junho",
    "Julho",
    "Agosto",
    "Setembro",
    "Outubro",
    "Novembro",
    "Dezembro"
}
```

Os nomes dos meses do ano.

Indexável com o os valores enumerados do tipo Mes.

Definido na linha 13 do ficheiro data.C.

Referenciado por Utilitarios::operator<<() e Utilitarios::operator>>().

8.7.5.3 int const Utilitarios::numero_total_de_dias_da_semana = 7

O número de dias na semana.

Definido na linha 88 do ficheiro data.H.

Referenciado por Utilitarios::Data::diaDaSemana(), Utilitarios::operator+=(), Utilitarios::operator-() e Utilitarios::operator>>().

8.7.5.4 string Utilitarios::nomes_dos_dias_da_semana

Valor inicial:

```
{
    "Domingo",
    "segunda-feira",
    "terça-feira",
    "quarta-feira",
    "quinta-feira",
    "sexta-feira",
    "Sábado"
}
```

Os nomes dos dias da semana.

Indexável com o os valores enumerados do tipo DiaDaSemana.

Definido na linha 30 do ficheiro data.C.

Referenciado por Utilitarios::operator<<() e Utilitarios::operator>>().

8.8 Ignoradores

Módulo correspondente ao ficheiro Utilitarios/ignoradores.H.

Ficheiros

- file **ignoradores.C**

  Ficheiro de implementação do módulo ignoradores.

- file **ignoradores.H**

  Ficheiro de interface do módulo ignoradores.

- file **ignoradores_impl.H**

  Ficheiro auxiliar de implementação do módulo ignoradores.

Componentes

- class **Ignorador**

  Representa manipuladores de canais de entrada (istream) que descartam caracteres.

Manipuladores extra para std::istream

- Ignorador const **il** ('\n')

  Manipulador que descarta todos os caracteres até ao próximo fim-de-linha ('\n').

- Ignorador const **ill** ('\n', true)

  Manipulador que descarta todos os caracteres até ao próximo fim-de-linha ('\n'), mas primeiro limpa possíveis condições de erro do canal.

Funções

- std::istream & **operator**>> (std::istream &entrada, Ignorador const &ignorador)

  Operador de extracção para os manipuladores de ignorar/descartar caracteres.

8.8.1 Descrição detalhada

Módulo correspondente ao ficheiro Utilitarios/ignoradores.H.

Contém manipuladores para ignorar caracteres de um canal.

8.8.2 Documentação das funções

8.8.2.1 istream & Utilitarios::operator>> (std::istream & *entrada*, Ignorador const & *ignorador*)

Operador de extracção para os manipuladores de ignorar/descartar caracteres.

É usado com os manipuladores pré-definidos **il** e **ill**.

Precondição:
V.

Poscondição:
O canal #entrada foi esvaziado dos caracteres disponíveis até ao primeiro caractere ignorador.terminador, tendo as suas possíveis condições de erro sido limpas no caso de ignorador.deve_limpar_o_canal ser verdadeira.

Definido na linha 11 do ficheiro ignoradores.C.

8.8.2.2 Ignorador const il ('\n')

Manipulador que descarta todos os caracteres até ao próximo fim-de-linha ('\n').

O nome é uma abreviatura de "ignora linha".

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?)

Referenciado por Utilitarios::Data::Data().

8.8.2.3 Ignorador const ill ('\n', true)

Manipulador que descarta 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".

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:

1. Se a extracção tiver sucesso, não garante que o valor lido é não-negativo.
2. 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(0 <= numero)
        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: ";
    cin >> numero >> il;

    if(cin.good() and numero >= 0)
        break;

    if(cin.good())
        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.

Veja também:
il

Referenciado por Utilitarios::Data::actual().

8.9 Localização para português

Módulo correspondente ao ficheiro Utilitarios/localizacao.H.

Ficheiros

- file **localizacao.C**

  Ficheiro de implementação do módulo localizacao.

- file **localizacao.H**

  Ficheiro de interface do módulo localizacao.

- file **localizacao_impl.H**

  Ficheiro auxiliar de implementação do módulo localizacao.

Ferramentas de localização para português.

- template<typename C> bool **eImprimivel** (C const caractere)

  Indica se o caractere recebido como argumento é imprimível de acordo com as convenções portuguesas.

- template<typename C> bool **eImprimivel** (std::basic_string< C > const &cadeia)

  Indica se a cadeia de caracteres recebida como argumento é imprimível de acordo com as convenções portuguesas.

8.9.1 Descrição detalhada

Módulo correspondente ao ficheiro Utilitarios/localizacao.H.

Contém ferramentas para localização para português.

8.9.2 Documentação das funções

8.9.2.1 template<typename C> bool Utilitarios::eImprimivel (C const *caractere*) [inline]

Indica se o caractere recebido como argumento é imprimível de acordo com as convenções portuguesas.

Bug

Se a localização para portugal não estiver disponível na máquina, esta função limita-se a devolver verdadeiro!

Precondição:

V.

Poscondição:

eImprimivel = caractere é imprimível de acordo com as convenções portuguesas.

Definido na linha 9 do ficheiro localizacao_impl.H.

Referenciado por Slang::CaixaDeTexto::interage().

8.9.2.2 template<typename C> bool Utilitarios::eImprimivel (std::basic_string< C > const & *cadeia*) [inline]

Indica se a cadeia de caracteres recebida como argumento é imprimível de acordo com as convenções portuguesas.

Bug

Se a localização para portugal não estiver disponível na máquina, esta função limita-se a devolver verdadeiro!

Precondição:

V.

Poscondição:

eImprimivel = cadeia é imprimível de acordo com as convenções portuguesas.

Definido na linha 23 do ficheiro localizacao_impl.H.

Capítulo 9

Pacotes Documentação dos namespaces

9.1 Referência ao namespace Erros

Este espaço nominativo envolve todas as ferramentas da biblioteca `Erros`.

Componentes

- class **Erro**

  Base de uma pequena hierarquia de classes representando excepções.

- class **ErroAoCarregar**

  Representa excepções ocorridas durante o carregamento de objectos a partir de canais.

- class **ErroAoGuardar**

  Representa excepções ocorridas durante o armazenamento de objectos usando canais.

9.1.1 Descrição detalhada

Este espaço nominativo envolve todas as ferramentas da biblioteca `Erros`.

Em particular envolve o módulo erros. A biblioteca `Erros` define algumas classes representando erros genéricos e erros associados ao armazenamento e carregamento de dados em e a partir de canais.

Os ficheiros fonte devem incluir o ficheiro de interface Erros/erros.H.

9.2 Referência ao namespace IPC

Este espaço nominativo envolve todas as ferramentas da biblioteca `IPC++`.

Componentes

- class **Erro**

  Representa as excepções da biblioteca IPC++.

- class **Mensageiro**

  Classe cuja única instância (em cada processo) representa um mensageiro entre o processo em causa e um outro na mesma máquina.

9.2.1 Descrição detalhada

Este espaço nominativo envolve todas as ferramentas da biblioteca `IPC++`.

Em particular envolve o módulo mensageiro. A biblioteca `IPC++` permite a utilização simplificada de comunicações entre processos (IPC, de Inter Process Communications). A biblioteca consiste de um pacote IPC++ (representado pelo espaço nominativo IPC) consistindo, para já, apenas no módulo fisico mensageiros e no módulo físico erros, com os correspondentes ficheiros de interface IPC++/mensageiro.H e IPC++/erros.H.

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

```
c++ opções_de_compilação -o teste_ipc teste_ipc.C -lIPC++ -lUtilitarios
```

A opção `-lIPC++` destina-se a fundir o ficheiro objecto de `teste_ipc` com a biblioteca `IPC++`. A opção `-lUtilitarios` destina-se a fundir o ficheiro objecto com a biblioteca **Utilitarios,** da qual a biblioteca `IPC++` depende.

Os ficheiros fonte devem incluir o ficheiro de interface IPC++/ipc.H, que por sua vez inclui os ficheiros de interface dos módulos físicos, ou incluir apenas o ficheiro de interface do módulo pretendido.

9.3 Referência ao namespace Slang

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

Componentes

- class **Posicao**

  Representa a posição de uma célula do ecrã.

- class **Dimensao**

  Representa uma dimensão em células do ecrã.

- class **Caixa**

  Representa caixas no ecrã.

- class **Ecra**

  Representa o ecrã.

- 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).

- class **Troco**

  Esta classe serve para representar um troço de ecrã.

- struct **ManipuladorDaLargura**

  Não usar directamente.

- struct **ManipuladorDeDesenhoDeSegmento**

  Não usar directamente.

- class **ApendiceComCor**

  Define ferramentas relacionadas com cores para os menus que as possuirem.

- class **Menu**

  Classe abstracta que representa a interface básica de todos os menus.

- class **MenuComCor**

  Classe abstracta que define a interface básica de todos os menus com cores.

- class **MenuSimples**

  Representa menus simples, que consistem numa sequência de itens.

- class **MenuDeCores**

  Representa menus de selecção das cores básicas usáveis no ecrã.

- class **MenuDeSimOuNao**

  Representa menus com apenas duas opções: sim e não.

- class **Aviso**

Representa caixas de aviso, que apenas mostram uma mensagem e das quais se sai pressionando 'enter'.

- class **CaixaDeTexto**

  Representa caixas de texto, que quando executadas permitem ao utilizador introduzir uma cadeia de caracteres.

- class **Tecla**

  Representa teclas premidas.

- class **Teclado**

  Representa o teclado.

Operadores associados ao tipo enumerado Cor

- std::ostream & **operator**<< (std::ostream &saida, **Cor** const cor)

  Insere uma cor num canal.

- std::istream & **operator**>> (std::istream &entrada, **Cor** &cor)

  Extrai uma cor de um canal.

- int const **numero_de_cores**

  Constante que guarda o número total de cores.

Manipuladores do ecrã

- **Posicao** const **cursor** (int const linha, int const coluna)

  Manipulador usado para colocar o cursor numa dada posição.

- **Posicao** const & **cursor** (**Posicao** const &posicao)

  Manipulador usado para colocar o cursor numa dada posição.

- **Caixa** const **caixa** (int const linha, int const coluna, int const numero_de_linhas, int const numero_de_colunas)

  Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição e com uma dada dimensão.

- **Caixa** const **caixa** (**Posicao** const &origem=**Posicao**(), **Dimensao** const &dimensao=**Dimensao**())

  Manipulador usado para desenhar uma caixa com caracteres especiais numa dada posição e com uma dada dimensão.

- void **fundo** (**Ecra** &**ecra**)

  Manipulador usado para indicar que as próximas escritas devem ser realizadas usando a cor do fundo do ecrã.

- void **parado** (**Ecra** &**ecra**)

  Manipulador usado para manter o cursor na posição corrente durante a próxima operação de inserção no ecrã.

- void **sobeCursor** (**Ecra** &**ecra**)

  Manipulador usado para subir o cursor no ecrã.

- void **baixaCursor** (**Ecra** &**ecra**)

  Manipulador usado para baixar o cursor no ecrã.

- void **recuaCursor** (**Ecra** &**ecra**)

  Manipulador usado para recuar o cursor no ecrã.

- void **avancaCursor** (**Ecra** &**ecra**)

  Manipulador usado para avançar o cursor no ecrã.

- **ManipuladorDaLargura largura** (int const largura)

  Manipulador usado para indicar o número de células do ecrã a ocupar na próxima operação de inserção no ecrã.

- **ManipuladorDeDesenhoDeSegmento segmento_horizontal** (int const largura)

  Manipulador usado para desenhar um segmento de recta horizontal.

- **ManipuladorDeDesenhoDeSegmento segmento_vertical** (int const altura)

  Manipulador usado para desenhar um segmento de recta vertical.

- void **refresca** (**Ecra** &**ecra**)

  Manipulador usado para refrescar o ecrã.

- void **refresca_tudo** (**Ecra** &**ecra**)

  Manipulador usado para fazer o refrescamento total do ecrã.

- void **apaga** (**Ecra** &**ecra**)

  Manipulador usado para apagar o ecrã (ou seja, para o pintar com a cor do fundo).

- void **apaga_fim_da_linha** (**Ecra** &**ecra**)

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

- void **apaga_fim_do_ecra** (**Ecra** &**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).

- void **campainha** (**Ecra** &**ecra**)

  Manipulador usado para fazer soar a campainha.

Constantes associadas à classe Caixa

- **Caixa** const **caixa_vazia**

  Constante representando uma caixa vazia, i.e., com dimensão nula.

- **Caixa** const **caixa_universal**

  Constante representando a caixa universal (enche todo o espaço).

Enumerações

- enum **Cor** { **preto**, **primeira_cor** = preto, **cinza**, **vermelho**, **vermelho_brilhante**, **verde**, **verde_brilhante**, **castanho**, **amarelo**, **azul**, **azul_brilhante**, **magenta**, **magenta_brilhante**, **ciano**, **ciano_brilhante**, **cinzento_claro**, **branco**, **ultima_cor** = branco }

  Representa as possíveis cores do texto e do fundo de cada célula do ecrã.

- enum **Simbolo** { **diamante** = '‘', **tijolo**, **ht**, **ff**, **cr**, **lf**, **grau**, **mais_menos**, **nl**, **vt**, **canto_inferior_direito**, **canto_superior_direito**, **canto_superior_esquerdo**, **canto_inferior_esquerdo**, **cruzamento**, **traco_topo**, **traco_superior**, **traco_horizontal**, **traco_inferior**, **traco_base**, **encontro_direita**, **encontro_esquerda**, **encontro_cima**, **encontro_baixo**, **traco_vertical**, **menor_ou_igual**, **maior_ou_igual**, **pi**, **diferente** }

  Representa caracteres especiais úteis para desenhar no ecrã caixas e outros símbolos pouco usuais.

- enum **Justificacao** { **ao_centro**, **a_esquerda**, **a_direita** }

  Representa os vários tipos de justificação possíveis ao escrever texto no ecrã.

Funções

- **Ecra** & **operator**<< (**Ecra** &**ecra**, **ManipuladorDaLargura** const &manipulador)

  Não documentada.

- **Ecra** & **operator**<< (**Ecra** &**ecra**, **ManipuladorDeDesenhoDeSegmento** const &manipulador)

  Não documentada.

Variáveis

- **Ecra ecra**

  Uma variável global representando o ecrã.

- std::string const **nomes_das_cores** [**numero_de_cores**]

  Constante global com os nomes das cores.

- **Teclado teclado**

  Representa o teclado.

9.3.1 Descrição detalhada

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

Em particular envolve os módulos ecras, teclados e menus. A biblioteca `Slang++` permite a utilização de algumas ferramentas 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 módulo físico possui 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.

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

```
c++ opções_de_compilação -o teste_slang teste_slang.C -lSlang++ -lUtilitarios -lslang
```

As opções `-lSlang++` e `-lslang` destinam-se a fundir o ficheiro objecto de `teste_slang` com as bibliotecas `Slang++` e S-Lang, na qual `Slang++` se baseia. A opção `-lUtilitarios` destina-se a fundir o ficheiro objecto com a biblioteca **Utilitarios,** da qual a biblioteca `Slang++` depende.

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 pretender executar 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.

Tarefa
Verificar todos os erros do S-Lang.

9.3.2 Documentação das funções

9.3.2.1 Slang::Ecra & Slang::operator<< (Ecra & *ecra*, ManipuladorDaLargura const & *manipulador*) [inline]

Não documentada.

Veja também:
Slang::largura()

Definido na linha 1089 do ficheiro ecra_impl.H.

Referências ecra, Slang::ManipuladorDaLargura::largura e Slang::Ecra::mudaLarguraDaProximaOperacaoDeEscritaPara().

9.3.2.2 Slang::Ecra & Slang::operator<< (Ecra & *ecra*, ManipuladorDeDesenhoDeSegmento const & *manipulador*) [inline]

Não documentada.

Veja também:
Slang::segmento_horizontal()
Slang::segmento_vertical()

Definido na linha 1108 do ficheiro ecra_impl.H.

Referências Slang::Ecra::desenhaSegmentoHorizontalCom(), Slang::Ecra::desenhaSegmentoVerticalCom(), Slang::ManipuladorDeDesenhoDeSegmento::dimensao, Slang::ManipuladorDeDesenhoDeSegmento::e_horizontal e ecra.

9.4 Referência ao namespace std

9.5 Referência ao namespace Utilitarios

Este espaço nominativo envolve todas as ferramentas da biblioteca `Utilitarios`.

Componentes

- class **Data**

  Representa datas posteriores a 1582, i.e., datas no calendário Gregoriano, adoptado por Portugal e outros países católicos em 1582.

- class **Ignorador**

  Representa manipuladores de canais de entrada (istream) que descartam caracteres.

Tipos básicos para datas e tempos

- typedef long **Duracao**

  Representa durações em dias.

- typedef int **Dia**

  Representa o dia do mês de uma data.

- typedef int **Ano**

  Representa o ano de uma data.

- enum **Mes** { **janeiro** = 1, **fevereiro**, **marco**, **abril**, **maio**, **junho**, **julho**, **agosto**, **setembro**, **outubro**, **novembro**, **dezembro** }

  Representa os meses do ano.

- enum **DiaDaSemana** { **domingo** = 0, **segunda_feira**, **terca_feira**, **quarta_feira**, **quinta_feira**, **sexta_feira**, **sabado** }

  Representa os dias da semana.

- int const **numero_total_de_meses** = 12

  A número de meses no ano.

- std::string **nomes_dos_meses** [**numero_total_de_meses**+1]

  Os nomes dos meses do ano.

- int const **numero_total_de_dias_da_semana** = 7

  O número de dias na semana.

- std::string **nomes_dos_dias_da_semana** [**numero_total_de_dias_da_semana**]

  Os nomes dos dias da semana.

Operações de concatenação de cadeias de caracteres com

inteiros.

- template<typename Char> std::basic_string< Char > & **operator+=** (std::basic_string< Char > &cadeia, int const valor)

  Acrescenta a representação em base decimal posicional de um inteiro a uma cadeia.

- template<typename Char> std::basic_string< Char > **operator+** (std::basic_string< Char > cadeia, int const valor)

  Devolve a concatenação de uma cadeia com a representação na base decimal posicional de um inteiro.

- template<typename Char> std::basic_string< Char > **operator+** (int const valor, std::basic_string< Char > cadeia)

  Devolve a concatenação de uma cadeia com a representação na base decimal posicional de um inteiro.

Funções utilitários para cálculo de datas.

- bool **eBissexto** (**Ano** const ano)

  Indica se o ano dado é bissexto.

- int **numeroDeDiasEm** (**Mes** const mes, **Ano** const ano)

  Devolve o número de dias numa dado mes de um dado ano.

Operações associadas ao tipo Mes

- **Mes** & **operator++** (**Mes** &mes)

  Incrementação prefixa de meses.

- **Mes** & **operator–** (**Mes** &mes)

  Decrementação prefixa de meses.

- **Mes operator++** (**Mes** &mes, int)

  Incrementação sufixa de meses.

- **Mes operator–** (**Mes** &mes, int)

  Decrementação sufixa de meses.

- **Mes** & **operator+=** (**Mes** &mes, int const numero_de_meses)

  Avança mês de um número de meses.

- **Mes** & **operator-=** (**Mes** &mes, int const numero_de_meses)

  Recua mês de um número de meses.

- **Mes operator+** (**Mes** const mes, int const numero_de_meses)

  Devolve a adição de um mês com um dado número de meses.

- **Mes operator+** (int const numero_de_meses, **Mes** const mes)

  Devolve a adição de um mês com um dado número de meses.

- int **operator-** (**Mes** const um_mes, **Mes** const outro_mes)

  Devolve a distância em número de meses entre dois meses.

- **Mes operator-** (**Mes** const mes, int const numero_de_meses)

  Devolve a subtracção de um mês de um dado número de meses.

- std::ostream & **operator<<** (std::ostream &saida, **Mes** const mes)

  Insere um mês num canal.

- std::istream & **operator>>** (std::istream &entrada, **Mes** &mes)

  Extrai um mês de um canal.

Operações associadas ao tipo DiaDaSemana

- **DiaDaSemana** & **operator++** (**DiaDaSemana** &dia_da_semana)

  Incrementação prefixa de dias da semana.

- **DiaDaSemana** & **operator--** (**DiaDaSemana** &dia_da_semana)

  Decrementação prefixa de dias da semana.

- **DiaDaSemana operator++** (**DiaDaSemana** &dia_da_semana, int)

  Incrementação sufixa de dias da semana.

- **DiaDaSemana operator--** (**DiaDaSemana** &dia_da_semana, int)

  Incrementação sufixa de dias da semana.

- **DiaDaSemana** & **operator+=** (**DiaDaSemana** &dia_da_semana, int const numero_de_dias)

  Avança dia da semana de um número de dias da semana.

- **DiaDaSemana** & **operator-=** (**DiaDaSemana** &dia_da_semana, int const numero_de_dias)

  Recua dia da semana de um número de dias da semana.

- **DiaDaSemana operator+** (**DiaDaSemana** const dia_da_semana, int const numero_de_dias)

  Devolve a adição de um dia da semana com um dado número de dias da semana.

- **DiaDaSemana operator+** (int const numero_de_dias, **DiaDaSemana** const dia_da_semana)

  Devolve a adição de um dia da semana com um dado número de dias da semana.

- int **operator-** (**DiaDaSemana** const um_dia_da_semana, **DiaDaSemana** const outro_dia_da_semana)

  Devolve a distância em número de dias da semana entre dois dias da semana.

- **DiaDaSemana operator-** (**DiaDaSemana** const dia_da_semana, int const numero_de_dias)

  Devolve a subtracção de um dia da semana de um dado número de dias da semana.

- std::ostream & **operator<<** (std::ostream &saida, **DiaDaSemana** const dia_da_semana)

  Insere um dia da semana num canal.

- std::istream & **operator>>** (std::istream &entrada, **DiaDaSemana** &dia_da_semana)

  Extrai um dia da semana de um canal.

Operações associadas à classe Data

- bool **operator==** (**Data** const &uma_data, **Data** const &outra_data)

  Indica se duas datas são iguais.

- bool **operator!=** (**Data** const &uma_data, **Data** const &outra_data)

  Indica se duas datas são diferentes.

- bool **operator<** (**Data** const &uma_data, **Data** const &outra_data)

  Indica se uma data é menor que outra.

- bool **operator>** (**Data** const &uma_data, **Data** const &outra_data)

  Indica se uma data é maior que outra.

- bool **operator<=** (**Data** const &uma_data, **Data** const &outra_data)

  Indica se uma data é menor ou igual a outra.

- bool **operator>=** (**Data** const &uma_data, **Data** const &outra_data)

  Indica se uma data é maior ou igual a outra..

- **Duracao operator-** (**Data** const &uma_data, **Data** const &outra_data)

  Devolve a distância entre duas datas.

- **Data operator-** (**Data** const &data, **Duracao** const &duracao)

  Devolve a subtracção de uma duração de uma data.

- **Data operator+** (**Data** const &data, **Duracao** const &duracao)

  Devolve a adição de uma duração a uma data.

- **Data operator+** (**Duracao** const &duracao, **Data** const &data)

  Devolve a adição de uma duração a uma data.

- std::ostream & **operator<<** (std::ostream &saida, **Data** const &data)

  Insere uma data num canal.

- std::istream & **operator>>** (std::istream &entrada, **Data** &data)

  Extrai uma data de um canal.

Manipuladores extra para std::istream

- **Ignorador** const **il** ('\n')

  Manipulador que descarta todos os caracteres até ao próximo fim-de-linha (`'\n'`*).*

- **Ignorador** const **ill** ('\n', true)

  Manipulador que descarta todos os caracteres até ao próximo fim-de-linha (`'\n'`*), mas primeiro limpa possíveis condições de erro do canal.*

Ferramentas de localização para português.

- template<typename C> bool **eImprimivel** (C const caractere)

  Indica se o caractere recebido como argumento é imprimível de acordo com as convenções portuguesas.

- template<typename C> bool **eImprimivel** (std::basic_string< C > const &cadeia)

  Indica se a cadeia de caracteres recebida como argumento é imprimível de acordo com as convenções portuguesas.

Funções

- std::istream & **operator**>> (std::istream &entrada, **Ignorador** const &ignorador)

  Operador de extracção para os manipuladores de ignorar/descartar caracteres.

9.5.1 Descrição detalhada

Este espaço nominativo envolve todas as ferramentas da biblioteca `Utilitarios`.

Em particular envolve os módulos ignoradores, datas e cadeias. A biblioteca `Utilitarios` contém utilitários variados, desde manipuladores extra para os canais de entrada e saída até classes para representar datas.

Para construir um programa (neste caso teste_data.C) que utilize esta biblioteca deve dar o seguinte comando:

```
c++ opções_de_compilação -o teste_data teste_data.C -lUtilitarios
```

9.5.2 Documentação das funções

9.5.2.1 template<typename Char> std::basic_string< Char > & Utilitarios::operator+= (std::basic_string< Char > & *cadeia*, int const *valor*) [inline]

Acrescenta a representação em base decimal posicional de um inteiro a uma cadeia.

Precondição:
cadeia = c.

Poscondição:
operator+= := cadeia = c + representação de valor.

Definido na linha 11 do ficheiro cadeia_impl.H.

9.5.2.2 template<typename Char> std::basic_string< Char > Utilitarios::operator+ (std::basic_string< Char > *cadeia*, int const *valor*)

Devolve a concatenação de uma cadeia com a representação na base decimal posicional de um inteiro.

Precondição:
V.

Poscondição:
operator+ = cadeia + representação de valor.

Definido na linha 21 do ficheiro cadeia_impl.H.

9.5.2.3 template<typename Char> std::basic_string< Char > Utilitarios::operator+ (int const *valor*, std::basic_string< Char > *cadeia*)

Devolve a concatenação de uma cadeia com a representação na base decimal posicional de um inteiro.

Precondição:
V.

Poscondição:
operator+ = cadeia + representação de valor.

Definido na linha 28 do ficheiro cadeia_impl.H.

9.5.2.4 bool Utilitarios::eBissexto (Ano const *ano*) [inline]

Indica se o ano dado é bissexto.

Precondição:
Data::ano_minimo <= ano.

Poscondição:
eBissexto = ano é bissexto segundo o calendário gregoriano.

Definido na linha 15 do ficheiro data_impl.H.

Referências Ano.

Referenciado por Utilitarios::Data::anoEBissexto() e numeroDeDiasEm().

9.5.2.5 int Utilitarios::numeroDeDiasEm (Mes const *mes*, Ano const *ano*) [inline]

Devolve o número de dias numa dado mes de um dado ano.

Precondição:
Data::ano_minimo < ano.

Poscondição:
numeroDeDiasEm = número de dias do mês *mes* no ano *ano*.

Definido na linha 20 do ficheiro data_impl.H.

Referências Ano, dezembro, eBissexto(), janeiro, Mes e numero_total_de_meses.

Referenciado por Utilitarios::Data::Data(), Utilitarios::Data::numeroDeDiasNoMes() e operator>>().

Capítulo 10

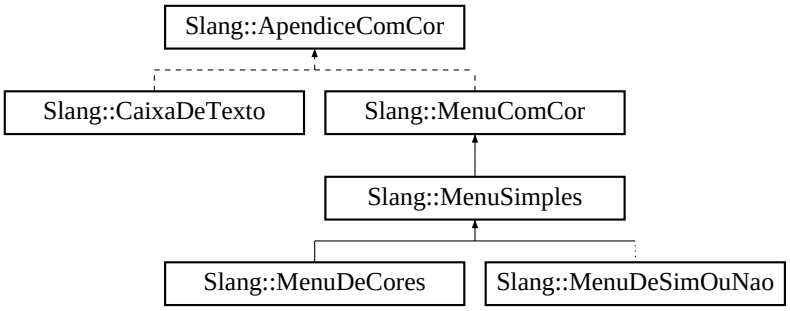
Pacotes Documentação da classe

10.1 Referência à classe Slang::ApendiceComCor

Define ferramentas relacionadas com cores para os menus que as possuirem.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::ApendiceComCor:



Membros protegidos

Construtores

- **ApendiceComCor** ()
  Constrói um novo apêndice com cor, atribuindo cores por omissão.
- **~ApendiceComCor** ()
  Destrói um apêndice com cor.

Inspectores

- **Ecra::ObjectoCor** const & **objectoCorDaBorda** () const
  Devolve o objecto cor da borda.
- **Ecra::ObjectoCor** const & **objectoCorDoTitulo** () const
  Devolve o objecto cor do título.

- **Ecra::ObjectoCor** const & **objectoCorDosItens** () const
  Devolve o objecto cor dos itens.

- **Ecra::ObjectoCor** const & **objectoCorDoItemCorrente** () const
  Devolve o objecto cor do item corrente.

Modificadores

- **Ecra::ObjectoCor** & **objectoCorDaBorda** ()
  Devolve o objecto cor da borda.

- **Ecra::ObjectoCor** & **objectoCorDoTitulo** ()
  Devolve o objecto cor do título.

- **Ecra::ObjectoCor** & **objectoCorDosItens** ()
  Devolve o objecto cor dos itens.

- **Ecra::ObjectoCor** & **objectoCorDoItemCorrente** ()
  Devolve o objecto cor do item corrente.

10.1.1 Descrição detalhada

Define ferramentas relacionadas com cores para os menus que as possuirem.

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

1. borda: preto sobre branco (ou cinzento).
2. título: vermelho sobre branco (ou cinzento).
3. itens: preto sobre branco (ou cinzento).
4. item corrente: amarelo sobre azul.

Um apêndice serve para equipar outras classes de ferramentas mais ou menos avulsas. Estas classes são conhecidas também por *mixin* .

Invariante:
V.

Definido na linha 46 do ficheiro menu.H.

10.1.2 Documentação dos Construtores & Destrutor

10.1.2.1 Slang::ApendiceComCor::ApendiceComCor () `[inline, protected]`

Constrói um novo apêndice com cor, atribuindo cores por omissão.

Precondição:
V.

Poscondição:
objectoCorDaBorda().corDoTexto() = preto e **objectoCorDaBorda()**.corDoFundo() = branco e **objectoCorDoTitulo()**.corDoTexto() = vermelho e **objectoCorDoTitulo()**.corDoFundo() = branco e **objectoCorDosItens()**.corDoTexto() = preto e **objectoCorDosItens()**.corDoFundo() = branco e **objectoCorDoItemCorrente()**.corDoTexto() = amarelo e **objectoCorDoItemCorrente()**.corDoFundo() = azul.

Definido na linha 11 do ficheiro menu_impl.H.

Referências Slang::amarelo, Slang::azul, Slang::branco, Slang::preto e Slang::vermelho.

10.1.2.2 Slang::ApendiceComCor::~ApendiceComCor () [inline, protected]

Destrói um apêndice com cor.

Precondição:
V.

Definido na linha 21 do ficheiro menu_impl.H.

10.1.3 Documentação dos métodos

10.1.3.1 Slang::Ecra::ObjectoCor const & Slang::ApendiceComCor::objectoCorDaBorda () const [inline, protected]

Devolve o objecto cor da borda.

Precondição:
V.

Poscondição:
objectoCorDaBorda é o objecto cor usado para representar a borda do menu.

Definido na linha 27 do ficheiro menu_impl.H.

10.1.3.2 Slang::Ecra::ObjectoCor const & Slang::ApendiceComCor::objectoCorDoTitulo () const [inline, protected]

Devolve o objecto cor do título.

Precondição:
V.

Poscondição:
objectoCorDoTitulo é o objecto cor usado para representar o título do menu.

Definido na linha 35 do ficheiro menu_impl.H.

10.1.3.3 Slang::Ecra::ObjectoCor const & Slang::ApendiceComCor::objectoCorDosItens () const [inline, protected]

Devolve o objecto cor dos itens.

Precondição:
V.

Poscondição:
objectoCorDosItens é o objecto cor usado para representar os itens do menu.

Definido na linha 43 do ficheiro menu_impl.H.

10.1.3.4 Slang::Ecra::ObjectoCor const & Slang::ApendiceComCor::objectoCorDoItemCorrente () const [inline, protected]

Devolve o objecto cor do item corrente.

Precondição:
V.

Poscondição:
objectoCorDoItemCorrente é o objecto cor usado para representar o item corrente do menu.

Definido na linha 51 do ficheiro menu_impl.H.

10.1.3.5 Slang::Ecra::ObjectoCor & Slang::ApendiceComCor::objectoCorDaBorda () [inline, protected]

Devolve o objecto cor da borda.

Precondição:
V.

Poscondição:
objectoCorDaBorda é o objecto cor usado para representar a borda do menu.

Definido na linha 58 do ficheiro menu_impl.H.

10.1.3.6 Slang::Ecra::ObjectoCor & Slang::ApendiceComCor::objectoCorDoTitulo () [inline, protected]

Devolve o objecto cor do título.

Precondição:
V.

Poscondição:
objectoCorDoTitulo é o objecto cor usado para representar o título do menu.

Definido na linha 65 do ficheiro menu_impl.H.

10.1.3.7 Slang::Ecra::ObjectoCor & Slang::ApendiceComCor::objectoCorDosItens () [inline, protected]

Devolve o objecto cor dos itens.

Precondição:
V.

Poscondição:
objectoCorDosItens é o objecto cor usado para representar os itens do menu.

Definido na linha 72 do ficheiro menu_impl.H.

10.1.3.8 Slang::Ecra::ObjectoCor & Slang::ApendiceComCor::objectoCorDoItemCorrente () [inline, protected]

Devolve o objecto cor do item corrente.

Precondição:
V.

Poscondição:
objectoCorDoItemCorrente é o objecto cor usado para representar o item corrente do menu.

Definido na linha 80 do ficheiro menu_impl.H.

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

- **menu.H**
- **menu_impl.H**

10.2 Referência à classe Slang::Aviso

Representa caixas de aviso, que apenas mostram uma mensagem e das quais se sai pressionando 'enter'.

`#include <Slang++/menu.H>`

Membros públicos

Construtores

- **Aviso** (std::string const &mensagem)
  Constrói uma nova caixa de aviso com a mensagem dada.
- virtual ~**Aviso** ()
  Destrutor polimórfico marca uma classe polimórfica.

Interface com o utilizador

- virtual void **interage** ()
  Executa a caixa de aviso, i.e., interage com o utilizador do programa.

10.2.1 Descrição detalhada

Representa caixas de aviso, que apenas mostram uma mensagem e das quais se sai pressionando 'enter'.

Invariante:
V.

Definido na linha 428 do ficheiro menu.H.

10.2.2 Documentação dos Construtores & Destrutor

10.2.2.1 Slang::Aviso::Aviso (std::string const & *mensagem*) `[inline, explicit]`

Constrói uma nova caixa de aviso com a mensagem dada.

Precondição:
mensagem <> "" e eImprimivel(mensagem).

Definido na linha 212 do ficheiro menu_impl.H.

10.2.2.2 Slang::Aviso::~Aviso () `[inline, virtual]`

Destrutor polimórfico marca uma classe polimórfica.

Precondição:
V.

Definido na linha 220 do ficheiro menu_impl.H.

10.2.3 Documentação dos métodos

10.2.3.1 void Slang::Aviso::interage () [inline, virtual]

Executa a caixa de aviso, i.e., interage com o utilizador do programa.

Precondição:
V.

Definido na linha 225 do ficheiro menu_impl.H.

Referências Slang::MenuSimples::interage().

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

- **menu.H**
- **menu_impl.H**

10.3 Referência à classe Slang::Caixa

Representa caixas no ecrã.

`#include <Slang++/ecra.H>`

Membros públicos

Construtores

- **Caixa** (**Posicao** const &origem=**Posicao**(), **Dimensao** const &dimensao=**Dimensao**())
  Constrói uma nova caixa.
- **Caixa** (**Posicao** const &origem, **Posicao** const &extremo)
  Constrói uma nova caixa.

Inspectores

- **Posicao** const & **origem** () const
  Devolve a origem da caixa, i.e., a posição do seu vértice superior esquerdo.
- **Posicao extremo** () const
  Devolve o extremo da caixa, i.e., a posição do seu vértice inferior direito.
- **Dimensao** const & **dimensao** () const
  Devolve a dimensao da caixa.

Predicados

- bool **eVazia** () const
  Indica se a caixa está vazia, i.e., se a sua dimensão é nula.
- bool **eCanonica** () const
  Indica se a caixa é canónica, i.e., se a sua dimensão é canónica.
- bool **contem** (**Posicao** const &posicao) const
  Indica se uma posição se encontra sobre a caixa, quer sobre a sua borda quer sobre o seu interior.
- bool **bordaContem** (**Posicao** const &posicao) const
  Indica se uma posição se encontra sobre a borda da caixa, quee existe mesmo para uma caixa vazia ou com dimensão negativa em alguma das direcções.

Modificadores

- void **mudaOrigemPara** (**Posicao** const &nova_origem)
  Muda a origem da caixa para nova_origem.
- void **mudaExtremoPara** (**Posicao** const &novo_extremo)
  Muda o extremo da caixa para novo_extremo.

- void **mudaDimensaoPara** (**Dimensao** const &nova_dimensao)
  Muda a dimensão da caixa para nova_dimensão.

Serializadores

- **Caixa** (std::istream &entrada)
  Constrói uma caixa por carregamento a partir de um canal de entrada.
- void **carregaDe** (std::istream &entrada)
  Carrega a caixa a partir de um canal.
- void **guardaEm** (std::ostream &saida) const
  Guarda a caixa num canal.

Operadores especiais de atribuição

- Caixa & **operator+=** (**Dimensao** const &deslocamento)
  Desloca a caixa de um dado deslocamento.
- Caixa & **operator-=** (**Dimensao** const &deslocamento)
  Desloca a caixa de um dado deslocamento.
- Caixa & **operator+=** (Caixa const &outra_caixa)
  Une a caixa com a caixa outra_caixa.
- Caixa & **operator** *= (Caixa const &outra_caixa)
  Intersecta a caixa com a caixa outra_caixa.

Funções associadas

(Note que não são funções membro)

- Caixa const **operator+** (Caixa caixa, **Dimensao** const &deslocamento)
  Devolve a adição de uma caixa e de uma dimensão.
- Caixa const **operator-** (Caixa caixa, **Dimensao** const &deslocamento)
  Devolve a subtracção de uma caixa e de uma dimensão.
- Caixa const **operator+** (**Dimensao** const &deslocamento, Caixa caixa)
  Devolve a adição de uma caixa e de uma dimensão.
- Caixa const **operator+** (Caixa uma_caixa, Caixa const &outra_caixa)
  Devolve a união de duas caixas.
- Caixa const **operator** * (Caixa uma_caixa, Caixa const &outra_caixa)
  Devolve a intersecção de duas caixas.
- bool **operator==** (Caixa const &uma_caixa, Caixa const &outra_caixa)

Indica se duas caixas são iguais.

- bool **operator!=** (Caixa const &uma_caixa, Caixa const &outra_caixa)

  Indica se duas caixas são diferentes.

10.3.1 Descrição detalhada

Representa caixas no ecrã.

Uma caixa é um conjunto de células formando um rectângulo de lados não oblíquos, i.e., verticais ou horizontais. Uma caixa tem uma origem e um extremo. A dimensão da caixa é a diferença entre o extremo e a origem, mas somada de Dimensao(1, 1), visto que quer a origem quer o extremo se encontram sobre a caixa. Uma caixa diz-se canónica se a sua dimensão for canónica. Uma caixa diz-se vazia se a sua dimensão for Dimensao(0, 0). As caixas podem ser inseridas no ecrã, sendo desenhado um rectângulo oco com as suas dimensões. Por exemplo, o código

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

using namespace Slang;

// Mostra uma caixa com 1/4 da área do ecrã (metade das suas dimensões
// horizontal e vertical) centrada no ecrã.
int main()
{
    Caixa uma_caixa(Posicao(ecra.dimensao().numeroDeLinhas() / 4,
                            ecra.dimensao().numeroDeColunas() / 4),
                    Dimensao(ecra.dimensao().numeroDeLinhas() / 2,
                             ecra.dimensao().numeroDeColunas() / 2));

    ecra << uma_caixa << refresca;

    teclado.leProximaTeclaDisponivel();
}
```

escreve uma caixa centrada no ecrã e com 1/4 da sua área.

Invariante:
V.

Tarefa
Falta implementar as operações de serialização.

Definido na linha 556 do ficheiro ecra.H.

10.3.2 Documentação dos Construtores & Destrutor

10.3.2.1 Slang::Caixa::Caixa (Posicao const & *origem* = Posicao(), Dimensao const & *dimensao* = Dimensao()) [inline, explicit]

Constrói uma nova caixa.

Por omissão a caixa localiza-se no canto superior esquerdo do ecrã e a sua dimensão é nula. Ao canto superior esquerdo de uma caixa chama-se a sua "origem". Ao canto oposto chama-se "extremo". Note-se que uma caixa pode ter dimensão nula ou mesmo negativa em alguma direcção.

Precondição:
V.

Poscondição:
origem() = *origem* e **dimensao**() = *dimensao*.

Definido na linha 293 do ficheiro ecra_impl.H.

Referenciado por carregaDe().

10.3.2.2 Slang::Caixa::Caixa (Posicao const & *origem*, Posicao const & *extremo*) [inline, explicit]

Constrói uma nova caixa.

Neste caso a construção é feita a partir das posições de dois vértices da caixa. Esses vértices são o superior esquerdo (origem) e o inferior direito (extremo). Se a origem estiver à direita ou abaixo do extremo, a caixa diz-se não-canónica.

Precondição:
V.

Poscondição:
origem() = origem e **extremo**() = *extremo*.

Definido na linha 300 do ficheiro ecra_impl.H.

10.3.2.3 Slang::Caixa::Caixa (std::istream & *entrada*) [inline, explicit]

Constrói uma caixa por carregamento a partir de um canal de entrada.

Assume-se que os dados no canal têm um formato equivalente ao produzido pela operação **guardaEm**().

Precondição:
entrada.good().

Poscondição:
Posicao é a caixa que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se a construção por carregamento falhar.

Definido na linha 394 do ficheiro ecra_impl.H.

10.3.3 Documentação dos métodos

10.3.3.1 Slang::Posicao const & Slang::Caixa::origem () const [inline]

Devolve a origem da caixa, i.e., a posição do seu vértice superior esquerdo.

Precondição:
V.

Poscondição:
origem() = origem da caixa.

Definido na linha 308 do ficheiro ecra_impl.H.

Referenciado por bordaContem(), contem(), Slang::Ecra::desenha(), mudaExtremoPara(), operator *=(), operator+=() e Slang::Ecra::trocoDe().

10.3.3.2 Slang::Posicao Slang::Caixa::extremo () const [inline]

Devolve o extremo da caixa, i.e., a posição do seu vértice inferior direito.

Precondição:
V.

Poscondição:
extremo() = extremo da caixa.

Definido na linha 315 do ficheiro ecra_impl.H.

Referenciado por bordaContem(), contem(), operator *=() e operator+=().

10.3.3.3 Slang::Dimensao const & Slang::Caixa::dimensao () const [inline]

Devolve a dimensao da caixa.

Precondição:
V.

Poscondição:
dimensao() = dimensão da caixa.

Definido na linha 322 do ficheiro ecra_impl.H.

Referenciado por Slang::Ecra::desenha(), eCanonica(), eVazia() e Slang::Ecra::trocoDe().

10.3.3.4 bool Slang::Caixa::eVazia () const [inline]

Indica se a caixa está vazia, i.e., se a sua dimensão é nula.

Precondição:
V.

Poscondição:
eVazia = (dimensao() = Dimensao(0, 0)).

Definido na linha 329 do ficheiro ecra_impl.H.

Referências dimensao().

10.3.3.5 bool Slang::Caixa::eCanonica () const [inline]

Indica se a caixa é canónica, i.e., se a sua dimensão é canónica.

Precondição:
V.

Poscondição:
eCanonica = **dimensao().eCanonica()**.

Definido na linha 336 do ficheiro ecra_impl.H.

Referências dimensao() e Slang::Dimensao::eCanonica().

Referenciado por Slang::Ecra::operator<<() e Slang::Ecra::trocoDe().

10.3.3.6 bool Slang::Caixa::contem (Posicao const & *posicao*) const [inline]

Indica se uma posição se encontra sobre a caixa, quer sobre a sua borda quer sobre o seu interior.

Note-se que uma caixa com dimensão negativa em alguma das direcções não tem interior.

Precondição:
V.

Poscondição:
contem = *posicao* dentro da caixa (interior ou borda).

Definido na linha 343 do ficheiro ecra_impl.H.

Referências Slang::Posicao::coluna(), extremo(), Slang::Posicao::linha() e origem().

10.3.3.7 bool Slang::Caixa::bordaContem (Posicao const & *posicao*) const [inline]

Indica se uma posição se encontra sobre a borda da caixa, quee existe mesmo para uma caixa vazia ou com dimensão negativa em alguma das direcções.

Precondição:
V.

Poscondição:
bordaContem = *posicao* na borda da caixa.

Definido na linha 353 do ficheiro ecra_impl.H.

Referências Slang::Posicao::coluna(), extremo(), Slang::Posicao::linha() e origem().

10.3.3.8 void Slang::Caixa::mudaOrigemPara (Posicao const & *nova_origem*) [inline]

Muda a origem da caixa para *nova_origem.*

Atenção: a dimensão mantém-se, pelo que o extremo da caixa é alterado em paralelo com a sua origem!

Precondição:
dimensao() = d.

Poscondição:
origem() = *nova_origem* e **dimensao()** = d.

Definido na linha 367 do ficheiro ecra_impl.H.

Referenciado por operator *=() e operator+=().

10.3.3.9 void Slang::Caixa::mudaExtremoPara (Posicao const & *novo_extremo*) [inline]

Muda o extremo da caixa para *novo_extremo*.

Atenção: a origem mantém-se, pelo que a dimensão da caixa é alterada, ao contrário do que acontece quando a origem é alterada directamente!

Precondição:
origem() = o.

Poscondição:
extremo() = *novo_extremo* e **origem()** = o.

Definido na linha 376 do ficheiro ecra_impl.H.

Referências origem().

Referenciado por operator *=() e operator+=().

10.3.3.10 void Slang::Caixa::mudaDimensaoPara (Dimensao const & *nova_dimensao*) [inline]

Muda a dimensão da caixa para *nova_dimensão*.

Atenção: a origem mantém-se, pelo que o extremo da caixa é alterado em paralelo com a sua dimensão!

Precondição:
origem() = o.

Poscondição:
dimensao() = *nova_dimensao* e **origem()** = o.

Definido na linha 385 do ficheiro ecra_impl.H.

10.3.3.11 void Slang::Caixa::carregaDe (std::istream & *entrada*) [inline]

Carrega a caixa a partir de um canal.

Assume-se que os dados no canal têm um formato equivalente ao produzido pela operação **guardaEm()**.

Precondição:
entrada.good().

Poscondição:
*this é a caixa que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se o carregamento falhar. Nesse caso o canal pode ficar parcialmente lido.

Definido na linha 401 do ficheiro ecra_impl.H.

Referências Caixa().

10.3.3.12 void Slang::Caixa::guardaEm (std::ostream & *saida*) const

Guarda a caixa num canal.

O formato produzido é compatível com o que o método **carregaDe()** espera.

Precondição:
saida.good().

Excepções:
ErroAoGuardar é lançada se a operação falhar. Nesse caso o canal pode ficar parcialmente escrito.

Definido na linha 63 do ficheiro ecra.C.

Referências Slang::Dimensao::guardaEm() e Slang::Posicao::guardaEm().

10.3.3.13 Slang::Caixa & Slang::Caixa::operator+= (Dimensao const & *deslocamento*) [inline]

Desloca a caixa de um dado deslocamento.

Precondição:
origem() = o e **dimensao**() = d.

Poscondição:
origem() = o + *deslocamento* e **dimensao**() = d.

Definido na linha 411 do ficheiro ecra_impl.H.

10.3.3.14 Slang::Caixa & Slang::Caixa::operator-= (Dimensao const & *deslocamento*) [inline]

Desloca a caixa de um dado deslocamento.

Precondição:
origem() = o e **dimensao**() = d.

Poscondição:
origem() = o - *deslocamento* e **dimensao**() = d.

Definido na linha 423 do ficheiro ecra_impl.H.

10.3.3.15 Slang::Caixa & Slang::Caixa::operator+= (Caixa const & *outra_caixa*) [inline]

Une a caixa com a caixa *outra_caixa*.

A união de duas caixas é a menor caixa que as envolve.

Precondição:
origem() = o e **extremo()** = e.

Poscondição:
origem().linha() = min(o.linha(), outra_caixa.origem().linha()) e **origem()**.coluna() = min(o.coluna(), outra_caixa.origem().coluna()) e **extremo()**.linha() = max(e.linha(), outra_caixa.extremo().linha()) e **extremo()**.coluna() = max(e.coluna(), outra_caixa.extremo().coluna()).

Definido na linha 434 do ficheiro ecra_impl.H.

Referências Slang::Posicao::coluna(), extremo(), Slang::Posicao::linha(), mudaExtremoPara(), mudaOrigemPara() e origem().

10.3.3.16 Slang::Caixa & Slang::Caixa::operator *= (Caixa const & *outra_caixa*) [inline]

Intersecta a caixa com a caixa *outra_caixa*.

A intersecção de duas caixas é a maior caixa que se inclui em ambas. Note-se, no entanto, que se as duas caixas forem regulares (i.e., com dimensão positiva) e se não intersectarem geometricamente, esta operação tem como resultado uma caixa com pelo menos uma dimensão negativa ou nula.

Precondição:
origem() = o e **extremo()** = e.

Poscondição:
origem().linha() = max(o.linha(), outra_caixa.origem().linha()) e **origem()**.coluna() = max(o.coluna(), outra_caixa.origem().coluna()) e **extremo()**.linha() = min(e.linha(), outra_caixa.extremo().linha()) e **extremo()**.coluna() = min(e.coluna(), outra_caixa.extremo().coluna()).

Definido na linha 453 do ficheiro ecra_impl.H.

Referências Slang::Posicao::coluna(), extremo(), Slang::Posicao::linha(), mudaExtremoPara(), mudaOrigemPara() e origem().

10.3.4 Documentação das classes amigas e funções relacionadas

10.3.4.1 Caixa const operator+ (Caixa *caixa*, Dimensao const & *deslocamento*) [related]

Devolve a adição de uma caixa e de uma dimensão.

A dimensão é interpretada como um vector de deslocamento. A caixa devolvida é igual à recebida como argumento, embora deslocada de acordo com o deslocamento dado.

Precondição:
caixa = c.

Poscondição:
operator+.**dimensao()** = c.dimensao() e operator+.**origem()** = c.origem() + deslocamento.

10.3.4.2 Caixa const operator- (Caixa *caixa*, Dimensao const & *deslocamento*) [related]

Devolve a subtracção de uma caixa e de uma dimensão.

A dimensão é interpretada como um vector de deslocamento. A caixa devolvida é igual à recebida como argumento, embora deslocada de acordo com o deslocamento dado.

Precondição:
caixa = c.

Poscondição:
operator-.**dimensao()** = c.dimensao() e operator-.**origem()** = c.origem() - deslocamento.

10.3.4.3 Caixa const operator+ (Dimensao const & *deslocamento*, Caixa *caixa*) [related]

Devolve a adição de uma caixa e de uma dimensão.

A dimensão é interpretada como um vector de deslocamento. A caixa devolvida é igual à recebida como argumento, embora deslocada de acordo com o deslocamento dado.

Precondição:
caixa = c.

Poscondição:
operator+.**dimensao()** = c.dimensao() e operator+.**origem()** = c.origem() + deslocamento.

10.3.4.4 Caixa const operator+ (Caixa *uma_caixa*, Caixa const & *outra_caixa*) [related]

Devolve a união de duas caixas.

A união de duas caixas é a menor caixa que as envolve.

Precondição:
uma_caixa = c.

Poscondição:
operator+.**origem()**.linha() = min(c.**origem()**.linha(), outra_caixa.origem().linha()) e operator+.**origem()**.coluna() = min(c.**origem()**.coluna(), outra_caixa.origem().coluna()) e operator+.**extremo()**.linha() = max(c.**extremo()**.linha(), outra_caixa.extremo().linha()) e operator+.**extremo()**.coluna() = max(c.**extremo()**.coluna(), outra_caixa.extremo().coluna()).

10.3.4.5 Caixa const operator * (Caixa *uma_caixa*, Caixa const & *outra_caixa*) [related]

Devolve a intersecção de duas caixas.

A intersecção de duas caixas é a maior caixa que se inclui em ambas. Note-se, no entanto, que se as duas caixas forem regulares (i.e., com dimensão positiva) e se não intersectarem geometricamente, esta operação tem como resultado uma caixa com pelo menos uma dimensão negativa ou nula.

Precondição:
uma_caixa = c.

Poscondição:
origem().linha() = max(c.**origem()**.linha(), outra_caixa.origem().linha()) e **origem()**.coluna() = max(c.**origem()**.coluna(), outra_caixa.origem().coluna()) e **extremo()**.linha() = min(c.**extremo()**.linha(), outra_caixa.extremo().linha()) e **extremo()**.coluna() = min(c.**extremo()**.coluna(), outra_caixa.extremo().coluna()).

10.3.4.6 bool operator== (Caixa const & *uma_caixa*, Caixa const & *outra_caixa*) [related]

Indica se duas caixas são iguais.

Precondição:
V.

Poscondição:
operator== = (uma_caixa.**origem()** = outra_caixa.origem() e uma_caixa.dimensao() = outra_caixa.dimensao()).

10.3.4.7 bool operator!= (Caixa const & *uma_caixa*, Caixa const & *outra_caixa*) [related]

Indica se duas caixas são diferentes.

Precondição:
V.

Poscondição:
operator!= = (uma_caixa.**origem()** <> outra_caixa.origem() ou uma_caixa.dimensao() <> outra_caixa.dimensao()).

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

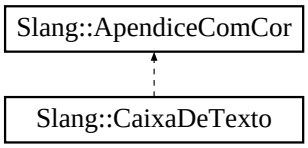
- **ecra.H**
- **ecra.C**
- **ecra_impl.H**

10.4 Referência à classe Slang::CaixaDeTexto

Representa caixas de texto, que quando executadas permitem ao utilizador introduzir uma cadeia de caracteres.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::CaixaDeTexto:



Membros públicos

Construtores

- **CaixaDeTexto** (std::string const &titulo, std::string const &texto_inicial="", std::string const &caracteres_admissiveis="", bool deve_impedir_caixa_vazia=false, std::string::size_type espacos=0)

  Constrói uma nova caixa de texto, com um dado título, um dado texto inicial (por omissão ""), um dado conjunto de caracteres admissíveis (por omissão "", que significa que qualquer caracter imprimível é admissível), com a opção de impedir a terminação da interacção com a caixa vazia (por omissão não há qualquer restrição), e com um determinado número de espaços visíveis (que pode ser espandido se o título for mais largo, e que tem valor nulo por omissão).

- virtual ~**CaixaDeTexto** ()

  Destrutor polimórfico marca uma classe polimórfica.

Inspectores

- virtual std::string const & **textoActual** () const

  Devolve o texto actual da caixa.

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

  Devolve o título da caixa de texto.

Interface com o utilizador

- virtual void **interage** ()

  Executa a caixa de texto, i.e., interage com o utilizador do programa.

10.4.1 Descrição detalhada

Representa caixas de texto, que quando executadas permitem ao utilizador introduzir uma cadeia de caracteres.

A cadeia introduzida é recordada entre interacções da caixa e pode-se restringir o seu formato impetindo a introdução de cadeias vazias e obrigando à introdução de caracteres de uma lista dada.

Invariante:
titulo <> "" e eImprimivel(titulo) e 1 <= numero_de_espacos_visiveis inicio_da_parte_visivel_do_texto <= texto_corrente.size() e posicao_do_cursor_nos_espacos_visiveis < numero_de_espacos_visiveis.

Tarefa
Permitir a passagem de um functor de verificação de sintaxe, que torne a classe verdadeiramente genérica.

Definido na linha 484 do ficheiro menu.H.

10.4.2 Documentação dos Construtores & Destrutor

10.4.2.1 Slang::CaixaDeTexto::CaixaDeTexto (std::string const & *titulo*, std::string const & *texto_inicial* = "", std::string const & *caracteres_admissiveis* = "", bool *deve_impedir_caixa_vazia* = false, std::string::size_type *espacos* = 0) [inline, explicit]

Constrói uma nova caixa de texto, com um dado título, um dado texto inicial (por omissão ""), um dado conjunto de caracteres admissíveis (por omissão "", que significa que qualquer caracter imprimível é admissível), com a opção de impedir a terminação da interacção com a caixa vazia (por omissão não há qualquer restrição), e com um determinado número de espaços visíveis (que pode ser espandido se o título for mais largo, e que tem valor nulo por omissão).

Precondição:
titulo <> "" e eImprimivel(titulo).

Definido na linha 243 do ficheiro menu_impl.H.

Referências titulo().

10.4.2.2 Slang::CaixaDeTexto::~CaixaDeTexto () [inline, virtual]

Destrutor polimórfico marca uma classe polimórfica.

Precondição:
V.

Definido na linha 270 do ficheiro menu_impl.H.

10.4.3 Documentação dos métodos

10.4.3.1 std::string const & Slang::CaixaDeTexto::textoActual () const [inline, virtual]

Devolve o texto actual da caixa.

Precondição:
V.

Poscondição:
textoActual = texto actual da caixa.

Definido na linha 275 do ficheiro menu_impl.H.

10.4.3.2 std::string Slang::CaixaDeTexto::titulo () const [inline]

Devolve o título da caixa de texto.

Precondição:
V.

Poscondição:
titulo = título da caixa de texto.

Definido na linha 282 do ficheiro menu_impl.H.

Referenciado por CaixaDeTexto().

10.4.3.3 void Slang::CaixaDeTexto::interage () [virtual]

Executa a caixa de texto, i.e., interage com o utilizador do programa.

Precondição:
V.

Definido na linha 154 do ficheiro menu.C.

Referências Slang::campainha(), Slang::Ecra::cola(), Slang::Tecla::comoChar(), Slang::Tecla::eChar(), Slang::ecra, Utilitarios::eImprimivel(), Slang::Ecra::foiRedimensionado(), Slang::fundo(), Slang::Teclado::haTeclaDisponivel(), Slang::Teclado::leProximaTeclaDisponivel(), Slang::Ecra::posicaoDoCursor(), Slang::refresca(), Slang::teclado, Slang::Teclado::teclaLida() e Slang::Ecra::trocoDoEcraCompleto().

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

- **menu.H**
- **menu.C**
- **menu_impl.H**

10.5 Referência à classe Utilitarios::Data

Representa datas posteriores a 1582, i.e., datas no calendário Gregoriano, adoptado por Portugal e outros países católicos em 1582.

`#include <Utilitarios/DataTempo/data.H>`

Constantes estáticas

- **Ano** const **inicio_do_calendario_gregoriano** = 1582
  Constante que guarda o ano de início do calendário Gregoriano:.
- int const **ano_minimo** = **inicio_do_calendario_gregoriano** + 1
  Constante que guarda o ano mínimo das datas (poder-se-ia ser menos estrito, considerando uma data mínima de dia 15 de Outubro de 1582).

Membros públicos

Construtores

- **Data** (**Ano** const ano, **Mes** const mes, **Dia** const dia)
  Constrói uma nova data com o ano, mes e dia dados.
- **Data** ()
  Constrói por omissão uma data com 2003/1/1.
- **Data** (long const dia_juliano)
  Constrói a classe a partir do dia Juliano.

Inspectores

- **Ano** const & **ano** () const
  Devolve o ano correspondente à data.
- **Mes** const & **mes** () const
  Devolve o mes correspondente à data.
- **Dia** const & **dia** () const
  Devolve o dia correspondente à data.
- **DiaDaSemana diaDaSemana** () const
  Devolve o dia da semana correspondente à data.
- long int **diaJuliano** () const
  Devolve o dia Juliano correspondente à data.
- int **numeroDeDiasNoMes** () const
  Devolve o número de dias no mês e ano da data.

Predicados

- bool **anoEBissexto** () const
  Indica se o ano da data é bissexto.

Serializadores

- **Data** (std::istream &entrada)
  Constrói uma data a partir de um canal de entrada.
- void **carregaDe** (std::istream &entrada)
  Carrega uma data a partir de um canal de entrada.
- void **guardaEm** (std::ostream &saida) const
  Guarda os dados da data num canal de saida.

Operadores aritméticos

- Data & **operator++** ()
  Incrementa uma data (versão prefixa).
- Data & **operator–** ()
  Decrementa uma data (versão prefixa).
- Data **operator++** (int)
  Incrementa uma data (versão sufixa).
- Data **operator–** (int)
  Decrementa uma data (versão sufixa).
- Data & **operator+=** (**Duracao** const &duracao)
  Avança uma data de uma dada duração.
- Data & **operator-=** (**Duracao** const &duracao)
  Recua uma data de uma dada duração.

Membros públicos estáticos

Operações de classe

- Data **actual** ()
  Devolve a data actual.
- void **estabeleceDataActualPedidaAoUtilizador** ()
  Faz com que a data actual seja pedida ao utilizador e não obtida do sistema.
- void **estabeleceDataActualObtidaDoSistema** ()
  Faz com que a data actual seja obtida do sistema e não pedida ao utilizador.

10.5.1 Descrição detalhada

Representa datas posteriores a 1582, i.e., datas no calendário Gregoriano, adoptado por Portugal e outros países católicos em 1582.

Esta classe ainda não está completa, pois não permite ainda o acrescento de meses ou anos, apenas de dias.

Note-se que em muitos países o calendário gregoriano foi adoptado mais tarde: 1752 no Reino Unido, por exemplo. Ver `http://www.geocities.com/calendopaedia/gregory.htm` para as datas precisas da mudança em vários países.

Segue abaixo um pequeno exemplo de utilização que se espera seja auto-explicativo:

```
#include <iostream>

#include <Utilitarios/data.H>

using namespace std;

using namespace Utilitarios;

int main()
{
    Data data(1965, setembro, 14);

    cout << "A data é " << data << '.' << endl;

    Data data_actual = Data::actual();

    cout << "A data actual é " << data_actual << '.' << endl;

    cout << "Passaram " << data_actual - data << " dias desde "
         << data << " até " << data_actual << '.' << endl;

    // A partir deste ponto a data actual não é obtida do sistema mas sim
    // pedida ao utilizador.  Usar para depuração!

    Data::estabeleceDataActualPedidaAoUtilizador();

    data_actual = Data::actual();

    cout << "A data actual é " << data_actual << '.' << endl;
}
```

Invariante:
ano_minimo <= ano_ e 0 < dia_ e dia_ <= numeroDeDiasEm(mes_, ano_)

Veja também:
Mes, **Ano**, **Dia**, **DiaDaSemana**

Definido na linha 402 do ficheiro data.H.

10.5.2 Documentação dos Construtores & Destrutor

10.5.2.1 Utilitarios::Data::Data (Ano const *ano*, Mes const *mes*, Dia const *dia*) [inline]

Constrói uma nova data com o ano, mes e dia dados.

Precondição:
ano_minimo <= ano e janeiro <= mes <= fevereiro e 0 <= dia <= numeroDeDiasEm(mes, ano).

Definido na linha 292 do ficheiro data_impl.H.

Referências ano(), Utilitarios::Ano, ano_minimo, Utilitarios::dezembro, dia(), Utilitarios::Dia, Utilitarios::janeiro, mes(), Utilitarios::Mes e Utilitarios::numeroDeDiasEm().

10.5.2.2 Utilitarios::Data::Data () [inline]

Constrói por omissão uma data com 2003/1/1.

Precondição:
V.

Poscondição:
ano() = 2002 e mes()= janeiro e **dia()** = 2.

Definido na linha 302 do ficheiro data_impl.H.

Referências Utilitarios::janeiro.

Referenciado por actual(), carregaDe(), operator+=(), operator–() e operator-=().

10.5.2.3 Utilitarios::Data::Data (long const *dia_juliano*) [inline]

Constrói a classe a partir do dia Juliano.

Precondição:
dia_juliano é mesmo dia Juliano.

Poscondição:
Data tem a data correspondente aos dias da data juliana: **Data(long const dia_juliano)**.

Definido na linha 309 do ficheiro data_impl.H.

Referências Utilitarios::Ano, Utilitarios::Dia e Utilitarios::Mes.

10.5.2.4 Utilitarios::Data::Data (std::istream & *entrada*) [inline]

Constrói uma data a partir de um canal de entrada.

Precondição:
canal.good().

Poscondição:
Data é a data que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se a construção por carregamento falhar.

Definido na linha 385 do ficheiro data_impl.H.

Referências Utilitarios::dezembro, Utilitarios::il(), Utilitarios::janeiro, Utilitarios::Mes e mes().

10.5.3 Documentação dos métodos

10.5.3.1 Utilitarios::Ano const & Utilitarios::Data::ano () const [inline]

Devolve o ano correspondente à data.

Precondição:
V.

Poscondição:
ano = ano correspondente à data.

Definido na linha 333 do ficheiro data_impl.H.

Referenciado por Data().

10.5.3.2 Utilitarios::Mes const & Utilitarios::Data::mes () const [inline]

Devolve o mes correspondente à data.

Precondição:
V.

Poscondição:
mes = mês correspondente à data.

Definido na linha 340 do ficheiro data_impl.H.

Referenciado por Data().

10.5.3.3 Utilitarios::Dia const & Utilitarios::Data::dia () const [inline]

Devolve o dia correspondente à data.

Precondição:
V.

Poscondição:
dia = dia correspondente à data.

Definido na linha 347 do ficheiro data_impl.H.

Referenciado por Data().

10.5.3.4 Utilitarios::DiaDaSemana Utilitarios::Data::diaDaSemana () const [inline]

Devolve o dia da semana correspondente à data.

Precondição:
V.

Poscondição:
diaDaSemana = dia da semana correspondente à data.

Definido na linha 354 do ficheiro data_impl.H.

Referências Utilitarios::DiaDaSemana, diaJuliano() e Utilitarios::numero_total_de_dias_da_semana.

10.5.3.5 long int Utilitarios::Data::diaJuliano () const [inline]

Devolve o dia Juliano correspondente à data.

O dia Juliano é o número total de dias desde o meio dia de 1 de Janeiro de 4713 AC.

Precondição:
V.

Poscondição:
diaJuliano = dia Juliano correspondente à data.

Definido na linha 361 do ficheiro data_impl.H.

Referenciado por diaDaSemana().

10.5.3.6 int Utilitarios::Data::numeroDeDiasNoMes () const [inline]

Devolve o número de dias no mês e ano da data.

Precondição:
V.

Poscondição:
numeroDediasNoMes = numero de diasno mês e ano da data.

Definido na linha 371 do ficheiro data_impl.H.

Referências Utilitarios::numeroDeDiasEm().

10.5.3.7 bool Utilitarios::Data::anoEBissexto () const [inline]

Indica se o ano da data é bissexto.

Precondição:
V.

Poscondição:
anoEBissexto = ano da data é bissexto.

Definido na linha 378 do ficheiro data_impl.H.

Referências Utilitarios::eBissexto().

10.5.3.8 void Utilitarios::Data::carregaDe (std::istream & *entrada*) [inline]

Carrega uma data a partir de um canal de entrada.

Precondição:
canal.good().

Poscondição:
*this é a data que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se o carregamento falhar.

Definido na linha 406 do ficheiro data_impl.H.

Referências Data().

10.5.3.9 void Utilitarios::Data::guardaEm (std::ostream & *saida*) const [inline]

Guarda os dados da data num canal de saida.

Precondição:
canal.good().

Poscondição:
*this é a data que se encontra no canal de entrada.

Excepções:
ErroAoGuardar é lançada se a construção por carregamento falhar.

Definido na linha 416 do ficheiro data_impl.H.

10.5.3.10 Utilitarios::Data & Utilitarios::Data::operator++ () [inline]

Incrementa uma data (versão prefixa).

Precondição:
*this = d.

Poscondição:
operator++ := *this = d + 1.

Definido na linha 429 do ficheiro data_impl.H.

10.5.3.11 Utilitarios::Data & Utilitarios::Data::operator– () [inline]

Decrementa uma data (versão prefixa).

Precondição:
Data(ano_minimo, 1, 1) < *this e *this = d.

Poscondição:
operator– := *this = d - 1.

Definido na linha 440 do ficheiro data_impl.H.

Referências ano_minimo, Data() e Utilitarios::janeiro.

10.5.3.12 Utilitarios::Data Utilitarios::Data::operator++ (int) [inline]

Incrementa uma data (versão sufixa).

Precondição:
*this = d.

Poscondição:
operator++ = d e *this = d + 1.

Definido na linha 452 do ficheiro data_impl.H.

10.5.3.13 Utilitarios::Data Utilitarios::Data::operator– (int) [inline]

Decrementa uma data (versão sufixa).

Precondição:
Data(ano_minimo, 1, 1) < *this e *this = d.

Poscondição:
operator– = d e *this = d - 1.

Definido na linha 464 do ficheiro data_impl.H.

Referências ano_minimo, Data() e Utilitarios::janeiro.

10.5.3.14 Utilitarios::Data & Utilitarios::Data::operator+= (Duracao const & *duracao*) [inline]

Avança uma data de uma dada duração.

Precondição:
Data(ano_minimo, 1, 1) + duracao <= *this e *this = d.

Poscondição:
operator+= := *this = d + duracao.

Definido na linha 478 do ficheiro data_impl.H.

Referências ano_minimo, Data(), Utilitarios::Duracao e Utilitarios::janeiro.

10.5.3.15 Utilitarios::Data & Utilitarios::Data::operator-= (Duracao const & *duracao*) [inline]

Recua uma data de uma dada duração.

Precondição:
Data(ano_minimo, 1, 1) + duracao <= *this e *this = d.

Poscondição:
operator-= := *this = d - duracao.

Definido na linha 490 do ficheiro data_impl.H.

Referências ano_minimo, Data(), Utilitarios::Duracao e Utilitarios::janeiro.

10.5.3.16 Utilitarios::Data Utilitarios::Data::actual () [static]

Devolve a data actual.

A data actual é, por omissão, obtida do sistema aquando a invocação do método de classe. Mas pode ser obtida do utilizador, para efeitos de depuração.

Precondição:
V.

Poscondição:
actual é a data actual no sistema ou, se a data manual estiver activa, é uma data pedida ao utilizador....

Definido na linha 75 do ficheiro data.C.

Referências Data(), Utilitarios::ill() e Utilitarios::Mes.

10.5.3.17 void Utilitarios::Data::estabeleceDataActualPedidaAoUtilizador () [inline, static]

Faz com que a data actual seja pedida ao utilizador e não obtida do sistema.

Útil para depuração.

Precondição:
V.

Poscondição:
Pedidos de data actual pedem valor ao utilizador.

Definido na linha 502 do ficheiro data_impl.H.

10.5.3.18 void Utilitarios::Data::estabeleceDataActualObtidaDoSistema () [inline, static]

Faz com que a data actual seja obtida do sistema e não pedida ao utilizador.

Precondição:
V.

Poscondição:
Pedidos de data actual passam a recorrer ao sistema.

Veja também:
estabeleceDataActualPedidaAoUtilizador().

Definido na linha 507 do ficheiro data_impl.H.

10.5.4 Documentação dos dados membro

10.5.4.1 Ano const Utilitarios::Data::inicio_do_calendario_gregoriano = 1582 [static]

Constante que guarda o ano de início do calendário Gregoriano:.

Definido na linha 614 do ficheiro data.H.

10.5.4.2 int const Utilitarios::Data::ano_minimo = inicio_do_calendario_gregoriano + 1 [static]

Constante que guarda o ano mínimo das datas (poder-se-ia ser menos estrito, considerando uma data mínima de dia 15 de Outubro de 1582).

Definido na linha 619 do ficheiro data.H.

Referenciado por Data(), operator+=(), operator−() e operator-=().

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

- **data.H**
- **data.C**
- **data_impl.H**

10.6 Referência à classe Slang::Dimensao

Representa uma dimensão em células do ecrã.

`#include <Slang++/ecra.H>`

Membros públicos

Construtores

- **Dimensao** (int const numero_de_linhas=0, int const numero_de_colunas=0)
  Constrói uma nova dimensão.
- **Dimensao** (**Posicao** const &posicao)
  Constrói uma nova dimensão à custa de uma posição.

Inspectores

- int **numeroDeLinhas** () const
  Devolve o número de linhas ecrã correspondente à dimensão.
- int **numeroDeColunas** () const
  Devolve o número de colunas ecrã correspondente à dimensão.

Predicados

- bool **eCanonica** () const
  Indica se a dimensão é canónica, i.e., se ambas as componentes são não-negativas.

Modificadores

- void **mudaNumeroDeLinhasPara** (int const novo_numero_de_linhas)
  Muda o número de linhas do ecrã correspondentes à dimensão para novo_numero_de_linhas.
- void **mudaNumeroDeColunasPara** (int const novo_numero_de_colunas)
  Muda o número de colunas do ecrã correspondentes à dimensão para novo_numero_de_colunas.

Serializadores

- **Dimensao** (std::istream &entrada)
  Constrói uma dimensão por carregamento a partir de um canal de entrada.
- void **carregaDe** (std::istream &entrada)
  Carrega a dimensão a partir de um canal.
- void **guardaEm** (std::ostream &saida) const
  Guarda a dimensão num canal.

Operadores

- Dimensao & **operator+=** (Dimensao const &dimensao)
  Adiciona a dimensão de outra dimensão.
- Dimensao & **operator-=** (Dimensao const &dimensao)
  Subtrai a dimensão de outra dimensão.

Funções associadas

(Note que não são funções membro)

- Dimensao const **operator+** (Dimensao uma_dimensao, Dimensao const &outra_dimensao)
  Devolve a adição de duas dimensões.
- Dimensao const **operator-** (Dimensao uma_dimensao, Dimensao const &outra_dimensao)
  Devolve a subtracção de duas dimensões.
- Dimensao const **operator-** (Dimensao const &dimensao)
  Devolve o simétrico de uma dimensao.
- bool **operator==** (Dimensao const &uma_dimensao, Dimensao const &outra_dimensao)
  Indica se duas dimensões são iguais.
- bool **operator!=** (Dimensao const &uma_dimensao, Dimensao const &outra_dimensao)
  Indica se duas dimensões são diferentes.

10.6.1 Descrição detalhada

Representa uma dimensão em células do ecrã.

É tipicamente usada para representar a dimensão de caixas. As dimensões são expressas em número de linhas e colunas. Qualquer das componentes de uma dimensão pode tomar valores negativos. Quando isso acontecer diz-se que a dimensão é "não-canónica".

As dimensões também podem ser vistas como vectores. Daí que possam ser adicionadas a posições. Da mesma forma, a subtracção entre duas posições é uma dimensão.

Invariante:
V.

Definido na linha 308 do ficheiro ecra.H.

10.6.2 Documentação dos Construtores & Destrutor

10.6.2.1 Slang::Dimensao::Dimensao (int const *numero_de_linhas* = 0, int const *numero_de_colunas* = 0) [inline, explicit]

Constrói uma nova dimensão.

Por omissão a dimensão é nula. Pode haver dimensões negativas.

Precondição:
V.

Poscondição:
numeroDeLinhas() = *numero_de_linhas* e **numeroDeColunas()** = *numero_de_colunas.*

Definido na linha 160 do ficheiro ecra_impl.H.

Referenciado por carregaDe().

10.6.2.2 Slang::Dimensao::Dimensao (Posicao const & *posicao*) [inline, explicit]

Constrói uma nova dimensão à custa de uma posição.

Define uma conversão explícita entre posições e dimensões. A dimensão é a dimensão de uma caixa com canto superior esquerdo na célula (0, 0) e com canto inferior direito na célula imediatamente acima e à esquerda da posição dada.

Precondição:
V.

Poscondição:
numeroDeLinhas() = posicao.linha() e **numeroDeColunas()** = posicao.coluna().

Definido na linha 169 do ficheiro ecra_impl.H.

10.6.2.3 Slang::Dimensao::Dimensao (std::istream & *entrada*) [explicit]

Constrói uma dimensão por carregamento a partir de um canal de entrada.

Assume-se que os dados no canal têm um formato equivalente ao produzido pela operação **guardaEm()**.

Precondição:
entrada.good().

Poscondição:
Dimensao é a dimensão que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se a construção por carregamento falhar.

Definido na linha 39 do ficheiro ecra.C.

10.6.3 Documentação dos métodos

10.6.3.1 int Slang::Dimensao::numeroDeLinhas () const [inline]

Devolve o número de linhas ecrã correspondente à dimensão.

Precondição:
V.

Poscondição:
numeroDeLinhas() = número de linhas a que a dimensão corresponde.

Definido na linha 177 do ficheiro ecra_impl.H.

Referenciado por Slang::Ecra::cola(), Slang::Ecra::desenha(), eCanonica(), Slang::Ecra::moveCursorPara(), operator+=(), Slang::Posicao::operator+=(), operator-=(), Slang::Posicao::operator-=(), Slang::Ecra::operator>>() e Slang::Ecra::trocoDe().

10.6.3.2 int Slang::Dimensao::numeroDeColunas () const [inline]

Devolve o número de colunas ecrã correspondente à dimensão.

Precondição:
V.

Poscondição:
numeroDeColunas() = número de colunas a que a dimensão corresponde.

Definido na linha 184 do ficheiro ecra_impl.H.

Referenciado por Slang::Ecra::cola(), Slang::Ecra::desenha(), eCanonica(), Slang::Ecra::moveCursorPara(), operator+=(), Slang::Posicao::operator+=(), operator-=(), Slang::Posicao::operator-=(), Slang::Ecra::operator>>() e Slang::Ecra::trocoDe().

10.6.3.3 bool Slang::Dimensao::eCanonica () const [inline]

Indica se a dimensão é canónica, i.e., se ambas as componentes são não-negativas.

Precondição:
V.

Poscondição:
eCanonica() = (0 <= **numeroDeLinhas()** and 0 <= numeroDeColunas()).

Definido na linha 191 do ficheiro ecra_impl.H.

Referências numeroDeColunas() e numeroDeLinhas().

Referenciado por Slang::Caixa::eCanonica(), Slang::Ecra::Troco::Troco() e Slang::Ecra::trocoNoCursorCom().

10.6.3.4 void Slang::Dimensao::mudaNumeroDeLinhasPara (int const *novo_numero_de_linhas*) [inline]

Muda o número de linhas do ecrã correspondentes à dimensão para *novo_numero_de_linhas*.

Precondição:
V.

Poscondição:
numeroDeLinhas() = *novo_numero_de_linhas*.

Definido na linha 199 do ficheiro ecra_impl.H.

10.6.3.5 void Slang::Dimensao::mudaNumeroDeColunasPara (int const *novo_numero_de_colunas*) [inline]

Muda o número de colunas do ecrã correspondentes à dimensão para *novo_numero_de_colunas.*

Precondição:
V.

Poscondição:
numeroDeColunas() = *novo_numero_de_colunas.*

Definido na linha 209 do ficheiro ecra_impl.H.

10.6.3.6 void Slang::Dimensao::carregaDe (std::istream & *entrada*) [inline]

Carrega a dimensão a partir de um canal.

Assume-se que os dados no canal têm um formato equivalente ao produzido pela operação **guardaEm()**.

Precondição:
entrada.good().

Poscondição:
*this é a dimensão que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se o carregamento falhar. Nesse caso o canal pode ficar parcialmente lido.

Definido na linha 218 do ficheiro ecra_impl.H.

Referências Dimensao().

10.6.3.7 void Slang::Dimensao::guardaEm (std::ostream & *saida*) const

Guarda a dimensão num canal.

O formato produzido é compatível com o que o método **carregaDe()** espera.

Precondição:
saida.good().

Excepções:
ErroAoGuardar é lançada se a operação falhar. Nesse caso o canal pode ficar parcialmente escrito.

Definido na linha 52 do ficheiro ecra.C.

Referenciado por Slang::Caixa::guardaEm().

10.6.3.8 Slang::Dimensao & Slang::Dimensao::operator+= (Dimensao const & *dimensao*) [inline]

Adiciona a dimensão de outra dimensão.

A dimensão resultante é a soma da original com a dimensão a somar, em cada uma das direcções.

Precondição:
*this = p.

Poscondição:
numeroDeLinhas() = p.numeroDeLinhas() + dimensao.numeroDeLinhas() e **numeroDeColunas()** = p.numeroDeColunas() + dimensao.numeroDeColunas().

Definido na linha 227 do ficheiro ecra_impl.H.

Referências cumpreInvariante(), numeroDeColunas() e numeroDeLinhas().

10.6.3.9 Slang::Dimensao & Slang::Dimensao::operator-= (Dimensao const & *dimensao*) [inline]

Subtrai a dimensão de outra dimensão.

A dimensão resultante é a subtracção da original da dimensão a subtrair, em cada uma das direcções.

Precondição:
*this = p.

Poscondição:
numeroDeLinhas() = p.numeroDeLinhas() - dimensao.numeroDeLinhas() e **numeroDeColunas()** = p.numeroDeColunas() - dimensao.numeroDeColunas().

Definido na linha 239 do ficheiro ecra_impl.H.

Referências cumpreInvariante(), numeroDeColunas() e numeroDeLinhas().

10.6.4 Documentação das classes amigas e funções relacionadas

10.6.4.1 Dimensao const operator+ (Dimensao *uma_dimensao*, Dimensao const & *outra_dimensao*) [related]

Devolve a adição de duas dimensões.

Precondição:
uma_dimensao = d.

Poscondição:
operator+.**numeroDeLinhas()** = d.numeroDeLinhas() + outra_dimensao.numeroDeLinhas() e operator+.**numeroDeColunas()** = d.numeroDeColunas() + outra_dimensao.numeroDeColunas().

10.6.4.2 Dimensao const operator- (Dimensao *uma_dimensao*, Dimensao const & *outra_dimensao*) [related]

Devolve a subtracção de duas dimensões.

Precondição:
uma_dimensao = d.

Poscondição:
operator-.**numeroDeLinhas()** = d.numeroDeLinhas() - outra_dimensao.numeroDeLinhas() e operator-.**numeroDeColunas()** = d.numeroDeColunas() - outra_dimensao.numeroDeColunas().

10.6.4.3 Dimensao const operator- (Dimensao const & *dimensao*) [related]

Devolve o simétrico de uma dimensao.

Precondição:
V.

Poscondição:
operator-.**numeroDeLinhas()** = -dimensao.**numeroDeLinhas()** e operator-.**numeroDeColunas()** = -dimensao.**numeroDeColunas()**.

10.6.4.4 bool operator== (Dimensao const & *uma_dimensao*, Dimensao const & *outra_dimensao*) [related]

Indica se duas dimensões são iguais.

Precondição:
V.

Poscondição:
operator== = (uma_dimensao.**numeroDeLinhas()** = outra_dimensao.numeroDeLinhas() e uma_dimensao.numeroDeColunas() = outra_dimensao.numeroDeColunas()).

10.6.4.5 bool operator!= (Dimensao const & *uma_dimensao*, Dimensao const & *outra_dimensao*) [related]

Indica se duas dimensões são diferentes.

Precondição:
V.

Poscondição:
operator!= = (uma_dimensao.**numeroDeLinhas()** <> outra_dimensao.numeroDeLinhas() ou uma_dimensao.numeroDeColunas() <> outra_dimensao.numeroDeColunas()).

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

- **ecra.H**
- **ecra.C**
- **ecra_impl.H**

10.7 Referência à classe Slang::Ecra

Representa o ecrã.

`#include <Slang++/ecra.H>`

Membros públicos

Construtores

- **Ecra** (**Cor** cor_do_texto=branco, **Cor** cor_do_fundo=preto, bool limita_cursor_ao_ecra_real=false)
  Constrói um ecrã.
- ~**Ecra** ()
  Destrutor da classe.

Inspectores

- **Dimensao** const **dimensao** () const
  Devolve a dimensão corrente do ecrã real.
- **Posicao** const **origem** () const
  Devolve a posição da origem do ecrã real no ecrã virtual.
- **Posicao** const **extremo** () const
  Devolve a posição do extremo do ecrã real no ecrã virtual.
- **Posicao** const **posicaoDoCursor** () const
  Devolve a posição corrente do cursor.
- **Justificacao** **justificacaoActual** () const
  Devolve a justificação actual das inserções no ecrã.
- int **larguraDaProximaOperacaoDeEscrita** () const
  Devolve a largura a usar na próxima inserção no ecrã.

Predicados

- bool **foiRedimensionado** () const
  Indica se o ecrã real foi redimensionado ou não.
- bool **cursorEstaVisivel** () const
  Indica se o cursor está sobre o ecrã real, i.e., se o cursor é visível.
- bool **parteVisivelContem** (**Posicao** const &posicao) const
  Indica se uma posição está sobre o ecrã real, i.e., se a parte visível do ecrã contém essa posição.
- bool **cursorEstaLimitadoAoEcraReal** () const
  Indica se o cursor está limitado ao ecrã real, i.e., se o ecrã proíbe o cursor de se deslocar para fora da sua parte visível.

- bool **cursorImovelNaProximaOperacaoDeInsercao** () const
  Indica se o cursor ficará imóvel na próxima operação de inserção.

Modificadores

- void **mudaJustificacaoPara** (**Justificacao** const nova_justificacao)
  Muda a justificação do ecrã para a justificação dada.
- void **impoeImobilidadeDoCursorNaProximaInsercao** ()
  Coloca o ecrã num estado tal que a próxima operação de inserção não tem efeito sobre a posição do cursor.
- void **mudaLarguraDaProximaOperacaoDeEscritaPara** (int const largura_da_proxima_operacao_de_escrita)
  Coloca o ecrã num estado tal que a próxima operação de inserção será feita usando exactamente largura_da_proxima_operacao_de_escrita *células do ecrã.*
- void **mudaObjectoCorEmUsoPara** (**ObjectoCor** const &objecto_cor)
  Muda o objecto cor em uso para objecto_cor.
- void **mudaObjectoCorEmUsoParaFundo** ()
  Muda o objecto cor em uso para o objecto cor do fundo.
- void **mudaCoresDasCelulasDoFundoPara** (**Cor** const cor_do_texto, **Cor** const cor_do_fundo)
  Modifica as cores das células do fundo.
- void **moveCursorPara** (**Posicao** const &nova_posicao_do_cursor)
  Desloca o cursor para a posição dada.
- void **sobeCursor** ()
  Sobe o cursor uma linha.
- void **baixaCursor** ()
  Baixa o cursor uma linha.
- void **recuaCursor** ()
  Recua o cursor uma coluna (para a esquerda).
- void **avancaCursor** ()
  Avança o cursor uma coluna (para a direita).
- void **moveCursorDeAcordoCom** (**Tecla** const &tecla)
  Desloca o cursor na direcção dada por uma tecla de deslocamento.

Operações para cópia e cola de troços de ecrã

- **Troco** const **trocoDe** (**Caixa** const &caixa) const
  Devolve um troço do ecrã correspondente à caixa dada.
- **Troco** const **trocoDoEcraCompleto** () const
  Devolve uma cópia integral do ecrã real.

- **Troco** const **trocoNoCursorCom** (**Dimensao** const &dimensao) const
  Devolve uma cópia do troço de ecrã com uma dada dimensão e com origem na posição corrente do cursor.

- void **cola** (**Troco** const &troco, **Posicao** const &em=**Posicao**())
  Cola no ecrã, na posição indicada (por omissão o canto superior esquerdo do ecrã), um troço de ecrã.

Operações para actualizar o ecrã

- void **tocaCampainha** () const
  Toca a "campaínha" do ecrã.

- void **apagaDoCursorAteOFimDaLinha** ()
  Apaga (i.e., pinta com cor do fundo) desde o cursor até ao fim da respectiva linha.

- void **apagaDoCursorAteOFim** ()
  Apaga (i.e., pinta com cor do fundo) desde o cursor até ao fim do ecrã.

- void **apaga** ()
  Apaga (i.e., pinta com cor do fundo) todo o ecrã.

- void **refresca** () const
  Refresca o ecrã, ou seja, reflecte no ecrã real as últimas alterações realizadas no ecrã virtual.

- void **refrescaTudo** () const
  Refresca totalmente o ecrã, ou seja, reflecte no ecrã real todo o ecrã virtual.

- void **desenha** (**Caixa** const &caixa)
  Desenha uma caixa no ecrã, sendo a borda desenhada com caracteres especiais.

- void **desenhaSegmentoHorizontalCom** (int const largura)
  Desenha um segmento de recta horizontal, com a largura dada, a partir da posição do cursor (para a direita).

- void **desenhaSegmentoVerticalCom** (int const altura)
  Desenha um segmento de recta vertical, com a altura dada, a partir da posição do cursor (para baixo).

Operadores de inserção no ecrã de tipos usuais

- Ecra & **operator**<< (std::string const &cadeia)
  Insere uma cadeia de caracteres na posição do cursor.

- template<typename T> Ecra & **operator**<< (T const &valor)
  Insere um valor do tipo T genérico na posição do cursor.

Operadores de inserção e extracção de objectos especiais no *ecrã*

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

Desenha uma caixa no ecrã.

- Ecra & **operator**<< (**Simbolo** const simbolo)
  Desenha um símbolo especial no ecrã.

- Ecra & **operator**<< (**Troco** const &troco)
  Cola no ecrã, na posição do cursor, um troço de ecrã.

Operadores de extracção de objectos especiais do ecrã

- Ecra const & **operator**>> (**Troco** &troco) const
  Extrai um troço do ecrã a partir da posição actual do cursor.

Operadores de inserção de manipuladores no ecrã

Veja-se a descrição dos correspondentes manipuladores, onde se explica o respectivo efeito.

- Ecra & **operator**<< (**Posicao** const &nova_posicao_do_cursor)
  Altera a posição do cursor para a posição dada (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

- Ecra & **operator**<< (**ObjectoCor** const &objecto_cor)
  Altera o objecto cor das próximas operações de inserção no ecrã (com excepção das colagens!).

- Ecra & **operator**<< (**Justificacao** const nova_justificacao)
  Muda a justificação do ecrã para a justificação dada.

- Ecra & **operator**<< (void manipulador(Ecra &**ecra**))
  "Gancho" para definir novos manipuladores do ecrã.

Amigos

- class **ObjectoCor**

10.7.1 Descrição detalhada

Representa o ecrã.

Esta classe é um solitão, pois admite-se não haver senão um ecrã.

Por ecrã entende-se um conjunto de células cada uma das quais contém um caractere com uma dada cor de texto e uma dada cor de fundo. Conceptualmente o ecrã estende-se em ambas as direcções até ao infinito, que vertical quer horizontalmente. No entanto, só uma parte do ecrã é visível, dependendo da dimensão do dispositivo onde ocorrerá a sua visualização. Por exemplo, se o ecrã estiver a ser visualizado na janela de uma consola (e.g., `xterm`) com 24 linhas e 80 colunas, só serão visíveis as linhas 0 a 23 do ecrã e as colunas 0 a 79. chamar-se ecrã real ao dispositivo no qual o ecrã (virtual) é visualizado ou, pelo menos, à zona do ecrã virtual que é visível no ecrã real.

O ecrã é virtual. As alterações nele feitas só têm efeito no ecrã real depois de uma operação de refrescamento. Veja-se a operação **refresca()** ou o manipulador refresca.

O ecrã permite várias operações:

1. inspeccionar a dimensão do ecrã real e a posição do cursor no ecrã (virtual);
2. verificar se o ecrã real foi redimensionado;
3. modificar a posição do cursor (de várias formas diferentes);
4. copiar ou colar um troço de/para uma posição do ecrã (um troço é um conjunto rectangular de células do ecrã);
5. apagar várias partes do ecrã;
6. refrescar o ecrã real (ou seja, mostrar ou ecrã virtual);
7. desenhar uma caixa no ecrã;
8. enviar para o ecrã um caractere, um inteiro, uma cadeia de caracteres, uma caixa, um troço ou um dos manipuladores definidos;
9. etc.

Refrescamento

As alterações só são visíveis depois de chamada uma das operações de refrescamento do ecrã. Quando há um redimensionamento do ecrã deve ser invocada a operação **refrescaTudo()** ou usado o manipulador refresca_tudo. Quando são feitas pequenas alterações deve ser invocada a operação **refresca()** ou usado o manipulador refresca.

Exemplo de utilização

O programa que se segue escreve no ecrã todas as letras do alfabeto (maiúsculas e minúsculas) e depois, a cada pressão de uma tecla, escreve os dígitos de zero a nove com outra cor, com largura proporcional ao dígito escrito e centrados.

```
#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.leProximaTeclaDisponivel();

    Ecra::ObjectoCor cor(amarelo, azul);

    ecra << cor << cursor(0, 0);
    for(int c = 0; c != 10; ++c) {
        ecra << baixaCursor << parado << largura(c) << ao_centro << c
             << refresca;

        teclado.leProximaTeclaDisponivel();
    }
}
```

Para mais exemplos ver documentação das operações específicas e das classe auxiliares.

Definido na linha 1103 do ficheiro ecra.H.

10.7.2 Documentação dos Construtores & Destrutor

10.7.2.1 Slang::Ecra::Ecra (Cor *cor_do_texto* = branco, Cor *cor_do_fundo* = preto, bool *limita_cursor_ao_ecra_real* = false) [explicit]

Constrói um ecrã.

Encarrega-se de inicializar o "screen management" do S-Lang. Se se construir com limita_cursor_ao_ecra_real verdadeiro, o ecrã garantirá que o cursor se encontra dentro dos limites do ecrã real em cada instante. Por omissão essa verificação não é feita, ou seja, o cursor pode-se deslocar para fora do ecrã real. O construtor recebe também as cores do texto e do fundo das chamadas "células do fundo", para as quais não se explicita nenhuma cor.

Tal como a biblioteca se encontra definida, este construtor não pode ser usado directamente, pois a classe é um solitão e a biblioteca pré-define uma variável global deste tipo. No entanto, numa versão futura, onde não exista variável global para representar o ecrã, o programador utilizador pode vir a usá-lo.

Precondição:
V.

Poscondição:
Ecra é um ecrã pronto a utilizar.

Definido na linha 93 do ficheiro ecra.C.

Referências Slang::a_esquerda, Slang::Cor e mudaCoresDasCelulasDoFundoPara().

10.7.2.2 Slang::Ecra::~Ecra () [inline]

Destrutor da classe.

Encarrega-se de finalizar o "screen management" do S-Lang.

Precondição:
V.

Definido na linha 542 do ficheiro ecra_impl.H.

10.7.3 Documentação dos métodos

10.7.3.1 Slang::Dimensao const Slang::Ecra::dimensao () const [inline]

Devolve a dimensão corrente do ecrã real.

No programa abaixo vê-se uma utilização deste inspector para controlar a dimensão de uma caixa inserida no ecrã.

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

using namespace Slang;

// Mostra uma caixa com 1/4 da área do ecrã (metade das suas dimensões
// horizontal e vertical) centrada no ecrã.
int main()
```

```
{
    Caixa uma_caixa(Posicao(ecra.dimensao().numeroDeLinhas() / 4,
                            ecra.dimensao().numeroDeColunas() / 4),
                    Dimensao(ecra.dimensao().numeroDeLinhas() / 2,
                             ecra.dimensao().numeroDeColunas() / 2));

    ecra << uma_caixa << refresca;

    teclado.leProximaTeclaDisponivel();
}
```

Precondição:
V.

Poscondição:
dimensao = dimensão do ecrã real.

Definido na linha 550 do ficheiro ecra_impl.H.

Referenciado por extremo(), moveCursorPara(), parteVisivelContem(), refrescaTudo(), Slang::Ecra::Troco::Troco() e trocoDoEcraCompleto().

10.7.3.2 Slang::Posicao const Slang::Ecra::origem () const [inline]

Devolve a posição da origem do ecrã real no ecrã virtual.

De momento devolve sempre (0, 0).

Precondição:
V.

Poscondição:
origem = origem do ecrã real, de momento sempre (0, 0).

Definido na linha 559 do ficheiro ecra_impl.H.

Referenciado por extremo() e parteVisivelContem().

10.7.3.3 Slang::Posicao const Slang::Ecra::extremo () const [inline]

Devolve a posição do extremo do ecrã real no ecrã virtual.

De momento devolve sempre **dimensao()** - Dimensao(1, 1).

Precondição:
V.

Poscondição:
extremo = extremo do ecrã real, de momento sempre (dimensao().numeroDeLinhas() - 1, **dimensao()**.numeroDeColunas() - 1).

Definido na linha 568 do ficheiro ecra_impl.H.

Referências dimensao() e origem().

10.7.3.4 Slang::Posicao const Slang::Ecra::posicaoDoCursor () const [inline]

Devolve a posição corrente do cursor.

O programa de exemplo abaixo permite deslocar o cursor de acordo com as teclas premidas, dizendo em cada instante se o cursor se encontra dentro ou fora do ecrã. Para isso recorre à operação **posicaoDoCursor()**, pois precisa de recolocar o cursor depois de uma operação de escrita:

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

using namespace std;
using namespace Slang;

int main()
{
    while(true) {
        if(teclado.haTeclaDisponivel(10)) {
            teclado.leProximaTeclaDisponivel();
            Tecla tecla_premida = teclado.teclaLida();

            if(tecla_premida.eDeDeslocamento())
                ecra.moveCursorDeAcordoCom(tecla_premida);
            else if(tecla_premida == 's')
                break;
        }

        if(ecra.foiRedimensionado())
            ecra << apaga;

        string mensagem;
        if(ecra.cursorEstaVisivel())
            mensagem = "Visível";
        else
            mensagem = "Invisível";

        Posicao posicao_do_cursor = ecra.posicaoDoCursor();

        ecra << cursor(ecra.dimensao().numeroDeLinhas() / 2, 0)
             << largura(ecra.dimensao().numeroDeColunas()) << ao_centro
             << mensagem;

        ecra << posicao_do_cursor << refresca;
    }
}
```

Precondição:
V.

Poscondição:
posicaoDoCursor = posição corrente do cursor.

Definido na linha 577 do ficheiro ecra_impl.H.

Referenciado por avancaCursor(), baixaCursor(), cola(), cursorEstaVisivel(), desenha(), desenhaSegmentoHorizontalCom(), desenhaSegmentoVerticalCom(), Slang::CaixaDeTexto::interage(), Slang::MenuSimples::interage(), operator<<(), operator>>(), recuaCursor(), sobeCursor(), trocoDe() e trocoNoCursorCom().

10.7.3.5 Slang::Justificacao Slang::Ecra::justificacaoActual () const [inline]

Devolve a justificação actual das inserções no ecrã.

Precondição:
V.

Poscondição:
justificacaoActual = justificação actual do ecrã.

Definido na linha 587 do ficheiro ecra_impl.H.

10.7.3.6 int Slang::Ecra::larguraDaProximaOperacaoDeEscrita () const [inline]

Devolve a largura a usar na próxima inserção no ecrã.

Note-se que o valor 0 significa que não se usa largura na inserção.

Precondição:
V.

Poscondição:
larguraDaProximaOperacaoDeEscrita = largura a usar na próxima inserção no ecrã.

Definido na linha 596 do ficheiro ecra_impl.H.

10.7.3.7 bool Slang::Ecra::foiRedimensionado () const [inline]

Indica se o ecrã real 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. Note-se que conceptualmente este predicado pode mudar de valor devolvido sem que tenha sido invocado nenhum modificador. Note-se ainda que logo após devolver verdadeiro devolverá falso até ocorrer novo redimensionamento.

É esta operação que permite ao programa

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

using namespace std;
using namespace Slang;

int main()
{
    while(true) {
        if(teclado.haTeclaDisponivel(10)) {
            teclado.leProximaTeclaDisponivel();
            Tecla tecla_premida = teclado.teclaLida();

            if(tecla_premida.eDeDeslocamento())
                ecra.moveCursorDeAcordoCom(tecla_premida);
            else if(tecla_premida == 's')
                break;
```

```
        }

        if(ecra.foiRedimensionado())
            ecra << apaga;

        string mensagem;
        if(ecra.cursorEstaVisivel())
            mensagem = "Visivel";
        else
            mensagem = "Invisivel";

        Posicao posicao_do_cursor = ecra.posicaoDoCursor();

        ecra << cursor(ecra.dimensao().numeroDeLinhas() / 2, 0)
             << largura(ecra.dimensao().numeroDeColunas()) << ao_centro
             << mensagem;

        ecra << posicao_do_cursor << refresca;
    }
}
```

adaptar-se a alterações na dimensão do ecrã real.

Precondição:
V.

Poscondição:
foiRedimensionado = ecrã real foi redimensionado deste a construção do ecrã ou desde a última invocação do predicado e não redimensionado.

Definido na linha 605 do ficheiro ecra_impl.H.

Referenciado por Slang::CaixaDeTexto::interage() e Slang::MenuSimples::interage().

10.7.3.8 bool Slang::Ecra::cursorEstaVisivel () const [inline]

Indica se o cursor está sobre o ecrã real, i.e., se o cursor é visível.

Note-se que quando o cursor está "invisível" é um seu "fantasma" visível no ecrã real. A biblioteca S-Lang não permite ocultar este "fantasma", infelismente.

Na documentação de **posicaoDoCursor()** encontra-se um programa de exemplo que faz uso deste predicado.

Precondição:
V.

Poscondição:
cursorEstaVisivel = parteVisivelContem(posicaoDoCursor()).

Definido na linha 619 do ficheiro ecra_impl.H.

Referências parteVisivelContem() e posicaoDoCursor().

10.7.3.9 bool Slang::Ecra::parteVisivelContem (Posicao const & *posicao*) const [inline]

Indica se uma posição está sobre o ecrã real, i.e., se a parte visível do ecrã contém essa posição.

Precondição:
V.

Poscondição:
parteVisivelContem = Caixa(origem(), extremo()).contem(posicao).

Definido na linha 628 do ficheiro ecra_impl.H.

Referências dimensao() e origem().

Referenciado por cursorEstaVisivel().

10.7.3.10 bool Slang::Ecra::cursorEstaLimitadoAoEcraReal () const [inline]

Indica se o cursor está limitado ao ecrã real, i.e., se o ecrã proíbe o cursor de se deslocar para fora da sua parte visível.

Precondição:
V.

Poscondição:
cursorEstaLimitadoAoEcraReal = cursor está proibído de abandonar a parte visível do ecrã.

Definido na linha 637 do ficheiro ecra_impl.H.

10.7.3.11 bool Slang::Ecra::cursorImovelNaProximaOperacaoDeInsercao () const [inline]

Indica se o cursor ficará imóvel na próxima operação de inserção.

Precondição:
V.

Poscondição:
cursorImovelNaProximaOperacaoDeInsercao = cursor ficará imóvel na próxima operação de inserção.

Definido na linha 647 do ficheiro ecra_impl.H.

10.7.3.12 void Slang::Ecra::mudaJustificacaoPara (Justificacao const *nova_justificacao*) [inline]

Muda a justificação do ecrã para a justificação dada.

A justificação mantém-se até à próxima alteração.

Precondição:
V.

Poscondição:
justificacaoActual() = nova_justificacao

Definido na linha 657 do ficheiro ecra_impl.H.

Referências Slang::Justificacao.

Referenciado por operator<<().

10.7.3.13 void Slang::Ecra::impoeImobilidadeDoCursorNaProximaInsercao () [inline]

Coloca o ecrã num estado tal que a próxima operação de inserção não tem efeito sobre a posição do cursor.

Precondição:
V.

Poscondição:
cursorImovelNaProximaOperacaoDeInsercao().

Definido na linha 668 do ficheiro ecra_impl.H.

10.7.3.14 void Slang::Ecra::mudaLarguraDaProximaOperacaoDeEscritaPara (int const *largura_da_proxima_operacao_de_escrita*) [inline]

Coloca o ecrã num estado tal que a próxima operação de inserção será feita usando exactamente *largura_da_proxima_operacao_de_escrita* células do ecrã.

Precondição:
0 <= largura_da_proxima_operacao_de_escrita.

Poscondição:
LarguraDaProximaOperacaoDeEscrita() = *largura_da_proxima_operacao_de_escrita*

Veja também:
largura()

Definido na linha 680 do ficheiro ecra_impl.H.

Referenciado por Slang::operator<<().

10.7.3.15 void Slang::Ecra::mudaObjectoCorEmUsoPara (ObjectoCor const & *objecto_cor*) [inline]

Muda o objecto cor em uso para *objecto_cor*.

Útil para alterar as cores das próximas inserções no ecrã. Normalmente usa-se o correspondente operador de inserção para obter o mesmo efeito. As duas formas de utilização de objectos cor estão demonstradas no programa de exemplo abaixo:

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

using namespace Slang;

int main()
{
    Ecra::ObjectoCor cor1(amarelo, azul);
    Ecra::ObjectoCor cor2(azul, amarelo);

    ecra << parado << "Cor do fundo..." << baixaCursor;
    ecra << cor1 << parado << "Amarelo sobre azul..." << baixaCursor;
    ecra << fundo << parado << "De novo cor do fundo..." << baixaCursor;
```

```
    ecra.mudaObjectoCorEmUsoPara(cor2);
    ecra << parado << "Azul sobre amarelo..." << baixaCursor;

    ecra << refresca;

    teclado.leProximaTeclaDisponivel();
}
```

Precondição:
V.

Poscondição:
As próximas inserções passarão a ser realizadas usando o objecto cor *objecto_cor*.

Definido na linha 693 do ficheiro ecra_impl.H.

Referências Slang::Ecra::ObjectoCor::numero_do_objecto.

Referenciado por operator<<().

10.7.3.16 void Slang::Ecra::mudaObjectoCorEmUsoParaFundo () [inline]

Muda o objecto cor em uso para o objecto cor do fundo.

Útil para que as próximas inserções sejam feitas como células do fundo.

Veja-se o exemplo na documentação da operação **mudaObjectoCorEmUsoPara()**.

Precondição:
V.

Poscondição:
As próximas escritas passarão a ser realizadas usando o objecto cor do fundo.

Definido na linha 704 do ficheiro ecra_impl.H.

10.7.3.17 void Slang::Ecra::mudaCoresDasCelulasDoFundoPara (Cor const *cor_do_texto*, Cor const *cor_do_fundo*) [inline]

Modifica as cores das células do fundo.

Útil para modificar o aspecto do fundo mesmo quando não se tem acesso ao construtor do ecrã.

Um exemplo de utilização desta operação pode ser visto abaixo:

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

using namespace Slang;

int main()
{
    ecra << parado << "Cor do fundo..." << baixaCursor;

    Ecra::ObjectoCor cor(branco, verde);
    ecra << cor << parado << "Branco sobre verde..." << baixaCursor;

    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
```

```
    // A cor do texto já inserido como fundo no ecrã mudará:
    ecra.mudaCoresDasCelulasDoFundoPara(vermelho, branco);

    ecra << refresca;
    teclado.leProximaTeclaDisponivel();

    ecra << fundo << parado
         << "Mais com a cor do fundo, que passou a vermelho sobre branco...";

    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
}
```

Precondição:
V.

Poscondição:
Células desenhadas com o objecto cor do fundo serão desenhadas com as cores dadas.

Definido na linha 716 do ficheiro ecra_impl.H.

Referências Slang::Cor.

Referenciado por Ecra().

10.7.3.18 void Slang::Ecra::moveCursorPara (Posicao const & *nova_posicao_do_cursor*)

Desloca o cursor para a posição dada.

Precondição:
V.

Poscondição:
posicaoDoCursor() = nova_posicao_do_cursor (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Definido na linha 123 do ficheiro ecra.C.

Referências Slang::Posicao::coluna(), dimensao(), Slang::Posicao::linha(), Slang::Dimensao::numeroDeColunas() e Slang::Dimensao::numeroDeLinhas().

Referenciado por avancaCursor(), baixaCursor(), cola(), desenha(), desenhaSegmentoHorizontalCom(), desenhaSegmentoVerticalCom(), operator<<(), operator>>(), recuaCursor(), sobeCursor() e trocoDe().

10.7.3.19 void Slang::Ecra::sobeCursor () [inline]

Sobe o cursor uma linha.

Precondição:
p = **posicaoDoCursor()**.

Poscondição:
posicaoDoCursor() = p - Dimensao(1, 0) (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Definido na linha 729 do ficheiro ecra_impl.H.

Referências moveCursorPara() e posicaoDoCursor().

Referenciado por moveCursorDeAcordoCom().

10.7.3.20 void Slang::Ecra::baixaCursor () [inline]

Baixa o cursor uma linha.

Precondição:
p = **posicaoDoCursor()**.

Poscondição:
posicaoDoCursor() = p + Dimensao(1, 0) (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Definido na linha 740 do ficheiro ecra_impl.H.

Referências moveCursorPara() e posicaoDoCursor().

Referenciado por moveCursorDeAcordoCom().

10.7.3.21 void Slang::Ecra::recuaCursor () [inline]

Recua o cursor uma coluna (para a esquerda).

Precondição:
p = **posicaoDoCursor()**.

Poscondição:
posicaoDoCursor() = p - Dimensao(0, 1) (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Definido na linha 751 do ficheiro ecra_impl.H.

Referências moveCursorPara() e posicaoDoCursor().

Referenciado por moveCursorDeAcordoCom().

10.7.3.22 void Slang::Ecra::avancaCursor () [inline]

Avança o cursor uma coluna (para a direita).

Precondição:
p = **posicaoDoCursor()**.

Poscondição:
posicaoDoCursor() = p + Dimensao(0, 1) (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Definido na linha 762 do ficheiro ecra_impl.H.

Referências moveCursorPara() e posicaoDoCursor().

Referenciado por moveCursorDeAcordoCom().

10.7.3.23 void Slang::Ecra::moveCursorDeAcordoCom (Tecla const & *tecla*)

Desloca o cursor na direcção dada por uma tecla de deslocamento.

Precondição:
tecla.eDeslocamento() e p = **posicaoDoCursor()**.

Poscondição:
(posicaoDoCursor() = p - Dimensao(1, 0) e tecla = Tecla::cima) ou (posicaoDoCursor() = p + Dimensao(1, 0) e tecla = Tecla::baixo) ou (posicaoDoCursor() = p - Dimensao(0, 1) e tecla = Tecla::esquerda) ou (posicaoDoCursor() = p + Dimensao(0, 1) e tecla = Tecla::direita) (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Definido na linha 150 do ficheiro ecra.C.

Referências avancaCursor(), baixaCursor(), Slang::Tecla::eDeDeslocamento(), recuaCursor() e sobeCursor().

10.7.3.24 Slang::Ecra::Troco const Slang::Ecra::trocoDe (Caixa const & *caixa*) const

Devolve um troço do ecrã correspondente à caixa dada.

Um exemplo de utilização pode ser encontrado em:

```
#include <string>

#include <Slang++/slang.H>

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

using namespace std;

using namespace Slang;

// Este exemplo deve ser testado em janelas com pelo menos 80 colunas!
int main()
{
    string letras = "abcdefghijklmnopqrstuvwxyz";

    // Preenche o ecrã com um padrão de letras:
    for(int i = 0; i != ecra.dimensao().numeroDeLinhas(); ++i) {
        ecra << cursor(i, 0);
        for(int j = 0; j != ecra.dimensao().numeroDeColunas(); ++j)
            ecra << letras[(i * ecra.dimensao().numeroDeColunas() +
                            j) % letras.size()];
    }

    ecra << refresca;

    Aviso("Ecrã preenchido.").interage();

    // Cria uma caixa de dimensão 10 × 20 com origem na posição (0, 0):
    Caixa caixa(Posicao(0, 0), Dimensao(10, 20));

    // Obtém cópia do troço de ecrã que se encontra sob a caixa:
    Ecra::Troco troco = ecra.trocoDe(caixa);
```

```
    Aviso("Troço copiado.  Tome atenção às alterações que "
          "ocorrerão no ecrã!").interage();

    // Move o cursor para a posição (11, 21), cola o troço copiado acima nessa
    // posição e refresca o ecrã de modo a que estas modificações se tornem
    // visíveis:
    ecra << cursor(11, 21) << troco << refresca;

    Aviso("Troço colado.  Não viu?  Vou colar em sequência...").interage();

    for(int j = 22; j != 80; ++j) {
        // Move o cursor para a posição (11, j), cola o troço copiado acima nessa
        // posição e refresca o ecrã de modo a que estas modificações se tornem
        // visíveis:
        ecra << cursor(11, j) << troco << refresca;

        // Espera 1 segundo:
        sleep(1);
    }

    Aviso("Agora viu...").interage();
}
```

Precondição:
caixa.eCanonica().

Poscondição:
trocoDe = troço representando a cópia da zona do ecrã correspondendo a *caixa.*

Definido na linha 181 do ficheiro ecra.C.

Referências Slang::Posicao::coluna(), Slang::Ecra::Troco::dados, Slang::Caixa::dimensao(), Slang::Caixa::eCanonica(), Slang::Posicao::linha(), moveCursorPara(), Slang::Dimensao::numeroDeColunas(), Slang::Dimensao::numeroDeLinhas(), Slang::Caixa::origem() e posicaoDoCursor().

Referenciado por trocoDoEcraCompleto() e trocoNoCursorCom().

10.7.3.25 Slang::Ecra::Troco const Slang::Ecra::trocoDoEcraCompleto () const [inline]

Devolve uma cópia integral do ecrã real.

Precondição:
V.

Poscondição:
trocoDoEcraCompleto = troço representando a cópia do ecrã real completo.

Definido na linha 773 do ficheiro ecra_impl.H.

Referências dimensao() e trocoDe().

Referenciado por Slang::CaixaDeTexto::interage() e Slang::MenuSimples::interage().

10.7.3.26 Slang::Ecra::Troco const Slang::Ecra::trocoNoCursorCom (Dimensao const & *dimensao*) const [inline]

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

Precondição:
dimensao.eCanonica().

Poscondição:
trocoNoCursorCom = troço representando a cópia da zona do ecrã com origem em **posicaoDoCursor()** e dimensão *dimensao.*

Definido na linha 783 do ficheiro ecra_impl.H.

Referências Slang::Dimensao::eCanonica(), posicaoDoCursor() e trocoDe().

10.7.3.27 void Slang::Ecra::cola (Troco const & *troco*, Posicao const & *em* = Posicao())

Cola no ecrã, na posição indicada (por omissão o canto superior esquerdo do ecrã), 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.

Precondição:
V.

Poscondição:
ecrã contém cópia do troço *troco* a partir da posição *em,* ficando o cursor na posição imediatamente após a última coluna da última linha onde o troço foi colocado (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor), excepto se o cursor tiver de ficar parado.

Veja também:
parado

Definido na linha 211 do ficheiro ecra.C.

Referências Slang::Posicao::coluna(), Slang::Ecra::Troco::dados, Slang::Ecra::Troco::dimensao(), Slang::fundo(), Slang::Posicao::linha(), moveCursorPara(), Slang::Dimensao::numeroDeColunas(), Slang::Dimensao::numeroDeLinhas() e posicaoDoCursor().

Referenciado por Slang::CaixaDeTexto::interage(), Slang::MenuSimples::interage() e operator<<().

10.7.3.28 void Slang::Ecra::tocaCampainha () const [inline]

Toca a "campaínha" do ecrã.

Por vezes o ecrã pisca, embora não seja nunca alterado. Por exemplo:

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

using namespace Slang;

int main()
{
    ecra << refresca;
```

```
    while(true) {
        teclado.leProximaTeclaDisponivel();
        Tecla tecla_premida = teclado.teclaLida();

        if(tecla_premida.eChar() and tecla_premida.comoChar() == 's')
            break;

        ecra << campainha;
    }
}
```

Precondição:
V.

Poscondição:
Uma campaínha soou.

Definido na linha 793 do ficheiro ecra_impl.H.

10.7.3.29 void Slang::Ecra::apagaDoCursorAteOFimDaLinha () [inline]

Apaga (i.e., pinta com cor do fundo) desde o cursor até ao fim da respectiva linha.

Precondição:
V.

Poscondição:
ecrã "vazio" entre **posicaoDoCursor()** e o final da respectiva linha.

Definido na linha 802 do ficheiro ecra_impl.H.

10.7.3.30 void Slang::Ecra::apagaDoCursorAteOFim () [inline]

Apaga (i.e., pinta com cor do fundo) desde o cursor até ao fim do ecrã.

Precondição:
V.

Poscondição:
ecrã vazio entre **posicaoDoCursor()** e o final do ecrã.

Definido na linha 813 do ficheiro ecra_impl.H.

10.7.3.31 void Slang::Ecra::apaga () [inline]

Apaga (i.e., pinta com cor do fundo) todo o ecrã.

Precondição:
V.

Poscondição:
ecrã vazio.

Definido na linha 824 do ficheiro ecra_impl.H.

10.7.3.32 void Slang::Ecra::refresca () const [inline]

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

Precondição:
V.

Poscondição:
Ecrã real reflecte exactamente a parte correspondente do ecrã virtual.

Definido na linha 835 do ficheiro ecra_impl.H.

Referenciado por refrescaTudo().

10.7.3.33 void Slang::Ecra::refrescaTudo () const [inline]

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

O efeito deve ser semelhante ao obtido pela operação **refresca()**, só diferindo em caso de erro do sistema. Útil para tentar ultrapassar erros de programação, fornecendo-se ao utilizador uma forma de refrescar o ecrã inteiro.

Precondição:
V.

Poscondição:
Ecrã real reflecte exactamente a parte correspondente do ecrã virtual.

Definido na linha 844 do ficheiro ecra_impl.H.

Referências dimensao() e refresca().

10.7.3.34 void Slang::Ecra::desenha (Caixa const & *caixa*)

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

O interior da caixa não é alterado. O cursor é colocado na célula imediatamente à direita do canto inferior direito da caixa (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor), excepto se se tiver optado por fixar o cursor.

O mesmo efeito pode ser obtido inserindo a caixa no ecrã. Veja-se o exemplo na documentação da classe **Caixa**.

Precondição:
caixa.eCanonica().

Poscondição:
ecrã contém caracteres desenhando uma caixa nas posições correspondentes à borda de *caixa* e o cursor está colocado na célula imediatamente à direita do canto inferior direito da caixa (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor), excepto se se tiver optado por fixar o cursor.

Definido na linha 250 do ficheiro ecra.C.

Referências Slang::Posicao::coluna(), Slang::Caixa::dimensao(), Slang::Posicao::linha(), moveCursorPara(), Slang::Dimensao::numeroDeColunas(), Slang::Dimensao::numeroDeLinhas(), Slang::Caixa::origem() e posicaoDoCursor().

Referenciado por operator<<().

10.7.3.35 void Slang::Ecra::desenhaSegmentoHorizontalCom (int const *largura*)

Desenha um segmento de recta horizontal, com a largura dada, a partir da posição do cursor (para a direita).

Veja-se um exemplo de utilização na documentação do manipulador **segmento_horizontal()**.

Precondição:
0 <= largura

Poscondição:
Ecrã contém segmento de recta horizontal com a largura dada e o cursor está colocado na célula imediatamente à direita do extremo direito do segmento (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor), excepto se se tiver optado por fixar o cursor.

Veja também:
segmento_horizontal()

Definido na linha 275 do ficheiro ecra.C.

Referências Slang::largura(), moveCursorPara() e posicaoDoCursor().

Referenciado por Slang::operator<<().

10.7.3.36 void Slang::Ecra::desenhaSegmentoVerticalCom (int const *altura*)

Desenha um segmento de recta vertical, com a altura dada, a partir da posição do cursor (para baixo).

Veja-se um exemplo de utilização na documentação do manipulador **segmento_horizontal()**.

Precondição:
0 <= altura

Poscondição:
Ecrã contém segmento de recta vertical com a altura dada e o cursor está colocado na célula imediatamente abaixo do extremo inferior do segmento (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor), excepto se se tiver optado por fixar o cursor.

Definido na linha 298 do ficheiro ecra.C.

Referências moveCursorPara() e posicaoDoCursor().

Referenciado por Slang::operator<<().

10.7.3.37 Slang::Ecra & Slang::Ecra::operator<< (std::string const & *cadeia*)

Insere uma cadeia de caracteres na posição do cursor.

O cursor é deslocado para a posição à direita do último caractere escrito (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor)., 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. Os espaços são inseridos à esquerda, à direita ou de ambos os lados, consoante o tipo de justificação em vigor no ecrã.

Precondição:
V.

Poscondição:
(ver descrição acima).

Veja também:
parado
largura
Justificacao

Definido na linha 321 do ficheiro ecra.C.

Referências Slang::a_direita, Slang::a_esquerda, Slang::ao_centro, moveCursorPara() e posicaoDoCursor().

10.7.3.38 template<typename T> Slang::Ecra & Slang::Ecra::operator<< (T const & *valor*) [inline]

Insere um valor do tipo T genérico na posição do cursor.

O efeito é idêntico ao do operador<< para cadeias de caracteres. O tipo T do valor inserido tem de ter o operador << disponível para inserções em canais *ostream* (ou pelo menos *ostringstream).*

Precondição:
V.

Poscondição:
(ver descrição acima).

Definido na linha 857 do ficheiro ecra_impl.H.

Referências Slang::ecra.

10.7.3.39 Slang::Ecra & Slang::Ecra::operator<< (Caixa const & *caixa*) [inline]

Desenha uma caixa no ecrã.

Efeito equivalente ao da operação desenha(Caixa).

Veja-se o exemplo na documentação da classe **Caixa**.

Precondição:
caixa.eCanonica().

Poscondição:
Ver descrição de **desenha()**.

Definido na linha 874 do ficheiro ecra_impl.H.

Referências Slang::caixa(), desenha() e Slang::Caixa::eCanonica().

10.7.3.40 Slang::Ecra & Slang::Ecra::operator<< (Simbolo const *simbolo*) [inline]

Desenha um símbolo especial no ecrã.

Veja-se um exemplo de utilização na classe Simbolo.

Precondição:
V.

Poscondição:
ecrã contém o símbolo simbolo a partir da posição do cursor (ver descrição de operador << para cadeias de caracteres), avançando o cursor para a posição à sua direita (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor).

Veja também:
Simbolo

Definido na linha 888 do ficheiro ecra_impl.H.

Referências Slang::Simbolo.

10.7.3.41 Slang::Ecra & Slang::Ecra::operator<< (Troco const & *troco*) [inline]

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 (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor), 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.

Veja-se um exemplo de utilização na documentação da operação **trocoDe()**.

Precondição:
V.

Poscondição:
ecrã contém cópia do troço *troco* a partir da posição *em,* ficando o cursor na posição imediatamente após a última coluna da última linha onde o troço foi colocado (mas limitado ao ecrã real se se optou por limitar o posicionamento do cursor), excepto se o cursor tiver de ficar parado.

Veja também:
parado
cola

Definido na linha 903 do ficheiro ecra_impl.H.

Referências cola() e posicaoDoCursor().

10.7.3.42 Slang::Ecra const & Slang::Ecra::operator>> (Troco & *troco*) const

Extrai um troço do ecrã a partir da posição actual do cursor.

Precondição:
V.

Poscondição:
troco é cópia da zona do ecrã começando na posição actual do cursor e com dimensão troco.dimensao().

Veja também:
trocoDe()

Definido na linha 390 do ficheiro ecra.C.

Referências Slang::Posicao::coluna(), Slang::Ecra::Troco::dados, Slang::Ecra::Troco::dimensao(), Slang::Posicao::linha(), moveCursorPara(), Slang::Dimensao::numeroDeColunas(), Slang::Dimensao::numeroDeLinhas() e posicaoDoCursor().

10.7.3.43 Slang::Ecra & Slang::Ecra::operator<< (Posicao const & *nova_posicao_do_cursor*) [inline]

Altera a posição do cursor para a posição dada (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Precondição:
V.

Poscondição:
posicaoDoCursor() = nova_posicao_do_cursor (mas limitada ao ecrã real se se optou por limitar o posicionamento do cursor).

Veja também:
cursor
moveCursorPara()

Definido na linha 917 do ficheiro ecra_impl.H.

Referências moveCursorPara().

10.7.3.44 Slang::Ecra & Slang::Ecra::operator<< (ObjectoCor const & *objecto_cor*) [inline]

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

Um exemplo de utilização pode ser encontrado na documentação da operacao **mudaObjectoCorEmUsoPara()**.

Precondição:
V.

Poscondição:
Próximas inserções no ecrã (excepto colagens) serão feitas usando o objecto cor passado.

Veja também:
mudaObjectoCorEmUsoPara()

Definido na linha 930 do ficheiro ecra_impl.H.

Referências mudaObjectoCorEmUsoPara().

10.7.3.45 Slang::Ecra & Slang::Ecra::operator<< (Justificacao const *nova_justificacao*) [inline]

Muda a justificação do ecrã para a justificação dada.

A justificação mantém-se até à próxima alteração.

Precondição:
V.

Poscondição:
justificacaoActual() = justificacao das inserções no ecrã em vigor.

Definido na linha 944 do ficheiro ecra_impl.H.

Referências Slang::Justificacao e mudaJustificacaoPara().

10.7.3.46 Slang::Ecra & Slang::Ecra::operator<< (void *manipulador*(Ecra &ecra)) [inline]

"Gancho" para definir novos manipuladores do ecrã.

Um novo manipulador consegue-se facilmente definindo um procedimento que receba um ecrã por referência. Esse procedimento será invocado automaticamente quando for inserido no ecrã através do operador <<.

O exemplo abaixo demonstra como se pode definir um manipulador para deslocar o cursor para uma posição aleatória do ecrã real:

```
#include <cstdlib>

#include <Slang++/slang.H>

using namespace std;

using namespace Slang;

void cursor_aleatorio(Ecra& ecra)
{
    ecra << cursor(rand() % ecra.dimensao().numeroDeLinhas(),
                   rand() % ecra.dimensao().numeroDeColunas());
}

int main()
{
    ecra << refresca;

    while(true) {
        teclado.leProximaTeclaDisponivel();
        Tecla tecla_premida = teclado.teclaLida();

        if(tecla_premida.eChar() and tecla_premida.comoChar() == 's')
            break;

        ecra << cursor_aleatorio << refresca;
    }
}
```

Precondição:
V.

Poscondição:
O efeito é o mesmo de invocar manipulador(*this).

Definido na linha 958 do ficheiro ecra_impl.H.

Referências Slang::ecra.

10.7.4 Documentação das classes amigas e funções relacionadas

10.7.4.1 friend class ObjectoCor [friend]

Definido na linha 1821 do ficheiro ecra.H.

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

- **ecra.H**
- **ecra.C**
- **ecra_impl.H**

10.8 Referência à classe Slang::Ecra::ObjectoCor

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

`#include <Slang++/ecra.H>`

Membros públicos

- **ObjectoCor** (**Cor** const cor_do_texto, **Cor** const cor_do_fundo)
  Constrói um novo objecto cor com as cores dadas.
- **~ObjectoCor** ()
  Destrói um objecto cor, libertando a respectiva posição na palete (ou pelo menos reduzindo o seu grau de partilha).

Inspectores

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

Modificadores

- void **mudaCorDoTextoPara** (**Cor** const nova_cor_do_texto)
  Modifica a cor do texto corrente deste objecto cor.
- void **mudaCorDoFundoPara** (**Cor** const nova_cor_do_fundo)
  Modifica a cor do fundo corrente deste objecto cor.

Amigos

- class **Ecra**

10.8.1 Descrição detalhada

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 S-Lang (base desta biblioteca `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 S-Lang), o número de objectos cor definidos em cada momento não deverá ultrapassar os 255! Se isso acontecer, as cores deixarão 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. Infelizmente a biblioteca S-Lang não permite detectar este problema.

Os objectos cor não são copiáveis! Este facto deve-se a que cada objecto criado ocupa um posição da palete. Assim, não é possível passar um objecto cor como argumento para uma função ou procedimento, excepto por referência.

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&)** do seguinte modo:

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

A partir desta instrução tudo que for inserido ficará vermelho sobre verde até que a cor seja mudada de novo. Note-se que é recomendável que a variável *cor* persista enquanto existirem no ecrã células desenhadas com recurso a essa cor, pois de outra forma a construção de um novo objecto cor poderá ocupar a mesma posição na palete, alterando a côr das respectivas células do ecrã.

Após uma colagem de um troço de ecrã copiado previamente a cor activa volta a ser a do fundo do ecrã. Espera-se que um dia esta restrição venha a ser eliminada.

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.

Os efeitos estranhos associados ao modelo de cor usado podem ser confirmados através exemplos que se seguem.

O primeiro exemplo demonstra o efeito dos tempos de vida dos objectos cor:

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

using namespace Slang;

Ecra::ObjectoCor const cor1(verde, branco);

void procedimento1()
{
    Ecra::ObjectoCor cor2(amarelo, azul);

    ecra << cor2 << parado << "Amarelo sobre azul..." << baixaCursor;
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
}

void procedimento2()
{
    Ecra::ObjectoCor cor3(azul, amarelo);

    ecra << cor3 << parado << "Azul sobre amarelo..." << baixaCursor;
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
}

int main()
{
```

```
    ecra << cor1 << parado << "Verde sobre branco..." << baixaCursor;
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();

    procedimento1();

    procedimento2();

    Ecra::ObjectoCor cor4(branco, verde);

    ecra << cor4 << parado << "Branco sobre verde..." << baixaCursor;
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
}
```

O segundo exemplo demonstra o efeito de alterar as cores de um objecto cor sobre as células do ecrã que com ele tenham sido previamente desenhadas:

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

using namespace Slang;

int main()
{
    Ecra::ObjectoCor cor(branco, verde);

    ecra << cor << parado << "Branco sobre verde..." << baixaCursor;
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();

    // A cor do texto já inserido no ecrã mudará com a alteração do
    // correspondente objecto cor:
    cor.mudaCorDoTextoPara(vermelho);
    ecra << refresca;
    teclado.leProximaTeclaDisponivel();
}
```

Invariante:

1 <= numero_do_objecto < numero_de_objectos_cor_no_s_lang.

Tarefa

Faltam as operações carregaDe() e guardaEm().
Dever-se-ia gerir melhor as posições da palete.

Definido na linha 1899 do ficheiro ecra.H.

10.8.2 Documentação dos Construtores & Destrutor

10.8.2.1 Slang::Ecra::ObjectoCor::ObjectoCor (Cor const *cor_do_texto*, Cor const *cor_do_fundo*) [inline]

Constrói um novo objecto cor com as cores dadas.

Ocupa uma posição da palete disponível. Se não houver nenhuma, passa a partilhar uma posição já ocupada, o que resulta em efeitos estranhos. Na prática nunca acontece, pois programas em modo texto não têm nunca mais de 256 cores...

Precondição:

V.

Poscondição:
corDoTexto() = cor_do_texto e **corDoFundo()** = cor_do_fundo.

Definido na linha 995 do ficheiro ecra_impl.H.

Referências Slang::Cor, Slang::ecra e Slang::Ecra::mudaCoresDe().

10.8.2.2 Slang::Ecra::ObjectoCor::~ObjectoCor () [inline]

Destrói um objecto cor, libertando a respectiva posição na palete (ou pelo menos reduzindo o seu grau de partilha).

Precondição:
V.

Definido na linha 1005 do ficheiro ecra_impl.H.

Referências Slang::ecra e Slang::Ecra::libertaObjecto().

10.8.3 Documentação dos métodos

10.8.3.1 Slang::Cor Slang::Ecra::ObjectoCor::corDoTexto () const [inline]

Devolve a cor do texto corrente deste objecto cor.

Precondição:
V.

Poscondição:
corDoTexto = cor do texto corrente do objecto cor.

Definido na linha 1012 do ficheiro ecra_impl.H.

10.8.3.2 Slang::Cor Slang::Ecra::ObjectoCor::corDoFundo () const [inline]

Devolve a cor do fundo corrente deste objecto cor.

Precondição:
V.

Poscondição:
corDoTexto = cor do fundo corrente do objecto cor.

Definido na linha 1019 do ficheiro ecra_impl.H.

10.8.3.3 void Slang::Ecra::ObjectoCor::mudaCorDoTextoPara (Cor const *nova_cor_do_texto*) [inline]

Modifica a cor do texto corrente deste objecto cor.

Afecta todas as células do ecrã desenhadas com este objecto cor.

Precondição:
V.

Poscondição:
corDoTexto() = nova_cor_do_texto.

Definido na linha 1027 do ficheiro ecra_impl.H.

Referências Slang::Cor, Slang::ecra e Slang::Ecra::mudaCoresDe().

10.8.3.4 void Slang::Ecra::ObjectoCor::mudaCorDoFundoPara (Cor const *nova_cor_do_fundo*) [inline]

Modifica a cor do fundo corrente deste objecto cor.

Afecta todas as células do ecrã desenhadas com este objecto cor.

Precondição:
V.

Poscondição:
corDoFundo() = nova_cor_do_fundo.

Definido na linha 1039 do ficheiro ecra_impl.H.

Referências Slang::Cor, Slang::ecra e Slang::Ecra::mudaCoresDe().

10.8.4 Documentação das classes amigas e funções relacionadas

10.8.4.1 friend class Ecra [friend]

Definido na linha 1979 do ficheiro ecra.H.

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

- **ecra.H**
- **ecra_impl.H**

10.9 Referência à classe Slang::Ecra::Troco

Esta classe serve para representar um troço de ecrã.

`#include <Slang++/ecra.H>`

Membros públicos

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

  Constrói um novo troço.

- **Dimensao dimensao** () const

  Devolve a dimensão do troço.

Amigos

- class **Ecra**

10.9.1 Descrição detalhada

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 (i.e., caracteres e cores (do texto e do fundo) de todas as células de uma zona, ou trço, rectangular do ecrã. 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.

Exemplo

O seguinte programa demonstra a utilização de troços de ecrã:

```
#include <string>

#include <Slang++/slang.H>

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

using namespace std;

using namespace Slang;

// Este exemplo deve ser testado em janelas com pelo menos 80 colunas!
int main()
{
    string letras = "abcdefghijklmnopqrstuvwxyz";

    // Preenche o ecrã com um padrão de letras:
    for(int i = 0; i != ecra.dimensao().numeroDeLinhas(); ++i) {
        ecra << cursor(i, 0);
        for(int j = 0; j != ecra.dimensao().numeroDeColunas(); ++j)
```

```
            ecra << letras[(i * ecra.dimensao().numeroDeColunas() +
                            j) % letras.size()];
    }

    ecra << refresca;

    Aviso("Ecrã preenchido.").interage();

    // Cria uma caixa de dimensão 10 × 20 com origem na posição (0, 0):
    Caixa caixa(Posicao(0, 0), Dimensao(10, 20));

    // Obtém cópia do troço de ecrã que se encontra sob a caixa:
    Ecra::Troco troco = ecra.trocoDe(caixa);

    Aviso("Troço copiado.  Tome atenção às alterações que "
          "ocorrerão no ecrã!").interage();

    // Move o cursor para a posição (11, 21), cola o troço copiado acima nessa
    // posição e refresca o ecrã de modo a que estas modificações se tornem
    // visíveis:
    ecra << cursor(11, 21) << troco << refresca;

    Aviso("Troço colado.  Não viu?  Vou colar em sequência...").interage();

    for(int j = 22; j != 80; ++j) {
        // Move o cursor para a posição (11, j), cola o troço copiado acima nessa
        // posição e refresca o ecrã de modo a que estas modificações se tornem
        // visíveis:
        ecra << cursor(11, j) << troco << refresca;

        // Espera 1 segundo:
        sleep(1);
    }

    Aviso("Agora viu...").interage();
}
```

Invariante:
dimensao_.numeroDeLinhas() * dimensao_.numeroDeColunas() = dados.size().

Definido na linha 2006 do ficheiro ecra.H.

10.9.2 Documentação dos Construtores & Destrutor

10.9.2.1 Slang::Ecra::Troco::Troco (Dimensao const & *dimensao*) [inline, explicit]

Constrói um novo troço.

É pouco usual ser usado directamente. Normalmente é a própria classe ecrã que constrói o troço e o devolve, por exemplo através da operação **Ecra::trocoDe()** (e portanto o construtor mais usado é o construtor por cópia).

Precondição:
dimensao.eCanonica().

Poscondição:
*this é um troço com a dimensão indicada, embora vazio.

Definido na linha 1059 do ficheiro ecra_impl.H.

Referências dimensao(), Slang::Ecra::dimensao() e Slang::Dimensao::eCanonica().

10.9.3 Documentação dos métodos

10.9.3.1 Slang::Dimensao Slang::Ecra::Troco::dimensao () const [inline]

Devolve a dimensão do troço.

Precondição:
V.

Poscondição:
dimensao = dimensão do troço.

Definido na linha 1068 do ficheiro ecra_impl.H.

Referenciado por Slang::Ecra::cola(), Slang::Ecra::operator>>() e Troco().

10.9.4 Documentação das classes amigas e funções relacionadas

10.9.4.1 friend class Ecra [friend]

Definido na linha 2039 do ficheiro ecra.H.

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

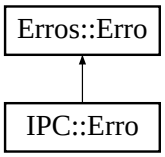
- **ecra.H**
- **ecra_impl.H**

10.10 Referência à classe IPC::Erro

Representa as excepções da biblioteca IPC++.

`#include <IPC++/mensageiro.H>`

Diagrama de heranças da classe IPC::Erro:



Membros públicos

- **Erro** (std::string const &mensagem)

  Constrói um erro guardando a mensagem de erro recebida como parâmetro.

- **Erro** ()

  Constrói excepção usando mensagem de erro padrão correspondente ao último erro **IPC** *ocorrido.*

10.10.1 Descrição detalhada

Representa as excepções da biblioteca IPC++.

Definido na linha 34 do ficheiro mensageiro.H.

10.10.2 Documentação dos Construtores & Destrutor

10.10.2.1 IPC::Erro::Erro (std::string const & *mensagem*) `[inline]`

Constrói um erro guardando a mensagem de erro recebida como parâmetro.

Reimplementado de **Erros::Erro** (p. 154).

Definido na linha 14 do ficheiro mensageiro_impl.H.

10.10.2.2 IPC::Erro::Erro () `[inline]`

Constrói excepção usando mensagem de erro padrão correspondente ao último erro **IPC** ocorrido.

Definido na linha 19 do ficheiro mensageiro_impl.H.

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

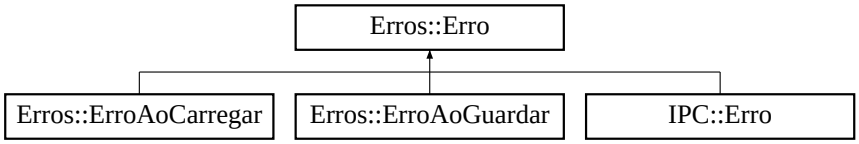
- **mensageiro.H**
- **mensageiro_impl.H**

10.11 Referência à classe Erros::Erro

Base de uma pequena hierarquia de classes representando excepções.

`#include <Erros/erros.H>`

Diagrama de heranças da classe Erros::Erro:



Membros públicos

- **Erro** (std::string const &mensagem)

  Constrói um erro guardando a mensagem de erro recebida como parâmetro.

- virtual ~**Erro** ()

  Destrutor virtual para poder sofrer derivações...

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

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

10.11.1 Descrição detalhada

Base de uma pequena hierarquia de classes representando excepções.

Definido na linha 30 do ficheiro erros.H.

10.11.2 Documentação dos Construtores & Destrutor

10.11.2.1 Erros::Erro::Erro (std::string const & *mensagem*) [inline]

Constrói um erro guardando a mensagem de erro recebida como parâmetro.

Reimplementado em **IPC::Erro** (p. 153).

Definido na linha 9 do ficheiro erros_impl.H.

10.11.2.2 Erros::Erro::~Erro () [inline, virtual]

Destrutor virtual para poder sofrer derivações...

Definido na linha 14 do ficheiro erros_impl.H.

10.11.3 Documentação dos métodos

10.11.3.1 Erros::Erro::operator std::string () const [inline, virtual]

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

Definido na linha 18 do ficheiro erros_impl.H.

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

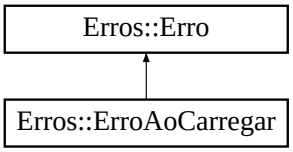
- **erros.H**
- **erros_impl.H**

10.12 Referência à classe Erros::ErroAoCarregar

Representa excepções ocorridas durante o carregamento de objectos a partir de canais.

`#include <Erros/erros.H>`

Diagrama de heranças da classe Erros::ErroAoCarregar:



Membros públicos

- **ErroAoCarregar** (std::string const &classe)
  Constrói erro ao carregar recebendo como parâmetro o nome da classe que originou a excepção.

10.12.1 Descrição detalhada

Representa excepções ocorridas durante o carregamento de objectos a partir de canais.

Definido na linha 54 do ficheiro erros.H.

10.12.2 Documentação dos Construtores & Destrutor

10.12.2.1 Erros::ErroAoCarregar::ErroAoCarregar (std::string const & *classe*) [inline]

Constrói erro ao carregar recebendo como parâmetro o nome da classe que originou a excepção.

Definido na linha 26 do ficheiro erros_impl.H.

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

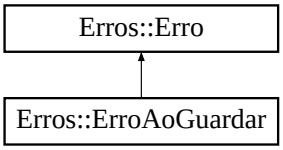
- **erros.H**
- **erros_impl.H**

10.13 Referência à classe Erros::ErroAoGuardar

Representa excepções ocorridas durante o armazenamento de objectos usando canais.

`#include <Erros/erros.H>`

Diagrama de heranças da classe Erros::ErroAoGuardar:



Membros públicos

- **ErroAoGuardar** (std::string const &classe)
  Constrói erro ao guardar recebendo como parâmetro o nome da classe que originou a excepção.

10.13.1 Descrição detalhada

Representa excepções ocorridas durante o armazenamento de objectos usando canais.

Definido na linha 67 do ficheiro erros.H.

10.13.2 Documentação dos Construtores & Destrutor

10.13.2.1 Erros::ErroAoGuardar::ErroAoGuardar (std::string const & *classe*) [inline]

Constrói erro ao guardar recebendo como parâmetro o nome da classe que originou a excepção.

Definido na linha 34 do ficheiro erros_impl.H.

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

- **erros.H**
- **erros_impl.H**

10.14 Referência à classe Utilitarios::Ignorador

Representa manipuladores de canais de entrada (istream) que descartam caracteres.

`#include <Utilitarios/ignoradores.H>`

Membros públicos

- **Ignorador** (char const terminador, bool const deve_limpar_o_canal=false)

  Constroí um ignorador de caracteres.

- char **terminador** () const

  Devolve o caractere que termina a sequência de caracteres descartados.

- bool **deveLimparOCanal** () const

  Indica se o ignorador deve limpar possíveis condições de erro do canal antes de descartar caracteres.

10.14.1 Descrição detalhada

Representa manipuladores de canais de entrada (istream) que descartam caracteres.

Veja também:
Utilitatios::il
Utilitarios::ill

Definido na linha 28 do ficheiro ignoradores.H.

10.14.2 Documentação dos Construtores & Destrutor

10.14.2.1 Utilitarios::Ignorador::Ignorador (char const *terminador*, bool const *deve_limpar_o_canal* = false) [inline]

Constroí um ignorador de caracteres.

Este ignorador ignora todos os caracteres encontrados até chegar a um **terminador**. Se #deve_limpar_o_canal for verdadeira, antes de tentar ignorar caracteres limpa as possíveis condições de erro do canal.

Precondição:
V.

Definido na linha 9 do ficheiro ignoradores_impl.H.

10.14.3 Documentação dos métodos

10.14.3.1 char Utilitarios::Ignorador::terminador () const [inline]

Devolve o caractere que termina a sequência de caracteres descartados.

I.e., serão descartados todos os caracteres encontrados até se encontrar o chamado "caractere terminador.

Precondição:
V.

Poscondição:
terminador = caractere terminador da sequência de caracteres descartados.

Definido na linha 16 do ficheiro ignoradores_impl.H.

10.14.3.2 bool Utilitarios::Ignorador::deveLimparOCanal () const [inline]

Indica se o ignorador deve limpar possíveis condições de erro do canal antes de descartar caracteres.

Precondição:
V.

Poscondição:
deveLimparOCanal = o ignorador deve limpar possíveis condições de erro do canal antes de descartar caracteres.

Definido na linha 21 do ficheiro ignoradores_impl.H.

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

- **ignoradores.H**
- **ignoradores_impl.H**

10.15 Referência à estrutura Slang::ManipuladorDaLargura

Não usar directamente.

`#include <ecra.H>`

Membros públicos

- **ManipuladorDaLargura** (int const **largura**=0)

Atributos Públicos

- int **largura**

10.15.1 Descrição detalhada

Não usar directamente.

Veja também:
Slang::largura()

Definido na linha 2057 do ficheiro ecra.H.

10.15.2 Documentação dos Construtores & Destrutor

10.15.2.1 Slang::ManipuladorDaLargura::ManipuladorDaLargura (int const *largura* = 0) [inline, explicit]

Definido na linha 1084 do ficheiro ecra_impl.H.

Referências Slang::largura().

10.15.3 Documentação dos dados membro

10.15.3.1 int Slang::ManipuladorDaLargura::largura

Definido na linha 2060 do ficheiro ecra.H.

Referenciado por Slang::operator<<().

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

- **ecra.H**
- **ecra_impl.H**

10.16 Referência à estrutura Slang::ManipuladorDeDesenhoDeSegmento

Não usar directamente.

`#include <ecra.H>`

Membros públicos

- **ManipuladorDeDesenhoDeSegmento** (bool const **e_horizontal**, int const **dimensao**=1)

Atributos Públicos

- bool **e_horizontal**
- int **dimensao**

10.16.1 Descrição detalhada

Não usar directamente.

Veja também:
Slang::segmento_horizontal()
Slang::segmento_vertical()

Definido na linha 2073 do ficheiro ecra.H.

10.16.2 Documentação dos Construtores & Destrutor

10.16.2.1 Slang::ManipuladorDeDesenhoDeSegmento::ManipuladorDeDesenhoDeSegmento (bool const *e_horizontal*, int const *dimensao* = 1) [inline, explicit]

Definido na linha 1101 do ficheiro ecra_impl.H.

10.16.3 Documentação dos dados membro

10.16.3.1 bool Slang::ManipuladorDeDesenhoDeSegmento::e_horizontal

Definido na linha 2077 do ficheiro ecra.H.

Referenciado por Slang::operator<<().

10.16.3.2 int Slang::ManipuladorDeDesenhoDeSegmento::dimensao

Definido na linha 2078 do ficheiro ecra.H.

Referenciado por Slang::operator<<().

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

- **ecra.H**
- **ecra_impl.H**

10.17 Referência à classe IPC::Mensageiro

Classe cuja única instância (em cada processo) representa um mensageiro entre o processo em causa e um outro na mesma máquina.

`#include <IPC++/mensageiro.H>`

Membros públicos

- **Mensageiro** (key_t const chave=getuid())

  Contrói um novo mensageiro.

- **~Mensageiro** ()

  Destrói o mensageiro, destruindo a caixa do correio criada se for o último mensageiro em existência.

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

  Devolve um identificador único para o processo, que consiste no nome do utilizador seguido do número do processo.

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

  Devolve o identificador único do outro processo, com o qual o corrente está emparelhado através do mensageiro.

- void **leMensagem** ()

  Lê uma nova mensagem vinda do "outro".

- template<typename Tipo> Tipo const **mensagemLida** () const

  Devolve a mensagem lida na forma de um valor do tipo Tipo.

- template<typename Tipo> void **envia** (Tipo &mensagem) const

  Envia uma mensagem correspondente a um valor do tipo Tipo (versão para variáveis).

- template<typename Tipo> void **envia** (Tipo const &mensagem) const

  Envia uma mensagem correspondente a um valor do tipo Tipo (versão para constantes e valores temporários).

- template<typename Char> void **envia** (Char *&mensagem) const

  Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros simples).

- template<typename Char> void **envia** (Char const *&mensagem) const

  Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros para caracteres constantes).

- template<typename Char> void **envia** (Char *const &mensagem) const

  Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros constantes).

- template<typename Char> void **envia** (Char const *const &mensagem) const

Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros constantes para caracteres constantes).

- template<typename Char, int dimensao> void **envia** (Char const(&mensagem)[dimensao]) const

  Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para matrizes de caracteres).

10.17.1 Descrição detalhada

Classe cuja única instância (em cada processo) representa um mensageiro entre o processo em causa e um outro na mesma máquina.

Esta classe é um solitão! A ideia é que um processo com uma instância desta classe fica em comunicação com outro com outra instância da mesma classe. A selecção do outro processo é feita automaticamente, de acordo com as disponibilidades. A ideia é permitir a comunicação entre dois processos do mesmo jogo, sendo que os jogadores não têm controlo sobre que será o seu opositor.

Nas descrições usa-se o termo "eu" para indicar o processo corrente, e "outro" para indicar o processo a que o "eu" está emparelhado pelo mensageiro.

A comunicação faz-se enviando e recebendo mensagens. As mensagens enviadas e recebidas são de qualquer tipo para o qual:

1. estejam definido os operadores << e >> para ostream e istream, respectivamente;
2. tenham definido um construtor por omissão; e
3. tenham definido um cnstrutor por cópia.

Não existe, para já, nenhuma forma de indagar se há alguma mensagem disponível para ser recebida e muito menos saber o seu tipo. O protocolo entre os processos é que tem de garantir que os processos recebem tantas mensagens quantas as enviadas pelo outro, enviam tantas mensagens quantas as recebidas pelo outro, e que as mensagens enviadas e recebidas têm tipos compatíveis.

A classe é simples de usar. Por exemplo, suponha que se pretendia fazer um pequeno jogo em que ganha quem gerar um número aleatório maior. Poder-se-ia escrever o seguinte programa:

```
#include <iostream>
#include <cstdlib>
#include <ctime>

#include <IPC++/mensageiro.H>

using namespace std;
using namespace IPC;

int main()
{
    // Se não se fizer isto os jogadores geram sempre o mesmo número
    // aleatório...
    srand(time(0));

    try {
        // Construir mensageiro:
        Mensageiro mensageiro;

        // Escrever identificações:
```

```
        cout << "Eu sou o " << mensageiro.meuIdentificador() << endl;
        cout << "Ele é o " << mensageiro.identificadorDoOutro() << endl;

        // Gerar número aleatório:
        int meu = rand();

        // Enviá-lo ao adversário:
        mensageiro.envia(meu);

        // Receber número do adeversário:

        mensageiro.leMensagem();
        int dele = mensageiro.mensagemLida<int>();

        // Verificar resultado:
        if(meu < dele)
            cout << "Perdi " << meu << " contra " << dele << "..."
                 << endl;
        else if(meu > dele)
            cout << "Ganhei " << meu << " contra " << dele << '!'
                 << endl;
        else
            cout << "Empatámos " << meu << " a " << dele << '.'
                 << endl;
    } catch(IPC::Erro& erro) {
        cerr << string(erro) << endl;
    }
}
```

Invariante:

meu_PID = int(getpid()) e meu_UID = int(getuid()) e PID_do_outro = PID do "outro" e UID_do_outro = UID do "outro" e 0 <= identificador_da_caixa_do_correio e 0 <= identificador_do_semaforo.

Definido na linha 89 do ficheiro mensageiro.H.

10.17.2 Documentação dos Construtores & Destrutor

10.17.2.1 IPC::Mensageiro::Mensageiro (key_t const *chave* = getuid())

Contrói um novo mensageiro.

Podem-se criar mensageiros de dois tipos: restritos ou não restritos. O primeiro tipo consegue-se não passando qualquer argumento ao construtor e restringe o emparelhamento dos processos a processos do mesmo utilizador. O segundo tipo é mais geral, pois permite o emparelhamento de processos de utilizadores arbitrários da mesma máquina, mas exige a passagem de uma chave ipc que deve ser única para cada tipo de aplicação prevista (por uma questão de compatibilidade de protocolos de comunicação).

Precondição:

V.

Poscondição:

O mensageiro está construído e ligado a um mensageiro de outro processo.

Excepções:

IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 11 do ficheiro mensageiro.C.

10.17.2.2 IPC::Mensageiro::~Mensageiro ()

Destrói o mensageiro, destruindo a caixa do correio criada se for o último mensageiro em existência.

Precondição:
V.

Definido na linha 76 do ficheiro mensageiro.C.

10.17.3 Documentação dos métodos

10.17.3.1 std::string IPC::Mensageiro::meuIdentificador () const [inline]

Devolve um identificador único para o processo, que consiste no nome do utilizador seguido do número do processo.

Precondição:
V.

Poscondição:
meuIdentificador = "UN + (PID)", onde UN é o nome do utilizador e PID é o número do processo.

Definido na linha 26 do ficheiro mensageiro_impl.H.

10.17.3.2 std::string IPC::Mensageiro::identificadorDoOutro () const [inline]

Devolve o identificador único do outro processo, com o qual o corrente está emparelhado através do mensageiro.

Precondição:
O processo do "outro" tem de existir.

Poscondição:
identificadorDoOutro = "UN + (PID)", onde UN é o nome do utilizador do "outro" e PID é o número do processo do "outro".

Definido na linha 34 do ficheiro mensageiro_impl.H.

10.17.3.3 void IPC::Mensageiro::leMensagem () [inline]

Lê uma nova mensagem vinda do "outro".

Espera pela mensagem a ler se não estiver nenhuma mensagem disponível.

Precondição:
V.

Poscondição:
Há uma mensagem lida.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 42 do ficheiro mensageiro_impl.H.

10.17.3.4 std::string const IPC::Mensageiro::mensagemLida< std::string > () const [inline]

Devolve a mensagem lida na forma de um valor do tipo *Tipo*.

É um erro se o tipo indicado não corresponder ao tipo do valor enviado pelo "outro".

Precondição:
Há uma mensagem lida.

Poscondição:
mensagemLida = mensagem enviada pelo "outro".

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos ou no caso de o tipo especificado ser incompatível com a mensagem lida.

Definido na linha 49 do ficheiro mensageiro_impl.H.

10.17.3.5 template<typename Tipo> void IPC::Mensageiro::envia (Tipo & *mensagem*) const [inline]

Envia uma mensagem correspondente a um valor do tipo Tipo (versão para variáveis).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 77 do ficheiro mensageiro_impl.H.

Referenciado por envia().

10.17.3.6 template<typename Tipo> void IPC::Mensageiro::envia (Tipo const & *mensagem*) const [inline]

Envia uma mensagem correspondente a um valor do tipo Tipo (versão para constantes e valores temporários).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 90 do ficheiro mensageiro_impl.H.

10.17.3.7 template<typename Char> void IPC::Mensageiro::envia (Char *& *mensagem*) const [inline]

Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros simples).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 121 do ficheiro mensageiro_impl.H.

Referências envia().

10.17.3.8 template<typename Char> void IPC::Mensageiro::envia (Char const *& *mensagem*) const [inline]

Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros para caracteres constantes).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 103 do ficheiro mensageiro_impl.H.

Referências envia().

10.17.3.9 template<typename Char> void IPC::Mensageiro::envia (Char *const & *mensagem*) const

Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros constantes).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

10.17.3.10 template<typename Char> void IPC::Mensageiro::envia (Char const *const & *mensagem*) const [inline]

Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para ponteiros constantes para caracteres constantes).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 109 do ficheiro mensageiro_impl.H.

Referências envia().

10.17.3.11 template<typename Char, int dimensao> void IPC::Mensageiro::envia (Char const & *mensagem*[dimensao]) const [inline]

Envia uma mensagem correspondente a um valor do tipo cadeia clássica de caracteres (versão para matrizes de caracteres).

Precondição:
V.

Poscondição:
Está disponível para o "outro" uma nova mensagem.

Excepções:
IPC::Erro no caso de haver erro na API de comunicação entre processos.

Definido na linha 128 do ficheiro mensageiro_impl.H.

Referências envia().

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

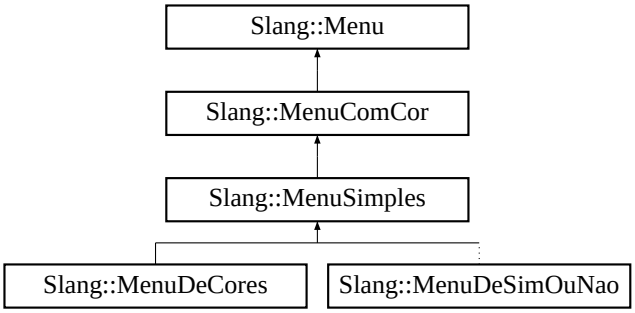
- **mensageiro.H**
- **mensageiro_impl.H**
- **mensageiro.C**

10.18 Referência à classe Slang::Menu

Classe abstracta que representa a interface básica de todos os menus.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::Menu:



Membros públicos

Construtores

- **Menu** (std::string const &titulo)
  Constrói um novo menu com o título dado.
- virtual ~**Menu** ()=0
  Destrutor abstracto marca uma classe abstracta.

Inspectores

- std::string **titulo** () const
  Devolve o título do menu.
- virtual int **opcaoActual** () const=0
  Devolve a opção actual do menu (o primeiro item do menu tem número 0).

Interacção com o utilizador

- virtual void **interage** ()=0
  Executa o menu, i.e., interage com o utilizador do programa.

10.18.1 Descrição detalhada

Classe abstracta que representa a interface básica de todos os menus.

Invariante:
titulo <> "" e eImprimivel(titulo)..

Definido na linha 167 do ficheiro menu.H.

10.18.2 Documentação dos Construtores & Destrutor

10.18.2.1 Slang::Menu::Menu (std::string const & *titulo*) [inline, explicit]

Constrói um novo menu com o título dado.

Precondição:
titulo <> "" e eImprimivel(titulo).

Poscondição:
titulo() = titulo.

Definido na linha 95 do ficheiro menu_impl.H.

Referências titulo().

10.18.2.2 Slang::Menu::~Menu () [inline, pure virtual]

Destrutor abstracto marca uma classe abstracta.

Precondição:
V.

Definido na linha 103 do ficheiro menu_impl.H.

10.18.3 Documentação dos métodos

10.18.3.1 std::string Slang::Menu::titulo () const [inline]

Devolve o título do menu.

Precondição:
V.

Poscondição:
titulo = título do menu.

Definido na linha 108 do ficheiro menu_impl.H.

Referenciado por Menu(), Slang::MenuComCor::MenuComCor(), Slang::MenuDeCores::MenuDeCores(), Slang::MenuDeSimOuNao::MenuDeSimOuNao() e Slang::MenuSimples::MenuSimples().

10.18.3.2 virtual int Slang::Menu::opcaoActual () const [pure virtual]

Devolve a opção actual do menu (o primeiro item do menu tem número 0).

Precondição:
V.

Poscondição:
opcaoActual = opção actual do menu.

Implementado em **Slang::MenuSimples** (p. 180) e **Slang::MenuDeSimOuNao** (p. 177).

10.18.3.3 virtual void Slang::Menu::interage () [pure virtual]

Executa o menu, i.e., interage com o utilizador do programa.

Precondição:
V.

Poscondição:
opcaoActual() é a última opção escolhida pelo utilizador.

Implementado em **Slang::MenuSimples** (p. 181).

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

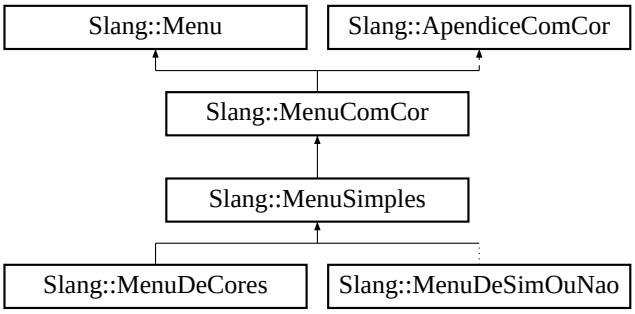
- **menu.H**
- **menu_impl.H**

10.19 Referência à classe Slang::MenuComCor

Classe abstracta que define a interface básica de todos os menus com cores.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::MenuComCor:



Membros públicos

Construtores

- **MenuComCor** (std::string const &titulo)
  Constrói um novo menu com o título dado.
- virtual ~**MenuComCor** ()=0
  Destrutor abstracto marca uma classe abstracta.

10.19.1 Descrição detalhada

Classe abstracta que define a interface básica de todos os menus com cores.

A parte das cores é herdada do apêndice **ApendiceComCor**.

Definido na linha 235 do ficheiro menu.H.

10.19.2 Documentação dos Construtores & Destrutor

10.19.2.1 Slang::MenuComCor::MenuComCor (std::string const & *titulo*) [inline, explicit]

Constrói um novo menu com o título dado.

Precondição:
titulo <> "" e eImprimivel(titulo).

Poscondição:
titulo() = titulo.

Definido na linha 123 do ficheiro menu_impl.H.

Referências Slang::Menu::titulo().

10.19.2.2 Slang::MenuComCor::~MenuComCor () [inline, pure virtual]

Destrutor abstracto marca uma classe abstracta.

Precondição:
V.

Definido na linha 130 do ficheiro menu_impl.H.

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

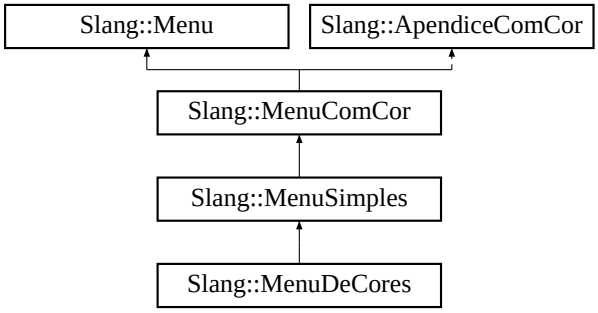
- **menu.H**
- **menu_impl.H**

10.20 Referência à classe Slang::MenuDeCores

Representa menus de selecção das cores básicas usáveis no ecrã.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::MenuDeCores:



Membros públicos

Construtores

- **MenuDeCores** (std::string const &titulo)
  Constrói um novo menu de cores com o título dado.

10.20.1 Descrição detalhada

Representa menus de selecção das cores básicas usáveis no ecrã.

Definido na linha 369 do ficheiro menu.H.

10.20.2 Documentação dos Construtores & Destrutor

10.20.2.1 Slang::MenuDeCores::MenuDeCores (std::string const & *titulo*) [inline, explicit]

Constrói um novo menu de cores com o título dado.

Precondição:
titulo <> "" e eImprimivel(titulo).

Poscondição:
titulo() = titulo.

Definido na linha 190 do ficheiro menu_impl.H.

Referências Slang::nomes_das_cores, Slang::numero_de_cores e Slang::Menu::titulo().

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

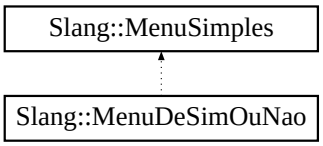
- **menu.H**
- **menu_impl.H**

10.21 Referência à classe Slang::MenuDeSimOuNao

Representa menus com apenas duas opções: sim e não.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::MenuDeSimOuNao:



Membros públicos

Construtores

- **MenuDeSimOuNao** (std::string const &titulo)
  Constrói um novo menu sim/não com o título dado.

Inspectores

- virtual int **opcaoActual** () const
  Devolve a opção actual do menu.

10.21.1 Descrição detalhada

Representa menus com apenas duas opções: sim e não.

Definido na linha 389 do ficheiro menu.H.

10.21.2 Documentação dos Construtores & Destrutor

10.21.2.1 Slang::MenuDeSimOuNao::MenuDeSimOuNao (std::string const & *titulo*) [inline, explicit]

Constrói um novo menu sim/não com o título dado.

Precondição:
titulo <> "" e eImprimivel(titulo).

Poscondição:
titulo() = titulo.

Definido na linha 198 do ficheiro menu_impl.H.

Referências Slang::Menu::titulo().

10.21.3 Documentação dos métodos

10.21.3.1 int Slang::MenuDeSimOuNao::opcaoActual () const [inline, virtual]

Devolve a opção actual do menu.

O valor 0 significa "não" e o valor 1 significa "não".

Precondição:
V.

Poscondição:
opcaoActual = opção actual do menu, i.e, a última opção escolhida pelo utilizador.

Reimplementado de **Slang::MenuSimples** (p. 180).

Definido na linha 204 do ficheiro menu_impl.H.

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

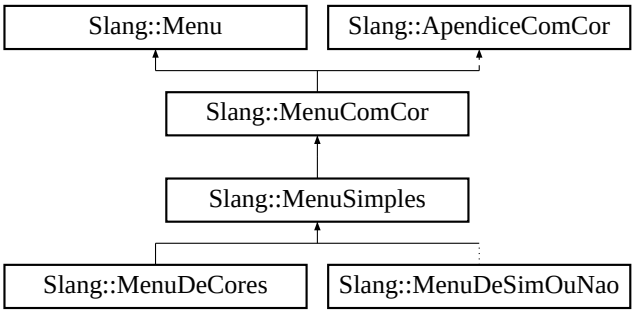
- **menu.H**
- **menu_impl.H**

10.22 Referência à classe Slang::MenuSimples

Representa menus simples, que consistem numa sequência de itens.

`#include <Slang++/menu.H>`

Diagrama de heranças da classe Slang::MenuSimples:



Membros públicos

Construtores

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

  Constrói um menu simples com o título dado e com os itens cujo texto é dado na forma de uma matriz clássica de cadeias de caracteres.

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

  Constrói um menu simples com o título dado e com os itens cujo texto é dado na forma de uma cadeia de caracteres onde os itens são terminados por '\n'.

Inspectores

- virtual int **opcaoActual** () const

  Devolve a opção actual do menu (o primeiro item do menu tem número 0).

Interface com o utilizador

- virtual void **interage** ()

  Executa o menu, i.e., interage com o utilizador do programa.

10.22.1 Descrição detalhada

Representa menus simples, que consistem numa sequência de itens.

O programa abaixo mostra um menu simples no ecrã e escreve a opção que for pressionada até ser seleccionada a opção "Basar":

```
#include <string>

#include <Slang++/slang.H>

using namespace std;

using namespace Slang;

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

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

    while(true) {
        menu.interage();

        if(menu.opcaoActual() == 0)
            break;

        ecra << parado << largura(20) << opcoes[menu.opcaoActual()] << refresca;
    }
}
```

O mesmo exemplo poderia ser escrito com mais simplicidade se não se precisasse de acedar às opções do menu individualmente:

```
#include <string>

#include <Slang++/slang.H>

using namespace std;

using namespace Slang;

int main ()
{
    MenuSimples menu("Um menu que não faz nada!",
                     "Basar\n"
                     "Nao faz nada...\n"
                     "Esta também não!\n"
                     "Nem esta...");

    while(true) {
        menu.interage();

        if(menu.opcaoActual() == 0)
            break;

        ecra << parado << largura(20) << menu.opcaoActual() << refresca;
    }
}
```

Invariante:

1 <= numero_de_itens e itens.size() == numero_de_itens e (Q j : 0 <= j < numero_de_itens : eImprimivel(itens[j])) e comprimento_maximo_dos_itens = max((M j: 0 <= j < numero_de_itens : itens[j].size()), **titulo()**.length() e 0 <= opcao_corrente < numero_de_itens.

Definido na linha 275 do ficheiro menu.H.

10.22.2 Documentação dos Construtores & Destrutor

10.22.2.1 Slang::MenuSimples::MenuSimples (std::string const & *titulo*, std::string const *itens*[], int const *numero_de_itens*) [inline]

Constrói um menu simples com o título dado e com os itens cujo texto é dado na forma de uma matriz clássica de cadeias de caracteres.

Precondição:
titulo <> "" e eImprimivel(titulo) e 1 <= numero_de_itens e itens tem numero_de_itens elementos e (Q j : 0 <= j < numero_de_itens : eImprimivel(itens[j])).

Poscondição:
titulo() = titulo.

Definido na linha 137 do ficheiro menu_impl.H.

Referências Slang::Menu::titulo().

10.22.2.2 Slang::MenuSimples::MenuSimples (std::string const & *titulo*, std::string const & *itens*)

Constrói um menu simples com o título dado e com os itens cujo texto é dado na forma de uma cadeia de caracteres onde os itens são terminados por '\n'.

Precondição:
titulo <> "" e eImprimivel(titulo) e itens tem pelo menos um item e os itens em itens são todos imprimíveis.

Poscondição:
titulo() = titulo.

Definido na linha 14 do ficheiro menu.C.

Referências Slang::Menu::titulo().

10.22.3 Documentação dos métodos

10.22.3.1 int Slang::MenuSimples::opcaoActual () const [inline, virtual]

Devolve a opção actual do menu (o primeiro item do menu tem número 0).

Precondição:
V.

Poscondição:
opcaoActual = opção actual do menu, i.e, a última opção escolhida pelo utilizador.

Implementa **Slang::Menu** (p. 171).

Reimplementado em **Slang::MenuDeSimOuNao** (p. 177).

Definido na linha 158 do ficheiro menu_impl.H.

10.22.3.2 void Slang::MenuSimples::interage () [virtual]

Executa o menu, i.e., interage com o utilizador do programa.

Precondição:
V.

Poscondição:
opcaoActual() é a última opção escolhida pelo utilizador.

Implementa **Slang::Menu** (p. 172).

Definido na linha 47 do ficheiro menu.C.

Referências Slang::Ecra::cola(), Slang::ecra, Slang::Ecra::foiRedimensionado(), Slang::fundo(), Slang::Teclado::haTeclaDisponivel(), Slang::Teclado::leProximaTeclaDisponivel(), Slang::Ecra::posicaoDoCursor(), Slang::refresca(), Slang::teclado, Slang::Teclado::teclaLida() e Slang::Ecra::trocoDoEcraCompleto().

Referenciado por Slang::Aviso::interage().

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

- **menu.H**
- **menu.C**
- **menu_impl.H**

10.23 Referência à classe Slang::Posicao

Representa a posição de uma célula do ecrã.

`#include <Slang++/ecra.H>`

Membros públicos

Construtores

- **Posicao** (int const linha=0, int const coluna=0)
  Constrói uma nova posição.

Inspectores

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

Modificadores

- void **mudaLinhaPara** (int const nova_linha)
  Muda a linha do ecrã correspondente à posição para nova_linha.
- void **mudaColunaPara** (int const nova_coluna)
  Muda a coluna do ecrã correspondente à posição para nova_coluna.

Serializadores

- **Posicao** (std::istream &entrada)
  Constrói uma posição por carregamento a partir de um canal de entrada.
- void **carregaDe** (std::istream &entrada)
  Carrega a posição a partir de um canal.
- void **guardaEm** (std::ostream &saida) const
  Guarda a posição num canal.

Operadores especiais de atribuição

- Posicao & **operator+=** (**Dimensao** const &deslocamento)
  Adiciona a posição de uma dimensão interpretada como um vector.
- Posicao & **operator-=** (**Dimensao** const &deslocamento)
  Subtrai a posição de uma dimensão interpretada como um vector.

Funções associadas

(Note que não são funções membro)

- Posicao const **operator+** (Posicao posicao, **Dimensao** const &deslocamento)

  Devolve a adição de uma posição com uma dimensão (a dimensão é interpretada como um vector).

- Posicao const **operator-** (Posicao posicao, **Dimensao** const &deslocamento)

  Devolve a subtracção de uma posição de uma dimensão (a dimensão é interpretada como um vector).

- Posicao const **operator+** (**Dimensao** const &deslocamento, Posicao posicao)

  Devolve a adição de uma posição com uma dimensão (a dimensão é interpretada como um vector).

- Posicao const **operator-** (**Dimensao** const &deslocamento, Posicao const &posicao)

  Devolve a subtracção de uma posição de uma dimensão (a dimensão é interpretada como um vector).

- **Dimensao** const **operator-** (Posicao const &destino, Posicao const &origem)

  Devolve a diferença entre duas posições, que é a dimensão correspondente ao vector entre as duas posições.

- Posicao const **operator-** (Posicao const &posicao)

  Devolve o simétrico de uma posição.

- bool **operator==** (Posicao const &uma_posicao, Posicao const &outra_posicao)

  Indica se duas posições são iguais.

- bool **operator!=** (Posicao const &uma_posicao, Posicao const &outra_posicao)

  Indica se duas posições são diferentes.

10.23.1 Descrição detalhada

Representa a posição de uma célula do ecrã.

As coordenadas de uma posição são inteiras e expressas em (linha, coluna). A linha 0 corresponde ao topo do ecrã e a coluna 0 corresponde ao lado esquerdo do ecrã. Note-se que conceptualmente o ecrã é ilimitado (ou melhor, é limitado apenas pela gama dos inteiros usados como coordenadas), o que significa que as posições podem ter coordenadas negativas.

É possível realizar operações aritméticas com posições e dimensões (i.e., valores da classe Dimensao). As posições são entendidas como pontuais, enquanto as dimensões são entendidas como vectoriais. Assim, a diferença entre duas posições é uma dimensão, sendo também possível adicinar ou subtrair dimensões a posições, de que resulta uma nova posição.

As posições podem ser inseridas no ecrã, usando o operador <<, o que tem como efeito colocar o cursor na posição inserida (ou na posição mais próxima do ecrã real, se o ecrã tiver o cursor limitado ao ecrã real).

Invariante:
V.

Definido na linha 69 do ficheiro ecra.H.

10.23.2 Documentação dos Construtores & Destrutor

10.23.2.1 Slang::Posicao::Posicao (int const *linha* = 0, int const *coluna* = 0) [inline, explicit]

Constrói uma nova posição.

Por omissão a posição corresponde ao canto superior esquerdo do ecrã.

Precondição:
V.

Poscondição:
linha() = *linha* e **coluna()** = *coluna*.

Definido na linha 19 do ficheiro ecra_impl.H.

Referenciado por carregaDe().

10.23.2.2 Slang::Posicao::Posicao (std::istream & *entrada*) [explicit]

Constrói uma posição por carregamento a partir de um canal de entrada.

Assume-se que os dados no canal têm um formato equivalente ao produzido pela operação **guardaEm()**.

Precondição:
entrada.good().

Poscondição:
Posicao é a posição que se encontrava no canal de entrada.

Excepções:
ErroAoCarregar é lançada se a construção por carregamento falhar.

Definido na linha 16 do ficheiro ecra.C.

10.23.3 Documentação dos métodos

10.23.3.1 int Slang::Posicao::linha () const [inline]

Devolve a linha do ecrã correspondente à posição.

Precondição:
V.

Poscondição:
linha() = linha a que a posição corresponde.

Definido na linha 26 do ficheiro ecra_impl.H.

Referenciado por Slang::Caixa::bordaContem(), Slang::Ecra::cola(), Slang::Caixa::contem(), Slang::Ecra::desenha(), Slang::Ecra::moveCursorPara(), Slang::Caixa::operator *=(), Slang::Caixa::operator+=(), Slang::Ecra::operator>>() e Slang::Ecra::trocoDe().

10.23.3.2 int Slang::Posicao::coluna () const [inline]

Devolve a coluna do ecrã correspondente à posição.

Precondição:
V.

Poscondição:
coluna() = coluna a que a posição corresponde.

Definido na linha 33 do ficheiro ecra_impl.H.

Referenciado por Slang::Caixa::bordaContem(), Slang::Ecra::cola(), Slang::Caixa::contem(), Slang::Ecra::desenha(), Slang::Ecra::moveCursorPara(), Slang::Caixa::operator *=(), Slang::Caixa::operator+=(), Slang::Ecra::operator>>() e Slang::Ecra::trocoDe().

10.23.3.3 void Slang::Posicao::mudaLinhaPara (int const *nova_linha*) [inline]

Muda a linha do ecrã correspondente à posição para *nova_linha.*

Precondição:
V.

Poscondição:
linha() = *nova_linha.*

Definido na linha 40 do ficheiro ecra_impl.H.

10.23.3.4 void Slang::Posicao::mudaColunaPara (int const *nova_coluna*) [inline]

Muda a coluna do ecrã correspondente à posição para *nova_coluna.*

Precondição:
V.

Poscondição:
coluna() = *nova_coluna.*

Definido na linha 49 do ficheiro ecra_impl.H.

10.23.3.5 void Slang::Posicao::carregaDe (std::istream & *entrada*) [inline]

Carrega a posição a partir de um canal.

Assume-se que os dados no canal têm um formato equivalente ao produzido pela operação **guardaEm()**.

Precondição:
entrada.good().

Poscondição:
*this é a posição que se encontrava no canal de entrada.

Excepções:

Erro Ao Carregar é lançada se o carregamento falhar. Nesse caso o canal pode ficar parcialmente lido.

Definido na linha 58 do ficheiro ecra_impl.H.

Referências Posicao().

10.23.3.6 void Slang::Posicao::guardaEm (std::ostream & *saida*) const

Guarda a posição num canal.

O formato produzido é compatível com o que o método **carregaDe()** espera.

Precondição:

saida.good().

Excepções:

Erro Ao Guardar é lançada se a operação falhar. Nesse caso o canal pode ficar parcialmente escrito.

Definido na linha 28 do ficheiro ecra.C.

Referenciado por Slang::Caixa::guardaEm().

10.23.3.7 Slang::Posicao & Slang::Posicao::operator+= (Dimensao const & *deslocamento*) [inline]

Adiciona a posição de uma dimensão interpretada como um vector.

Precondição:

*this = p.

Poscondição:

*this = p + *deslocamento.*

Definido na linha 68 do ficheiro ecra_impl.H.

Referências Slang::Dimensao::numeroDeColunas() e Slang::Dimensao::numeroDeLinhas().

10.23.3.8 Slang::Posicao & Slang::Posicao::operator-= (Dimensao const & *deslocamento*) [inline]

Subtrai a posição de uma dimensão interpretada como um vector.

Precondição:

*this = p.

Poscondição:

*this = p - *deslocamento.*

Definido na linha 81 do ficheiro ecra_impl.H.

Referências Slang::Dimensao::numeroDeColunas() e Slang::Dimensao::numeroDeLinhas().

10.23.4 Documentação das classes amigas e funções relacionadas

10.23.4.1 Posicao const operator+ (Posicao *posicao*, Dimensao const & *deslocamento*) [related]

Devolve a adição de uma posição com uma dimensão (a dimensão é interpretada como um vector).

Precondição:
p = posicao.

Poscondição:
operator+.**linha()** = p.linha() + deslocamento.numeroDeLinhas() e operator+.**coluna()** = p.coluna() + deslocamento.numeroDeColunas().

10.23.4.2 Posicao const operator- (Posicao *posicao*, Dimensao const & *deslocamento*) [related]

Devolve a subtracção de uma posição de uma dimensão (a dimensão é interpretada como um vector).

Precondição:
p = posicao.

Poscondição:
operator+.**linha()** = p.linha() - deslocamento.numeroDeLinhas() e operator+.**coluna()** = p.coluna() - deslocamento.numeroDeColunas().

10.23.4.3 Posicao const operator+ (Dimensao const & *deslocamento*, Posicao *posicao*) [related]

Devolve a adição de uma posição com uma dimensão (a dimensão é interpretada como um vector).

Precondição:
p = posicao.

Poscondição:
operator+.**linha()** = p.linha() + deslocamento.numeroDeLinhas() e operator+.**coluna()** = p.coluna() + deslocamento.numeroDeColunas().

10.23.4.4 Posicao const operator- (Dimensao const & *deslocamento*, Posicao const & *posicao*) [related]

Devolve a subtracção de uma posição de uma dimensão (a dimensão é interpretada como um vector).

Precondição:
V.

Poscondição:
operator+.**linha()** = deslocamento.numeroDeLinhas() - posicao.linha() e operator+.**coluna()** = deslocamento.numeroDeColunas() - posicao.coluna().

10.23.4.5 Dimensao const operator- (Posicao const & *destino*, Posicao const & *origem*) [related]

Devolve a diferença entre duas posições, que é a dimensão correspondente ao vector entre as duas posições.

Precondição:
V.

Poscondição:
operator+.numeroDeLinhas() = destino.linha() - origem.linha() e operator+.numeroDeColunas() = destino.coluna() - origem.coluna().

10.23.4.6 Posicao const operator- (Posicao const & *posicao*) [related]

Devolve o simétrico de uma posição.

Precondição:
V.

Poscondição:
operator-.**linha()** = -posicao.**linha()** e operator-.**coluna()** = -posicao.**coluna()**.

10.23.4.7 bool operator== (Posicao const & *uma_posicao*, Posicao const & *outra_posicao*) [related]

Indica se duas posições são iguais.

Precondição:
V.

Poscondição:
operator== = (uma_posicao.**linha()** = outra_posicao.linha() e uma_posicao.coluna() = outra_posicao.coluna()).

10.23.4.8 bool operator!= (Posicao const & *uma_posicao*, Posicao const & *outra_posicao*) [related]

Indica se duas posições são diferentes.

Precondição:
V.

Poscondição:
operator!= = (uma_posicao.**linha()** <> outra_posicao.linha() ou uma_posicao.coluna() <> outra_posicao.coluna()).

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

- **ecra.H**
- **ecra.C**
- **ecra_impl.H**

10.24 Referência à classe Slang::Tecla

Representa teclas premidas.

`#include <Slang++/teclado.H>`

Tipos Públicos

- enum **TeclaEnum** { **nula** = 0, **ctrl_a** = 1, **ctrl_b** = 2, **ctrl_c** = 3, **ctrl_d** = 4, **ctrl_e** = 5, **ctrl_f** = 6, **ctrl_g** = 7, **ctrl_h** = 8, **ctrl_i** = 9, **ctrl_j** = 10, **ctrl_k** = 11, **ctrl_l** = 12, **ctrl_m** = 13, **ctrl_n** = 14, **ctrl_o** = 15, **ctrl_p** = 16, **ctrl_q** = 17, **ctrl_r** = 18, **ctrl_s** = 19, **ctrl_t** = 20, **ctrl_u** = 21, **ctrl_v** = 22, **ctrl_w** = 23, **ctrl_x** = 24, **ctrl_y** = 25, **ctrl_z** = 26, **refresca** = 12, **cima** = 0x101, **baixo** = 0x102, **esquerda** = 0x103, **direita** = 0x104, **pagina_anterior** = 0x105, **pagina_seguinte** = 0x106, **casa** = 0x107, **fim** = 0x108, **a1** = 0x109, **a3** = 0x10A, **b2** = 0x10B, **c1** = 0x10C, **c3** = 0x10D, **refaz** = 0x10E, **desfaz** = 0x10F, **apaga_para_tras** = 0x110, **entrada** = 0xD, **insere** = 0x112, **apaga** = 0x113, **F0** = 0x200, **F1** = 0x201, **F2** = 0x202, **F3** = 0x203, **F4** = 0x204, **F5** = 0x205, **F6** = 0x206, **F7** = 0x207, **F8** = 0x208, **F9** = 0x209, **F10** = 0x20A, **F11** = 0x20B, **F12** = 0x20C, **erro** = 0xFFFF }

  Representa as várias teclas que podem ser premidas.

Membros públicos

Construtores

- **Tecla** (**TeclaEnum** tecla)

  Constrói nova tecla a partir do valor enumerado correspondente.

Conversores

- **operator TeclaEnum** () const

  Devolve a conversão da tecla no valor enumerado correspondente.

- char **comoChar** () const

  Devolve conversão da tecla para char.

Predicados

- bool **eDeDeslocamento** () const

  Indica se a tecla é de deslocamento (i.e., seta para a esquerda, direita, cima ou baixo).

- bool **eChar** () const

  Indica se a tecla for é um caractere.

10.24.1 Descrição detalhada

Representa teclas premidas.

As teclas podem corresponder a caracteres (e.g., 'a', 'x', '1', '.') ou a teclas de controlo (e.g., 'delete', 'home', etc.). Nem todas as teclas são reconhecidas: depende da forma como o S-Lang lida com cada terminal.

Definido na linha 28 do ficheiro teclado.H.

10.24.2 Documentação das enumerações

10.24.2.1 enum Slang::Tecla::TeclaEnum

Representa as várias teclas que podem ser premidas.

Não são enumerados explicitamente os caracteres normais, mas os seus valores podem ser representados neste tipo enumerado. O par de tipos **Tecla::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 operações úteis.

Valores da enumeração:

nula Caractere nulo.

ctrl_a 'ctrl-a'.

ctrl_b 'ctrl-b'.

ctrl_c 'ctrl-c'.

ctrl_d 'ctrl-d'.

ctrl_e 'ctrl-e'.

ctrl_f 'ctrl-f'.

ctrl_g 'ctrl-g'.

ctrl_h 'ctrl-h'.

ctrl_i 'ctrl-i'.

ctrl_j 'ctrl-j'.

ctrl_k 'ctrl-k'.

ctrl_l 'ctrl-l'.

ctrl_m 'ctrl-m'.

ctrl_n 'ctrl-n'.

ctrl_o 'ctrl-o'.

ctrl_p 'ctrl-p'.

ctrl_q 'ctrl-q'.

ctrl_r 'ctrl-r'.

ctrl_s 'ctrl-s'.

ctrl_t 'ctrl-t'.

ctrl_u 'ctrl-u'.

ctrl_v 'ctrl-v'.

ctrl_w 'ctrl-w'.

ctrl_x 'ctrl-x'.

ctrl_y 'ctrl-y'.

ctrl_z 'ctrl-z'.

refresca Refrescar ('ctrl-l').

cima Ir para cima.

baixo Ir para baixo.

esquerda Ir para a esquerda.

direita Ir para a direita.

pagina_anterior Passar à página anterior.

pagina_seguinte Passar à página seguinte.

casa Voltar a casa.

fim Ir para o fim.

a1

a3

b2

c1

c3

refaz Refazer (só em alguns teclados).

desfaz Desfazer (só em alguns teclados).

apaga_para_tras Apaga para trás.

entrada Dar entrada.

insere Inserir.

apaga Apagar.

F0

F1 'F1'.

F2 'F2'.

F3 'F3'.

F4 'F4'.

F5 'F5'.

F6 'F6'.

F7 'F7'.

F8 'F8'.

F9 'F9'.

F10 'F10'.

F11 'F11'.

F12 'F12'.

erro Valor em caso de erro.

Definido na linha 38 do ficheiro teclado.H.

10.24.3 Documentação dos Construtores & Destrutor

10.24.3.1 Slang::Tecla::Tecla (TeclaEnum *tecla*) [inline]

Constrói nova tecla a partir do valor enumerado correspondente.

Define conversão implícita a partir de Tecla::TeclaEnum).

Precondição:
V.

Poscondição:
TeclaEnum(*this) = tecla.

Definido na linha 15 do ficheiro teclado_impl.H.

10.24.4 Documentação dos métodos

10.24.4.1 Slang::Tecla::operator TeclaEnum () const

Devolve a conversão da tecla no valor enumerado correspondente.

Define conversão implícita para **Tecla::TeclaEnum**.

Precondição:
V.

Poscondição:
TeclaEnum = valor do enumerado correspondente à tecla.

10.24.4.2 char Slang::Tecla::comoChar () const [inline]

Devolve conversão da tecla para char.

Precondição:
eChar().

Poscondição:
comoChar = caractere correspondente à tecla representada.

Definido na linha 23 do ficheiro teclado_impl.H.

Referências eChar().

Referenciado por Slang::CaixaDeTexto::interage().

10.24.4.3 bool Slang::Tecla::eDeDeslocamento () const [inline]

Indica se a tecla é de deslocamento (i.e., seta para a esquerda, direita, cima ou baixo).

Precondição:
V.

Poscondição:
eDeDeslocamento = tecla representada é seta de deslocamento.

Definido na linha 30 do ficheiro teclado_impl.H.

Referências baixo, cima, direita e esquerda.

Referenciado por Slang::Ecra::moveCursorDeAcordoCom().

10.24.4.4 bool Slang::Tecla::eChar () const [inline]

Indica se a tecla for é um caractere.

Precondição:
V.

Poscondição:
eChar = tecla representada é um caractere.

Definido na linha 36 do ficheiro teclado_impl.H.

Referenciado por comoChar() e Slang::CaixaDeTexto::interage().

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

- **teclado.H**
- **teclado_impl.H**

10.25 Referência à classe Slang::Teclado

Representa o teclado.

`#include <Slang++/teclado.H>`

Membros públicos

Construtores

- **Teclado** ()
  Constrói um novo teclado.
- **~Teclado** ()
  Destrói o teclado.

Inspectores

- **Tecla teclaLida** () const
  Devolve a última tecla lida.

Predicados

- bool **haTeclaDisponivel** (int decimos_de_segundo=0) const
  Indica se alguma tecla está disponível para leitura.

Modificadores

- void **leProximaTeclaDisponivel** ()
  Lê a primeira das teclas premidas e ainda não lidas.
- void **descartaTeclasDisponiveis** ()
  Descarta todas as teclas premidas e ainda não lidas.

10.25.1 Descrição detalhada

Representa o teclado.

Esta classe é um solitão, pois admite-se não haver senão um teclado.

A classe **Teclado** permite obter informação acerca da pressão de teclas. Para isso usa-se a operação **teclaLida()**, que devolve uma instância da classe **Tecla**. Esta função não deve ser utilizada sem que haja uma tecla lida (i.e., sem que antes se tenha recorrido à operação leProximaTeclaDisponivel()).

A operação **leProximaTeclaDisponivel()** lê a próxima tecla disponível, i.e., a primeira das teclas premidas que ainda não foram lidas. Só deve ser utilizada se houver uma tecla disponível para leitura, a não ser que se pretenda que o programa páre a execução enquanto espera que o utilizador pressione uma tecla.

Para verificar se o utilizador pressionou numa tecla, existe o predicado **haTeclaDisponivel()**. Este predicado tem como argumento o número de décimos de segundo que se deve esperar pela pressão de uma tecla. Se o predicado for usado num ciclo, não se deve usar o valor zero como argumento, pois pode tornar o sistema muito lento (10 décimos de segundo é um valor aceitável para o argumento desta função nessas circunstâncias).

A operação **descartaTeclasDisponiveis()** 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**, 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 Tecla::TeclaEnum) ou um caractere normal. É possível verificar se o valor armazenado numa instância da classe **Tecla** é um deslocamento ou um caracter normal através dos predicados Tecla::eDeslocamento() e **Tecla::eChar()**.

Exemplo

O programa que se segue 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.haTeclaDisponivel(10)) {
            teclado.leProximaTeclaDisponivel();
            Tecla tecla_premida = teclado.teclaLida();

            if(tecla_premida.eDeDeslocamento())
                ecra.moveCursorDeAcordoCom(tecla_premida);
            else if(tecla_premida.eChar())
                if(isprint(tecla_premida.comoChar()))
                    ecra << cor_normal << tecla_premida.comoChar();
                else
                    ecra << cor_aviso << parado
                         << "Este nao se pode imprimir!";
            else
                ecra << cor_aviso << parado << "Tecla inválida!";
            if(tecla_premida == 's')
                break;
        }
        ecra.refresca();
    }
}
```

Definido na linha 215 do ficheiro teclado.H.

10.25.2 Documentação dos Construtores & Destrutor

10.25.2.1 Slang::Teclado::Teclado ()

Constrói um novo teclado.

Encarrega-se de inicializar o "keyboard interface" do S-Lang.

Precondição:
Não pode haver mais nenhum teclado definido.

Definido na linha 13 do ficheiro teclado.C.

Referências nada().

10.25.2.2 Slang::Teclado::~Teclado () [inline]

Destrói o teclado.

Encarrega-se de re-iniciar o "keyboard interface" do S-Lang, que volta ao estado em que estava antes de o teclado ter sido construído.

Precondição:
V.

Definido na linha 43 do ficheiro teclado_impl.H.

10.25.3 Documentação dos métodos

10.25.3.1 Slang::Tecla Slang::Teclado::teclaLida () const [inline]

Devolve a última tecla lida.

Precondição:
V.

Poscondição:
teclaLida() = última tecla lida ou **Tecla::nula**, caso ainda não tenha sido lida nenhuma tecla.

Definido na linha 54 do ficheiro teclado_impl.H.

Referenciado por Slang::CaixaDeTexto::interage() e Slang::MenuSimples::interage().

10.25.3.2 bool Slang::Teclado::haTeclaDisponivel (int *decimos_de_segundo* = 0) const [inline]

Indica se alguma tecla está disponível para leitura.

Desiste ao fim do tempo passado como argumento (em décimos de segundo).

Precondição:
0 <= decimos_de_segundo.

Poscondição:
haTeclaDisponivel = está disponível uma tecla para leitura.

Definido na linha 49 do ficheiro teclado_impl.H.

Referenciado por Slang::CaixaDeTexto::interage() e Slang::MenuSimples::interage().

10.25.3.3 void Slang::Teclado::leProximaTeclaDisponivel () [inline]

Lê a primeira das teclas premidas e ainda não lidas.

Se nenhuma tecla tiver sido premida, espera que tal aconteça.

Precondição:
V.

Poscondição:
teclaLida() = tecla lida pela operação.

Definido na linha 59 do ficheiro teclado_impl.H.

Referenciado por Slang::CaixaDeTexto::interage() e Slang::MenuSimples::interage().

10.25.3.4 void Slang::Teclado::descartaTeclasDisponiveis () [inline]

Descarta todas as teclas premidas e ainda não lidas.

Não afecta a última tecla lida.

Precondição:
V.

Poscondição:
nao **haTeclaDisponivel()**.

Definido na linha 64 do ficheiro teclado_impl.H.

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

- **teclado.H**
- **teclado_impl.H**
- **teclado.C**

Capítulo 11

Pacotes Documentação do ficheiro

11.1 Referência ao ficheiro cadeia.C

Ficheiro de implementação do módulo cadeias.

```
#include <Utilitarios/cadeia.H>
#include <cassert>
```

11.1.1 Descrição detalhada

Ficheiro de implementação do módulo cadeias.

Define ferramentas para somas valores a cadeias.

Definido no ficheiro **cadeia.C**.

11.2 Referência ao ficheiro cadeia.H

Ficheiro de interface do módulo cadeias.

`#include <string>`

`#include <Utilitarios/cadeia_impl.H>`

Namespaces

- namespace **Utilitarios**

11.2.1 Descrição detalhada

Ficheiro de interface do módulo cadeias.

Definido no ficheiro **cadeia.H**.

11.3 Referência ao ficheiro cadeia_impl.H

Ficheiro auxiliar de implementação do módulo cadeias.

```
#include <sstream>
```

11.3.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo cadeias.

Definido no ficheiro **cadeia_impl.H**.

11.4 Referência ao ficheiro caixa1.C

Programa que demostra o desenho de uma caixa no ecrã.

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

Funções

- int **main** ()

11.4.1 Descrição detalhada

Programa que demostra o desenho de uma caixa no ecrã.

Definido no ficheiro **caixa1.C**.

11.4.2 Documentação das funções

11.4.2.1 int main ()

Definido na linha 12 do ficheiro caixa1.C.

Referências Slang::ecra, Slang::refresca() e Slang::teclado.

11.5 Referência ao ficheiro campainha1.C

Este programa toca a campainha do ecrã a cada tecla premida, saindo quando se pressiona 's'.

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

Funções

- int **main** ()

11.5.1 Descrição detalhada

Este programa toca a campainha do ecrã a cada tecla premida, saindo quando se pressiona 's'.

Definido no ficheiro **campainha1.C**.

11.5.2 Documentação das funções

11.5.2.1 int main ()

Definido na linha 11 do ficheiro campainha1.C.

Referências Slang::campainha(), Slang::ecra, Slang::refresca() e Slang::teclado.

11.6 Referência ao ficheiro data.C

Ficheiro de implementação do módulo datas.

```
#include "Utilitarios/data.H"
#include <ctime>
```

11.6.1 Descrição detalhada

Ficheiro de implementação do módulo datas.

Define ferramentas que permitem lidar com datas e tempos.

Definido no ficheiro **data.C**.

11.7 Referência ao ficheiro data.H

Ficheiro de interface do módulo datas.

```
#include <iostream>
#include <string>
#include <Utilitarios/data_impl.H>
```

Namespaces

- namespace **Utilitarios**

11.7.1 Descrição detalhada

Ficheiro de interface do módulo datas.

Definido no ficheiro **data.H**.

11.8 Referência ao ficheiro data1.C

Programa que demostra a utilização de datas e tempos de forma simples.

```
#include <iostream>
#include <Utilitarios/data.H>
```

Funções

- int **main** ()

11.8.1 Descrição detalhada

Programa que demostra a utilização de datas e tempos de forma simples.

Definido no ficheiro **data1.C**.

11.8.2 Documentação das funções

11.8.2.1 int main ()

Definido na linha 14 do ficheiro data1.C.

Referências Utilitarios::setembro.

11.9 Referência ao ficheiro data_impl.H

Ficheiro auxiliar de implementação do módulo datas.

```
#include <cassert>
#include <Utilitarios/ignoradores.H>
#include <Erros/erros.H>
```

11.9.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo datas.

Definido no ficheiro **data_impl.H**.

11.10 Referência ao ficheiro ecra.C

Ficheiro de implementação do módulo ecras.

`#include <Slang++/ecra.H>`

`#include <Utilitarios/ignoradores.H>`

`#include <Erros/erros.H>`

11.10.1 Descrição detalhada

Ficheiro de implementação do módulo ecras.

Define classes que permitem usar o ecrã alfa-numérico.

Definido no ficheiro **ecra.C**.

11.11 Referência ao ficheiro ecra.H

Ficheiro de interface do módulo ecras.

```
#include <string>
#include <vector>
#include <iostream>
#include <limits>
#include <Slang++/teclado.H>
#include <Slang++/ecra_impl.H>
```

Namespaces

- namespace **Slang**

11.11.1 Descrição detalhada

Ficheiro de interface do módulo ecras.

Define a classe **Slang::Ecra** e outras classes associadas.

Tarefa

Ver se versões mais actualizadas do S-Lang indicam com rigor cores válidas no fundo e posições da palete; verificar se se dão bem com a Konsole.
A restrição do cursor ao ecrã real não está a ser feita sistematicamente. Por enquanto não é problemático, pois o ecrã solitão é contruído sem essa restrição e o consumidor do código não a pode impor a posteriori.
Na cologem de troços dever-se-ia guardar o objecto cor actual para o poder repor no final da colagem.
Escrever testes para o ecrã. É possível, pois pode-se analizar o conteúdo dos troços, em princípio. Mas é necessário alterar muito código.

Definido no ficheiro **ecra.H**.

11.12 Referência ao ficheiro ecra1.C

Programa que demostra a utilização do ecrã de forma simples.

```
#include <Slang++/slang.H>
#include <unistd.h>
```

Funções

- int **main** ()

11.12.1 Descrição detalhada

Programa que demostra a utilização do ecrã de forma simples.

Definido no ficheiro **ecra1.C**.

11.12.2 Documentação das funções

11.12.2.1 int main ()

Definido na linha 14 do ficheiro ecra1.C.

Referências Slang::amarelo, Slang::ao_centro, Slang::azul, Slang::baixaCursor(), Slang::cursor(), Slang::ecra, Slang::largura(), Slang::parado(), Slang::refresca() e Slang::teclado.

11.13 Referência ao ficheiro ecra2.C

Programa que demostra a utilização do troços de ecrã e respectiva cópia e colagem.

```
#include <string>
#include <Slang++/slang.H>
#include <unistd.h>
```

Funções

- int **main** ()

11.13.1 Descrição detalhada

Programa que demostra a utilização do troços de ecrã e respectiva cópia e colagem.

Definido no ficheiro **ecra2.C**.

11.13.2 Documentação das funções

11.13.2.1 int main ()

Definido na linha 20 do ficheiro ecra2.C.

Referências Slang::caixa(), Slang::cursor(), Slang::ecra e Slang::refresca().

11.14 Referência ao ficheiro ecra3.C

Programa que permite ao utilizador deslocar o cursor através das setas do teclado, indicando em cada instante se o cursor está ou não visível.

```
#include <string>
```

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

Funções

- int **main** ()

11.14.1 Descrição detalhada

Programa que permite ao utilizador deslocar o cursor através das setas do teclado, indicando em cada instante se o cursor está ou não visível.

É capaz de se adaptar a redimensionamentos do ecrã.

Definido no ficheiro **ecra3.C**.

11.14.2 Documentação das funções

11.14.2.1 int main ()

Definido na linha 14 do ficheiro ecra3.C.

Referências Slang::ao_centro, Slang::apaga(), Slang::cursor(), Slang::ecra, Slang::largura(), Slang::refresca() e Slang::teclado.

11.15 Referência ao ficheiro ecra4.C

Programa que demonstra que a alteração das cores do fundo do ecrã altera todas as células do ecrã que com ele tenham sido desenhadas.

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

Funções

- int **main** ()

11.15.1 Descrição detalhada

Programa que demonstra que a alteração das cores do fundo do ecrã altera todas as células do ecrã que com ele tenham sido desenhadas.

Definido no ficheiro **ecra4.C**.

11.15.2 Documentação das funções

11.15.2.1 int main ()

Definido na linha 11 do ficheiro ecra4.C.

Referências Slang::baixaCursor(), Slang::branco, Slang::ecra, Slang::fundo(), Slang::parado(), Slang::refresca(), Slang::teclado, Slang::verde e Slang::vermelho.

11.16 Referência ao ficheiro ecra_impl.H

Ficheiro auxiliar de implementação do módulo ecra.

```
#include <cassert>
#include <sstream>
#include <algorithm>
#include <slang.h>
#include <signal.h>
```

11.16.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo ecra.

Definido no ficheiro **ecra_impl.H**.

11.17 Referência ao ficheiro erros.H

Ficheiro de interface da biblioteca **Erros** e do módulo erros.

`#include <Erros/erros_impl.H>`

Namespaces

- namespace **Erros**

11.17.1 Descrição detalhada

Ficheiro de interface da biblioteca **Erros** e do módulo erros.

Define as classe **Erros::Erro**, **Erros::Erro Ao Carregar** e **Erros::Erro Ao Guardar**.

Definido no ficheiro **erros.H**.

11.18 Referência ao ficheiro erros_impl.H

Ficheiro auxiliar de implementação do módulo erros.

11.18.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo erros.

Definido no ficheiro **erros_impl.H**.

11.19 Referência ao ficheiro exemplo1.C

Programa que demostra a utilização simultânea do teclado e do ecrã de forma simples.

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

Funções

- int **main** ()

11.19.1 Descrição detalhada

Programa que demostra a utilização simultânea do teclado e do ecrã de forma simples.

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.

Definido no ficheiro **exemplo1.C**.

11.19.2 Documentação das funções

11.19.2.1 int main ()

Definido na linha 20 do ficheiro exemplo1.C.

Referências Slang::ecra, Slang::parado(), Slang::refresca() e Slang::teclado.

11.20 Referência ao ficheiro exemplo2.C

Programa que demostra a utilização do ecrã e dos menus de forma simples.

```
#include <cstdlib>
#include <string>
#include <Slang++/slang.H>
```

Funções

- int **main** ()

11.20.1 Descrição detalhada

Programa que demostra a utilização do ecrã e dos menus de forma simples.

Definido no ficheiro **exemplo2.C**.

11.20.2 Documentação das funções

11.20.2.1 int main ()

Definido na linha 15 do ficheiro exemplo2.C.

Referências Slang::baixaCursor(), Slang::Cor, Slang::ecra e Slang::refresca().

11.21 Referência ao ficheiro exemplo3.C

Programa que demostra a utilização do teclado e do ecrã de forma simples.

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

Funções

- int **main** ()

11.21.1 Descrição detalhada

Programa que demostra a utilização do teclado e do ecrã de forma simples.

As teclas lidas são escritas no ecrã através do seu código e, no caso de correspoderem a caracteres, também na forma de caracteres (embora entre parênteses).

Definido no ficheiro **exemplo3.C**.

11.21.2 Documentação das funções

11.21.2.1 int main ()

Definido na linha 17 do ficheiro exemplo3.C.

Referências Slang::ecra, Slang::refresca() e Slang::teclado.

11.22 Referência ao ficheiro ignoradores.C

Ficheiro de implementação do módulo ignoradores.

`#include <Utilitarios/ignoradores.H>`

11.22.1 Descrição detalhada

Ficheiro de implementação do módulo ignoradores.

Define manipuladores para ignorar caracteres.

Definido no ficheiro **ignoradores.C**.

11.23 Referência ao ficheiro ignoradores.H

Ficheiro de interface do módulo ignoradores.

```
#include <iostream>
#include <Utilitarios/ignoradores_impl.H>
```

Namespaces

- namespace **Utilitarios**

11.23.1 Descrição detalhada

Ficheiro de interface do módulo ignoradores.

Definido no ficheiro **ignoradores.H**.

11.24 Referência ao ficheiro ignoradores_impl.H

Ficheiro auxiliar de implementação do módulo ignoradores.

11.24.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo ignoradores.

Definido no ficheiro **ignoradores_impl.H**.

11.25 Referência ao ficheiro ipc.H

Ficheiro de interface da biblioteca `IPC++`.

`#include <IPC++/mensageiro.H>`

Namespaces

- namespace **IPC**

11.25.1 Descrição detalhada

Ficheiro de interface da biblioteca `IPC++`.

Usa-se se não se pretender discriminar entre os módulos físicos realmente necessários.

Definido no ficheiro **ipc.H**.

11.26 Referência ao ficheiro justificacao1.C

Programa que demostra os vários tipos de justificação e o seu efeito.

`#include <string>`

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

Funções

- int **main** ()

11.26.1 Descrição detalhada

Programa que demostra os vários tipos de justificação e o seu efeito.

Definido no ficheiro **justificacao1.C**.

11.26.2 Documentação das funções

11.26.2.1 int main ()

Definido na linha 12 do ficheiro justificacao1.C.

Referências Slang::a_direita, Slang::a_esquerda, Slang::amarelo, Slang::ao_centro, Slang::azul, Slang::baixaCursor(), Slang::ecra, Slang::largura(), Slang::parado(), Slang::refresca() e Slang::teclado.

11.27 Referência ao ficheiro localizacao.C

Ficheiro de implementação do módulo localizacao.

`#include <Utilitarios/localizacao.H>`

11.27.1 Descrição detalhada

Ficheiro de implementação do módulo localizacao.

Define ferramentas de localização para portugal.

Definido no ficheiro **localizacao.C**.

11.28 Referência ao ficheiro localizacao.H

Ficheiro de interface do módulo localizacao.

```
#include <string>
#include <locale>
#include <Utilitarios/localizacao_impl.H>
```

Namespaces

- namespace **Utilitarios**

11.28.1 Descrição detalhada

Ficheiro de interface do módulo localizacao.

Definido no ficheiro **localizacao.H**.

11.29 Referência ao ficheiro localizacao_impl.H

Ficheiro auxiliar de implementação do módulo localizacao.

```
#include <stdexcept>
```

11.29.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo localizacao.

Definido no ficheiro **localizacao_impl.H**.

11.30 Referência ao ficheiro manipulador1.C

Programa que define um manipulador que move o cursor para uma posição arbitrária mas visível do ecrã.

```
#include <cstdlib>
```

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

Funções

- void **cursor_aleatorio** (Ecra &**ecra**)
- int **main** ()

11.30.1 Descrição detalhada

Programa que define um manipulador que move o cursor para uma posição arbitrária mas visível do ecrã.

Definido no ficheiro **manipulador1.C**.

11.30.2 Documentação das funções

11.30.2.1 void cursor_aleatorio (Ecra & *ecra*)

Definido na linha 15 do ficheiro manipulador1.C.

Referências Slang::cursor() e Slang::ecra.

Referenciado por main().

11.30.2.2 int main ()

Definido na linha 21 do ficheiro manipulador1.C.

Referências cursor_aleatorio(), Slang::ecra, Slang::refresca() e Slang::teclado.

11.31 Referência ao ficheiro mensageiro.C

Ficheiro de implementação do módulo mensageiros.

```
#include <cassert>
#include <IPC++/mensageiro.H>
```

11.31.1 Descrição detalhada

Ficheiro de implementação do módulo mensageiros.

Define classes que permitem comunicar entre processos de uma forma simples.

Definido no ficheiro **mensageiro.C**.

11.32 Referência ao ficheiro mensageiro.H

Ficheiro de interface do módulo mensageiros.

```
#include <string>
#include <Erros/erros.H>
#include <Utilitarios/cadeia.H>
#include <sys/msg.h>
#include <sys/sem.h>
#include <unistd.h>
#include <pwd.h>
#include <IPC++/mensageiro_impl.H>
```

Namespaces

- namespace **IPC**

11.32.1 Descrição detalhada

Ficheiro de interface do módulo mensageiros.

Define a classe **IPC::Mensageiro**.

Definido no ficheiro **mensageiro.H**.

11.33 Referência ao ficheiro mensageiro1.C

Programa que demostra a utilização de mensageiros para comunicar entre processos.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <IPC++/mensageiro.H>
```

Namespaces

- namespace **std**

Funções

- int **main** ()

11.33.1 Descrição detalhada

Programa que demostra a utilização de mensageiros para comunicar entre processos.

Atenção! Execute duas instâncias (processos) este programa em simultâneo! Se não funcionar, dê o comando:

```
ipcrmall
```

Definido no ficheiro **mensageiro1.C**.

11.33.2 Documentação das funções

11.33.2.1 int main ()

Definido na linha 20 do ficheiro mensageiro1.C.

11.34 Referência ao ficheiro mensageiro_impl.H

Ficheiro auxiliar de implementação do módulo mensageiros.

```
#include <typeinfo>
#include <sstream>
#include <cerrno>
#include <Utilitarios/cadeia.H>
```

11.34.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo mensageiros.

Definido no ficheiro **mensageiro_impl.H**.

11.35 Referência ao ficheiro menu.C

Ficheiro de implementação do módulo menus.

`#include <Slang++/menu.H>`

`#include <Slang++/teclado.H>`

11.35.1 Descrição detalhada

Ficheiro de implementação do módulo menus.

Define classes que permitem usar menus.

Definido no ficheiro **menu.C**.

11.36 Referência ao ficheiro menu.H

Ficheiro de interface do módulo menus.

```
#include <vector>
#include <string>
#include <Slang++/ecra.H>
#include <Slang++/menu_impl.H>
```

Namespaces

- namespace **Slang**

11.36.1 Descrição detalhada

Ficheiro de interface do módulo menus.

Define classes que permitem usar menus.

Definido no ficheiro **menu.H**.

11.37 Referência ao ficheiro menu1.C

Programa que demostra a utilização de menus simples.

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

Funções

- int **main** ()

11.37.1 Descrição detalhada

Programa que demostra a utilização de menus simples.

Definido no ficheiro **menu1.C**.

11.37.2 Documentação das funções

11.37.2.1 int main ()

Definido na linha 14 do ficheiro menu1.C.

Referências Slang::ecra, Slang::largura(), Slang::parado() e Slang::refresca().

11.38 Referência ao ficheiro menu2.C

Programa que demonstra como se usa um menu simples.

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

Funções

- int **main** ()

11.38.1 Descrição detalhada

Programa que demonstra como se usa um menu simples.

Definido no ficheiro **menu2.C**.

11.38.2 Documentação das funções

11.38.2.1 int main ()

Definido na linha 14 do ficheiro menu2.C.

Referências Slang::ecra, Slang::largura(), Slang::parado() e Slang::refresca().

11.39 Referência ao ficheiro menu_impl.H

Ficheiro auxiliar de implementação do módulo menus.

```
#include <Utilitarios/localizacao.H>
```

11.39.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo menus.

Definido no ficheiro **menu_impl.H**.

11.40 Referência ao ficheiro objectocor1.C

Programa que demostra o efeito dos objectos cor e da alterção das suas cores já depois de feitas inserções por seu intermédio.

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

Funções

- int **main** ()

11.40.1 Descrição detalhada

Programa que demostra o efeito dos objectos cor e da alterção das suas cores já depois de feitas inserções por seu intermédio.

Definido no ficheiro **objectocor1.C**.

11.40.2 Documentação das funções

11.40.2.1 int main ()

Definido na linha 11 do ficheiro objectocor1.C.

Referências Slang::amarelo, Slang::azul, Slang::baixaCursor(), cor1(), Slang::ecra, Slang::fundo(), Slang::parado(), Slang::refresca() e Slang::teclado.

11.41 Referência ao ficheiro objectocor2.C

Programa que demonstra o efeito do tempo de vida sobre os objectos cor.

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

Funções

- Ecra::ObjectoCor const **cor1** (verde, branco)
- void **procedimento1** ()
- void **procedimento2** ()
- int **main** ()

11.41.1 Descrição detalhada

Programa que demonstra o efeito do tempo de vida sobre os objectos cor.

Quando um objecto cor é destruído, as células do ecrã com ele desenhadas ficam "órfãs", sendo "adoptadas" logo que se constrói um novo objecto cor. O resultado é que podem mudar de cor.

Definido no ficheiro **objectocor2.C**.

11.41.2 Documentação das funções

11.41.2.1 Ecra::ObjectoCor const cor1 (verde, branco)

Referenciado por main().

11.41.2.2 void procedimento1 ()

Definido na linha 15 do ficheiro objectocor2.C.

Referências Slang::amarelo, Slang::azul, Slang::baixaCursor(), Slang::ecra, Slang::parado(), Slang::refresca() e Slang::teclado.

Referenciado por main().

11.41.2.3 void procedimento2 ()

Definido na linha 24 do ficheiro objectocor2.C.

Referências Slang::amarelo, Slang::azul, Slang::baixaCursor(), Slang::ecra, Slang::parado(), Slang::refresca() e Slang::teclado.

Referenciado por main().

11.41.2.4 int main ()

Definido na linha 33 do ficheiro objectocor2.C.

Referências Slang::baixaCursor(), Slang::branco, cor1(), Slang::ecra, Slang::parado(), procedimento1(), procedimento2(), Slang::refresca(), Slang::teclado e Slang::verde.

11.42 Referência ao ficheiro objectocor3.C

Programa que demonstra que a alteração de um objecto cor altera todas as células do ecrã que com ele tenham sido desenhadas.

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

Funções

- int **main** ()

11.42.1 Descrição detalhada

Programa que demonstra que a alteração de um objecto cor altera todas as células do ecrã que com ele tenham sido desenhadas.

Definido no ficheiro **objectocor3.C**.

11.42.2 Documentação das funções

11.42.2.1 int main ()

Definido na linha 11 do ficheiro objectocor3.C.

Referências Slang::baixaCursor(), Slang::branco, Slang::ecra, Slang::parado(), Slang::refresca(), Slang::teclado, Slang::verde e Slang::vermelho.

11.43 Referência ao ficheiro pacotes.dox

Contém a introdução da documentação.

11.43.1 Descrição detalhada

Contém a introdução da documentação.

Definido no ficheiro **pacotes.dox**.

11.44 Referência ao ficheiro segmentos1.C

Programa que demonstra a utilização de segmentos de recta horizontais e verticais.

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

Funções

- int **main** ()

11.44.1 Descrição detalhada

Programa que demonstra a utilização de segmentos de recta horizontais e verticais.

Definido no ficheiro **segmentos1.C**.

11.44.2 Documentação das funções

11.44.2.1 int main ()

Definido na linha 11 do ficheiro segmentos1.C.

Referências Slang::apaga(), Slang::cursor(), Slang::ecra, Slang::largura(), Slang::refresca(), Slang::segmento_horizontal(), Slang::segmento_vertical() e Slang::teclado.

11.45 Referência ao ficheiro simbolo1.C

Programa que mostra os símbolos possíveis de inserir no ecrã.

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

Funções

- int **main** ()

11.45.1 Descrição detalhada

Programa que mostra os símbolos possíveis de inserir no ecrã.

Definido no ficheiro **simbolo1.C**.

11.45.2 Documentação das funções

11.45.2.1 int main ()

Definido na linha 10 do ficheiro simbolo1.C.

Referências Slang::diamante, Slang::diferente, Slang::ecra, Slang::largura(), Slang::refresca(), Slang::Simbolo e Slang::teclado.

11.46 Referência ao ficheiro slang.H

Ficheiro de interface da biblioteca `Slang++`.

`#include <Slang++/ecra.H>`

`#include <Slang++/teclado.H>`

`#include <Slang++/menu.H>`

Namespaces

- namespace **Slang**

11.46.1 Descrição detalhada

Ficheiro de interface da biblioteca `Slang++`.

Usa-se se não se pretender discriminar entre os módulos físicos realmente necessários.

Definido no ficheiro **slang.H**.

11.47 Referência ao ficheiro teclado.C

Ficheiro de implementação do módulo teclados.

`#include <Slang++/teclado.H>`

Funções

- void **nada** (int)

11.47.1 Descrição detalhada

Ficheiro de implementação do módulo teclados.

Define classes que permitem usar o teclado.

Definido no ficheiro **teclado.C**.

11.47.2 Documentação das funções

11.47.2.1 void nada (int)

Definido na linha 9 do ficheiro teclado.C.

Referenciado por Slang::Teclado::Teclado().

11.48 Referência ao ficheiro teclado.H

Ficheiro de interface do módulo teclados.

```
#include "teclado_impl.H"
```

Namespaces

- namespace **Slang**

11.48.1 Descrição detalhada

Ficheiro de interface do módulo teclados.

Define a classe **Slang::Teclado** e outras classes associadas.

Tarefa
Tentar escrever pelo menos testes elementares para as ferramentas deste módulo.

Definido no ficheiro **teclado.H**.

11.49 Referência ao ficheiro teclado1.C

Programa que demostra a utilização do teclado de forma simples.

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

Funções

- int **main** ()

11.49.1 Descrição detalhada

Programa que demostra a utilização do teclado de forma simples.

Definido no ficheiro **teclado1.C**.

11.49.2 Documentação das funções

11.49.2.1 int main ()

Definido na linha 14 do ficheiro teclado1.C.

Referências Slang::amarelo, Slang::ecra, Slang::parado(), Slang::preto, Slang::teclado e Slang::vermelho.

11.50 Referência ao ficheiro teclado_impl.H

Ficheiro auxiliar de implementação do módulo teclados.

```
#include <cassert>
#include <slang.h>
```

11.50.1 Descrição detalhada

Ficheiro auxiliar de implementação do módulo teclados.

Definido no ficheiro **teclado_impl.H**.

11.51 Referência ao ficheiro utilitarios.H

Ficheiro de interface da biblioteca **Utilitarios**.

```
#include <Utilitarios/ignoradores.H>
#include <Utilitarios/data.H>
#include <Utilitarios/cadeia.H>
#include <Utilitarios/localizacao.H>
```

Namespaces

- namespace **Utilitarios**

11.51.1 Descrição detalhada

Ficheiro de interface da biblioteca **Utilitarios**.

Usa-se se não se pretender discriminar entre os módulos físicos realmente necessários.

Definido no ficheiro **utilitarios.H**.

Capítulo 12

Pacotes Documentação da página

12.1 Lista de tarefas

Classe Slang::Caixa Falta implementar as operações de serialização.

Classe Slang::CaixaDeTexto Permitir a passagem de um functor de verificação de sintaxe, que torne a classe verdadeiramente genérica.

Classe Slang::Ecra::ObjectoCor Faltam as operações carregaDe() e guardaEm().

Dever-se-ia gerir melhor as posições da palete.

Ficheiro ecra.H Ver se versões mais actualizadas do S-Lang indicam com rigor cores válidas no fundo e posições da palete; verificar se se dão bem com a Konsole.

A restrição do cursor ao ecrã real não está a ser feita sistematicamente. Por enquanto não é problemático, pois o ecrã solitão é contruído sem essa restrição e o consumidor do código não a pode impor a posteriori.

Na cologem de troços dever-se-ia guardar o objecto cor actual para o poder repor no final da colagem.

Escrever testes para o ecrã. É possível, pois pode-se analizar o conteúdo dos troços, em princípio. Mas é necessário alterar muito código.

Ficheiro teclado.H Tentar escrever pelo menos testes elementares para as ferramentas deste módulo.

Namespace Slang Verificar todos os erros do S-Lang.

12.2 Lista de Bugs

Membro Utilitarios::eImprimivel (C const caractere) Se a localização para portugal não estiver disponível na máquina, esta função limita-se a devolver verdadeiro!

Membro Utilitarios::eImprimivel (std::basic_string< C > const &cadeia) Se a localização para portugal não estiver disponível na máquina, esta função limita-se a devolver verdadeiro!

Índice