Biblaureano

3.5

22 de Setembro de 2013

Sumário

Capítulo 1

Índice dos Componentes

1.1 Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

Capítulo 3

Classes

3.1 Referência da Classe Imagem

Classe para manipulação de imagens em modo texto, como sprites. Contém métodos para manipulação de imagens como pontos na tela.

`#include <biblaureano.h>`

Métodos Públicos

- Imagem ()

  Construtor da classe Imagem sem argumentos.
- Imagem (string sprite)

  Construtor da classe Imagem usando como argumento uma string com o sprite. Inicializa uma nova instância da classe Imagem.
- Imagem (string sprite, int px, int py)

  Construtor da classe Imagem usando como argumentos uma string com o sprite e as coordenadas da imagem. Inicializa uma nova instância da classe Imagem.
- void imprime (int px, int py)

  Altera as coordenadas da imagem e a imprime no novo posicionamento.
- void imprime ()

  Imprime a imagem na sua posicao atual.
- void limpa (int px, int py)

  Apaga a imagem conforme as novas coordenadas.
- void limpa ()

  Apaga a imagem da tela.
- void mudaCor (COR pCor)

  Muda a cor da imagem.
- void mudaCor (COR pCor, COR pCorFundo)

  Muda a cor da imagem.
- bool colisao (Imagem i)

  Verifica se houve colisão de imagens.
- bool colisao (Imagem i, int x1, int y1, int x2, int y2)

  Verifica se houve colisão de imagens.
- int getX ()

  Retorna a coordenada horizontal da imagem.
- int getY ()

  Retorna a coordenada vertical da imagem.

- vector< Ponto > getPontos ()

  Retorna os pontos da imagem.
- void setX (int px)

  Altera a coordenada horizontal da imagem.
- void setY (int py)

  Retorna a coordenada vertical da imagem.
- int incrementaX ()

  Aumenta a coordena horizontal da imagem em um.
- int incrementaY ()

  Aumenta a coordena vertical da imagem em um.
- int decrementaX ()

  Diminui a coordena horizontal da imagem em um.
- int decrementaY ()

  Diminui a coordena vertical da imagem em um.
- void setaPontos (vector< Ponto > ppontos)

  Altera a imagem.
- void setLimites (int _xMin, int _yMin, int _xMax, int _yMax)

  Seta as coordenadas limites que a imagem pode ter.
- int getColisaoX (Imagem i)

  Verifica em qual coluna houve colisão.
- int getColisaoY (Imagem i)

  Verifica em qual linha houve colisão.
- int getAltura ()

  Retorna a altura da imagem.
- int getLargura ()

  Retorna a largura da imagem.

3.1.1 Descrição Detalhada

Classe para manipulação de imagens em modo texto, como sprites. Contém métodos para manipulação de imagens como pontos na tela.

Veja também

Ponto

3.1.2 Construtores & Destrutores

3.1.2.1 Imagem::Imagem (string *sprite*)

Construtor da classe Imagem usando como argumento uma string com o sprite. Inicializa uma nova instância da classe Imagem.

Parâmetros

in	*sprite*	Sprite que forma a imagem.

Veja também

Imagem::Imagem(string sprite, int px, int py)

3.1.2.2 Imagem::Imagem (string *sprite*, int *px*, int *py*)

Construtor da classe Imagem usando como argumentos uma string com o sprite e as coordenadas da imagem. Inicializa uma nova instância da classe Imagem.

Parâmetros

in	*sprite*	Sprite que forma a imagem.
in	*px*	Coordenada X de início da imagem.
in	*py*	Coordenada Y de início da imagem.

Veja também

Imagem::Imagem(string sprite)

3.1.3 Métodos

3.1.3.1 bool Imagem::colisao (Imagem *i*)

Verifica se houve colisão de imagens.

Parâmetros

in	*i*	Imagem com a qual será feito o teste de colisão.

Retorna

true se houve colisão e false caso contrário.

Veja também

Imagem::colisao(Imagem i, int x1, int y1, int x2, int y2)

3.1.3.2 bool Imagem::colisao (Imagem *i*, int *x1*, int *y1*, int *x2*, int *y2*)

Verifica se houve colisão de imagens.

Parâmetros

in	*i*	Imagem com a qual será feito o teste de colisão.
in	*x1*	Coordenada horizontal da primeira imagem.
in	*y1*	Coordenada vertical da primeira imagem.
in	*x2*	Coordenada horizontal da segunda imagem.
in	*y2*	Coordenada vertical da segunda imagem.

Retorna

true se houve colisão e false caso contrário.

Veja também

Imagem::colisao(Imagem i)

3.1.3.3 int Imagem::getAltura ( )

Retorna a altura da imagem.

Retorna

Altura da imagem

3.1.3.4 int Imagem::getColisaoX (Imagem *i*)

Verifica em qual coluna houve colisão.

Parâmetros

in	*i*	Imagem com a qual será feito o teste de colisão

Retorna

-1 se não houve colisão, caso contrário retorna a posição no eixo X em que houve a mesma.

3.1.3.5 int Imagem::getColisaoY (Imagem *i*)

Verifica em qual linha houve colisão.

Parâmetros

in	*i*	Imagem com a qual será feito o teste de colisão

Retorna

-1 se não houve colisão, caso contrário retorna a posição no eixo Y em que houve a mesma.

3.1.3.6 int Imagem::getLargura ( )

Retorna a largura da imagem.

Retorna

Largura da imagem

3.1.3.7 vector< Ponto > Imagem::getPontos ( )

Retorna os pontos da imagem.

Retorna

Um vetor de Pontos que formam o sprite.

3.1.3.8 int Imagem::getX ( )

Retorna a coordenada horizontal da imagem.

Retorna

Um inteiro com a coordenada X da imagem.

3.1.3.9 int Imagem::getY ( )

Retorna a coordenada vertical da imagem.

Retorna

Um inteiro com a coordenada Y da imagem.

3.1.3.10 void Imagem::imprime (int *px*, int *py*)

Altera as coordenadas da imagem e a imprime no novo posicionamento.

Parâmetros

in	*px*	Nova coordenada horizontal da imagem.
in	*py*	Nova coordenada vertical da imagem.

Veja também

Imagem::imprime()

3.1.3.11 void Imagem::imprime ( )

Imprime a imagem na sua posicao atual.

Veja também

Imagem::imprime(int px, int py)

3.1.3.12 void Imagem::limpa (int *px*, int *py*)

Apaga a imagem conforme as novas coordenadas.

Parâmetros

in	*px*	Nova coordenada horizontal.
in	*py*	Nova coordenada vertical.

Veja também

Imagem::limpa()

3.1.3.13 void Imagem::limpa ( )

Apaga a imagem da tela.

Veja também

Imagem::limpa(int px, int py)

3.1.3.14 void Imagem::mudaCor (COR *pCor*)

Muda a cor da imagem.

Parâmetros

in	*pCor*	Nova cor do texto da imagem.

Veja também

Imagem::mudaCor(COR pCor, COR pCorFundo)

3.1.3.15 void Imagem::mudaCor (COR *pCor,* COR *pCorFundo*)

Muda a cor da imagem.

Parâmetros

in	*pCor*	Nova cor do texto da imagem.
in	*pCorFundo*	Nova cor do fundo da imagem.

Veja também

Imagem::mudaCor(COR pCor)

3.1.3.16 void Imagem::setaPontos (vector< Ponto > *ppontos*)

Altera a imagem.

Parâmetros

in	*ppontos*	Vetor com os novos pontos da imagem.

3.1.3.17 void Imagem::setLimites (int *_xMin,* int *_yMin,* int *_xMax,* int *_yMax*)

Seta as coordenadas limites que a imagem pode ter.

Parâmetros

in	*_xMin*	Coordenada mínima no eixo X da imagem. Determina o quão à esquerda a imagem pode ficar.
in	*_yMin*	Coordenada mínima no eixo Y da imagem. Determina o quão para cima a imagem pode ficar.
in	*_xMax*	Coordenada máxima no eixo X da imagem. Determina o quão à direita a imagem pode ficar.
in	*_yMax*	Coordenada máxima no eixo Y da imagem. Determina o quão para baixo a imagem pode ficar.

3.1.3.18 void Imagem::setX (int *px*)

Altera a coordenada horizontal da imagem.

Parâmetros

in	*px*	Um inteiro com a nova coordenada X da imagem.

3.1.3.19 void Imagem::setY (int *py*)

Retorna a coordenada vertical da imagem.

Parâmetros

in	*py*	Um inteiro com a coordenada Y da imagem.

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

- biblaureano.h
- main.cpp

3.2 Referência da Classe Ponto

Classe para manipulação de pontos na tela. Contém métodos básicos para manipulação de caracteres como pontos na tela.

`#include <biblaureano.h>`

Métodos Públicos

- Ponto (int px, int py, string pcaracter)

  Construtor da classe Ponto usando argumentos de coordenadas e caracter Inicializa uma nova instância da classe Ponto.
- Ponto (int px, int py, string pcaracter, COR pcor)

  Construtor da classe Ponto usando argumentos de coordenadas, caracter e cor Inicializa uma nova instância da classe Ponto.
- Ponto (int px, int py, string pcaracter, COR pcor, COR pcorFundo)

  Construtor da classe Ponto usando argumentos de coordenadas, caracter e cor Inicializa uma nova instância da classe Ponto.
- int getX ()

  Retorna o valor da coordenada X do ponto.
- int getY ()

  Retorna o valor da coordenada Y do ponto.
- COR getCor ()

  Retorna a cor do texto do ponto.
- COR getCorFundo ()

  Retorna a cor do fundo do ponto.
- void colore ()

  Atualiza a cor do ponto. Só atualiza se a cor foi alterada anteriormente pelos métodos setCor(COR pcor) ou setCor(COR pcor, COR pcorFundo)
- string getChar ()

  Retorna o caractere daquele ponto.

- void setCor (COR pcor)

  Altera a cor do texto daquele ponto.
- void setCor (COR pcor, COR pcorFundo)

  Altera a cor do texto e do fundo daquele ponto.
- void imprime (int px=0, int py=0)

  Imprime o ponto na tela.
- void limpa (int px=0, int py=0)

  Limpa a área em que o ponto foi impressa.

3.2.1 Descrição Detalhada

Classe para manipulação de pontos na tela. Contém métodos básicos para manipulação de caracteres como pontos na tela.

Veja também

Imagem

3.2.2 Construtores & Destrutores

3.2.2.1 Ponto::Ponto (int *px*, int *py*, string *pcaracter*)

Construtor da classe Ponto usando argumentos de coordenadas e caracter Inicializa uma nova instância da classe Ponto.

Parâmetros

in	*px*	Coordenada horizontal do ponto.
in	*py*	Coordenada vertical do ponto.
in	*pcaracter*	Caractere do ponto.

Veja também

Ponto::Ponto(int px, int py, string pcaracter, COR pcor)
Ponto::Ponto(int px, int py, string pcaracter, COR pcor, COR pcorFundo)

3.2.2.2 Ponto::Ponto (int *px*, int *py*, string *pcaracter*, COR *pcor*)

Construtor da classe Ponto usando argumentos de coordenadas, caracter e cor Inicializa uma nova instância da classe Ponto.

Parâmetros

in	*px*	Coordenada horizontal do ponto.
in	*py*	Coordenada vertical do ponto.
in	*pcaracter*	Caractere que será impresso naquele ponto.
in	*pcor*	Cor do texto daquele ponto. Pode ser omitido.

Veja também

Ponto::Ponto(int px, int py, string pcaracter)
Ponto::Ponto(int px, int py, string pcaracter, COR pcor, COR pcorFundo)

3.2.2.3 Ponto::Ponto (int *px,* int *py,* string *pcaracter,* COR *pcor,* COR *pcorFundo*)

Construtor da classe Ponto usando argumentos de coordenadas, caracter e cor Inicializa uma nova instância da classe Ponto.

Parâmetros

in	*px*	Coordenada horizontal do ponto.
in	*py*	Coordenada vertical do ponto.
in	*pcaracter*	Caractere que será impresso naquele ponto.
in	*pcor*	Cor do texto daquele ponto. Pode ser omitido.
in	*pcorFundo*	Cor do fundo do ponto. Pode ser omitido.

Veja também

Ponto::Ponto(int px, int py, string pcaracter)
Ponto::Ponto(int px, int py, string pcaracter, COR pcor)

3.2.3 Métodos

3.2.3.1 void Ponto::colore ( )

Atualiza a cor do ponto. Só atualiza se a cor foi alterada anteriormente pelos métodos setCor(COR pcor) ou setCor(COR pcor, COR pcorFundo)

Veja também

Ponto::setCor(COR pcor)
Ponto::setCor(COR pcor, COR pcorFundo)

3.2.3.2 string Ponto::getChar ( )

Retorna o caractere daquele ponto.

Retorna

Um a string contendo o caractere do ponto.

3.2.3.3 COR Ponto::getCor ( )

Retorna a cor do texto do ponto.

Retorna

Um valor do tipo COR com a cor do texto.

3.2.3.4 COR Ponto::getCorFundo ( )

Retorna a cor do fundo do ponto.

Retorna

Um valor do tipo COR com a cor do fundo do ponto.

3.2.3.5 int Ponto::getX ( )

Retorna o valor da coordenada X do ponto.

Retorna

Um inteiro com a coordenada X do ponto.

3.2.3.6 int Ponto::getY ( )

Retorna o valor da coordenada Y do ponto.

Retorna

Um inteiro com a coordenada Y do ponto.

3.2.3.7 void Ponto::imprime (int *px* = 0, int *py* = 0)

Imprime o ponto na tela.

Parâmetros

in	*px*	Coordenada X de inicio do sprite. Caso seja omitido o valor padrão é 0.
in	*py*	Coordenada Y de início do sprite. Caso seja omitido o valor padrão é 0.

Anotações

Os parâmetros são usados no caso de haver vários pontos, dessa forma, informando px e py, a função calcula o deslocamento que o ponto deve sofrer para ser impresso no local correto.

3.2.3.8 void Ponto::limpa (int *px* = 0, int *py* = 0)

Limpa a área em que o ponto foi impressa.

Parâmetros

in	*px*	Coordenada X de início do sprite. Caso seja omitido o valor padrão é 0.
in	*py*	Coordenada Y de início do sprite. Caso seja omitido o valor padrão é 0.

Anotações

Os parâmetros são usados no caso de haver vários pontos, dessa forma, informando px e py, a função calcula o deslocamento que deve fazer para apagar o local correto.

3.2.3.9 void Ponto::setCor (COR *pcor*)

Altera a cor do texto daquele ponto.

Parâmetros

in	*pcor*	Variável do tipo COR contendo a nova cor do ponto.

Veja também

Ponto::setCor(COR pcor, COR pcorFundo)
Ponto::colore()

3.2.3.10 void Ponto::setCor (COR *pcor,* COR *pcorFundo*)

Altera a cor do texto e do fundo daquele ponto.

Parâmetros

in	*pcor*	Variável do tipo COR contendo a nova cor do ponto.
in	*pcorFundo*	Variável do tipo COR contendo a nova cor de fundo do ponto.

Veja também

Ponto::setCor(COR pcor)
Ponto::colore()

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

- biblaureano.h
- main.cpp

Capítulo 4

Arquivos

4.1 Referência do ArquivoBiblaureano/biblaureano.h

Protótipos da Biblaureano.

```
#include <string>
#include <iostream>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <vector>
#include <algorithm>
#include <cstring>
#include <unistd.h>
#include <fcntl.h>
#include <fstream>
#include <iomanip>
```

Componentes

- class Ponto

  Classe para manipulação de pontos na tela. Contém métodos básicos para manipulação de caracteres como pontos na tela.
- class Imagem

  Classe para manipulação de imagens em modo texto, como sprites. Contém métodos para manipulação de imagens como pontos na tela.

Definições e Macros

- #define K_UP
- #define K_LEFT
- #define K_RIGHT
- #define K_DOWN
- #define END_FILE_CHARACTER 0x04
- #define TEMPO clock_t

Enumerações

- enum COR

Enumerador com as possíveis cores para alterar o texto e seu fundo.

4.1.1 Descrição Detalhada

Protótipos da Biblaureano.

4.1.2 LICENÇA

Autor

Marcos Laureano marcos.laureano@ifpr.edu.br
Gabriel Candido gabiru.vinicius@gmail.com
Thiago Romano

4.1.3 Definições e macros

!biblaureano.h@biblaureano.h

4.1.3.1 #define K_UP

Valor ASCII referente a seta para cima do teclado !biblaureano.h@biblaureano.h

4.1.3.2 #define K_LEFT

Valor ASCII referente a seta para a esquerda do teclado !biblaureano.h@biblaureano.h

4.1.3.3 #define K_RIGHT

Valor ASCII referente a seta para a direita do teclado !biblaureano.h@biblaureano.h

4.1.3.4 #define K_DOWN

Valor ASCII referente a seta para baixo do teclado

4.1.3.5 #define END_FILE_CHARACTER 0x04

Valor hexadecimal referente ao caractere de fim de arquivo

4.1.3.6 #define TEMPO clock_t

Definição para uso do termo TEMPO no lugar de clock_t, para melhor entendimento do código

4.1.4 Enumerações

4.1.4.1 enum COR

Enumerador com as possíveis cores para alterar o texto e seu fundo.

Anotações

Varia de acordo com o SO.
QTY_COR é a quantidade de cores possíveis. Usado para randomização de cores.

Veja também

mudaCor()

4.2 Referência do Arquivo Biblaureano/main.cpp

Biblioteca para auxiliar os alunos no decorrer dos cursos tecnicos em Informatica e em Programacao de Jogos Digitais do IFPR - Campus Salgado Filho.

```
#include "biblaureano.h"
```

Funções

- void mudaCor (COR NumeroCorLetra)

  Muda a cor do texto. Altera a cor de qualquer texto que venha depois da chamada dessa funcao, a nao ser que ela seja chamada novamente ou os efeitos de cor sejam limpados.
- void mudaCor (COR NumeroCorLetra, COR NumeroCorFundo)

  Muda a cor do texto e do fundo. Altera a cor de qualquer texto e fundo que venham depois da chamada dessa funcao, a nao ser que ela seja chamada novamente ou os efeitos de cor sejam limpados.
- void limpaEfeito ()

  Limpa efeitos do texto. Reseta qualquer alteracao da cor do texto e fundo, feitas anteriormente pelas funcoes mudaCor().
- int kbhit ()

  Verifica se alguma tecla foi pressionada. Nao para a execucao de um programa, pois nao aguarda uma operacao de I/O.
- int getch ()

  Verifica se alguma tecla foi pressionada. Espera por uma operacao de I/O, parando o programa.
- int verificaKB (char &tecla)

  Verifica se alguma tecla foi pressionada. Para a execução do programa. Útil quando você deve esperar que o usário digite algo, economizando processamento.
- void limparTela (void)

  Limpa a tela.
- void gotoXY (int x, int y)

  Seta a posicao do cursor do teclado.
- void desligaCursor (bool Liga)

  Altera a visibilidade do cursor do teclado.
- void apagaLinha (int yInicial, int yFinal)

Apaga um intervalo de linhas.

- void mudaTamanhoTerminal (int x, int y)

  Muda o tamanho da janela do programa.
- void noecho (bool liga)

  Retira o echo do output.
- long randomico (int inicial, int final)

  Gera um numero randomicamente.
- time_t tempoDecorrido (time_t Entrada)

  Calcula o tempo que se passou. Inicia uma contagem de tempo em segundos ou retorna o tempo que se passou.
- void espera (long int tempo)

  Pausa a execucao do programa.
- clock_t tempoInicio ()

  Usada para iniciar a contagem do tempo.
- int tempoPassado (clock_t inicio)

  Calcula quanto tempo se passou desde o início da contagem.
- int readInt (string mensagem)

  Le um inteiro do teclado. Le outro inteiro ate que o mesmo seja valido.
- float readFloat (string mensagem)

  Le um float do teclado. Le outro float ate que o mesmo seja valido.
- double readDouble (string mensagem)

  Le um double do teclado. Le outro valor ate que o mesmo seja valido.
- bool readBool (string mensagem)

  Le um valor booleano do teclado. Le outro bool ate que o mesmo seja valido.
- string readString (string mensagem)

  Le uma string do teclado. Le outra string ate que a mesma seja valida.
- char readChar (string mensagem)

  Le um caractere do teclado. Le outro caractere ate que o mesmo seja valido.
- string numeroToString (int valor)

  Converte um numero para string.
- string numeroToString (double valor)

  Converte um numero para string.
- string numeroToString (float valor)

  Converte um numero para string.
- void box (int xInicial, int yInicial, int xFinal, int yFinal, string sequencia)

  Desenha um quadrilatero na tela.
- void circulo (int x, int y, int raio)

  Desenha um circulo na tela.
- string retornaConteudoArquivo (string nomeArquivo)

  Retorna o conteudo de um arquivo. Podem acontecer falhas de acordo com o tipo do arquivo.
- vector< string > retornaArquivoSprites (string nomeArquivo, string separador)

  Retorna as sprites contidas em um arquivo de texto simples.
- vector< Imagem > retornaImagens (string nomeArquivo, string separador)

  Retorna as imagens criadas a partir de um arquivo de texto simples.
- void animaSprites (vector< string > sprites, int x, int y, int tempo)

  Imprime uma sequencia de sprites. Faz uma animação de diversos sprites.
- void imprimeSprite (string sprite, int x, int y)

  Imprime um sprite na tela O sprite é um arquivo de texto simples contendo strings de caracteres formando imagens.
- vector< Imagem > criaImagens (const string imagens[], int x, int y, int tamanho)

  Cria um vetor de imagens.
- vector< Imagem > criaImagens (const vector< string > imagens, int x, int y)

  Cria um vetor de imagens.

- vector< Imagem > criaImagens (const string imagem, int x, int y)

  Cria um vetor com uma imagem.
- void limpaArea (int xInicial, int yInicial, int xFinal, int yFinal)

  Limpa uma área da tela.
- string palavraAleatoria (string palavras)

  Randomiza uma palavra de um arquivo.
- int mostraMenuV (int x, int y, string opcoes[], int qtd, COR ativo, COR inativo)

  Mostra um menu vertical na tela.
- int mostraMenuH (int x, int y, string opcoes[], int qtd, COR ativo, COR inativo)

  Mostra um menu horizontal na tela.
- string geraLetraRandomico (int qtd)

  Gera letras randômicas.
- string geraLetraRandomicoMaiscula (int qtd)

  Gera letras maiúsculas randômicas.
- string geraLetraRandomicoMinuscula (int qtd)

  Gera letras minúsculas randômicas.
- Imagem modificaCorPontos (Imagem aColorir, Imagem referencia)

  Colore uma image, permite mais de uma cor na mesma imagem.

4.2.1 Descrição Detalhada

Biblioteca para auxiliar os alunos no decorrer dos cursos tecnicos em Informatica e em Programacao de Jogos Digitais do IFPR - Campus Salgado Filho.

4.2.2 LICENÇA

Biblaureano: biblioteca para o auxílio no desenvolvimento de jogos.

Copyright (C) 2011-2013 Marcos Laureano, Gabriel Candido e Thiago Romano

Este arquivo é parte do programa Biblaureano.

Biblaureano é um software livre; você pode redistribui-lo e/ou modifica-lo dentro dos termos da Licença Pública Geral GNU como publicada pela Fundação do Software Livre (FSF); na versão 2 da Licença, ou (na sua opnião) qualquer versão.

Este programa é distribuido na esperança que possa ser util, mas SEM NENHUMA GARANTIA; sem uma garantia implicita de ADEQUAÇÂO a qualquer MERCADO ou APLICAÇÃO EM PARTICULAR. Veja a Licença Pública Geral GNU para maiores detalhes.

Você deve ter recebido uma cópia da Licença Pública Geral GNU junto com este programa, se não, escreva para a Fundação do Software Livre(FSF) Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Autor

Marcos Laureano marcos.laureano@ifpr.edu.br
Gabriel Candido gabiru.vinicius@gmail.com
Thiago Romano

4.2.3 Funções

4.2.3.1 void animaSprites (vector< string > *sprites*, int *x*, int *y*, int *tempo*)

Imprime uma sequencia de sprites. Faz uma animação de diversos sprites.

Parâmetros

in	*sprites*	Vetor com os sprites.
in	*x*	Coordenada horizontal de início dos sprites.
in	*y*	Coordenada vertical de início dos sprites.
in	*tempo*	Intervalo de tempo que os sprites levam para ser alterados. Caso seja omitido o valor padrão é 1 segundo

Veja também

imprimeSprite(string sprite, int x, int y)
retornaArquivoSprites(string nomeArquivo, string separador)

4.2.3.2 void apagaLinha (int *yInicial,* int *yFinal*)

Apaga um intervalo de linhas.

Parâmetros

in	*yInicial*	Primeira valor do intervalo de linhas a serem apagadas.
in	*yFinal*	Ultimo valor do intervalo de linhas a serem apagadas.

Anotações

Caso yInicial e yFinal sejam iguais apaga somente a linha informada pelos dois argumentos.

4.2.3.3 void box (int *xInicial,* int *yInicial,* int *xFinal,* int *yFinal,* string *sequencia*)

Desenha um quadrilatero na tela.

Parâmetros

in	*xInicial*	Coluna onde o quadrilatero começa (Canto superior esquerdo).
in	*yInicial*	Linha onde o quadrilatero começa (Canto superior esquerdo).
in	*xFinal*	Coluna onde o quadrilatero termina (Canto inferior direito).
in	*yFinal*	Coluna onde o quadrilatero termina (Canto inferior direito).
in	*sequencia*	String de tres caracteres que determina o desenho do quadrilatero: O primeiro caractere e usado nos quatro cantos do box. O segundo caractere e usado nas linhas inferior e superior. O terceiro caractere e usado nas linhas laterais. Caso nao seja passado o padrao e "+-|".

4.2.3.4 void circulo (int *x,* int *y,* int *raio*)

Desenha um circulo na tela.

Parâmetros

in	*x*	Coluna do centro do circulo.
in	*y*	Linha do centro do circulo.
in	*raio*	Tamanho do raio do circulo.

4.2.3.5 vector<Imagem> crialmagens (const string *imagens[],* int *x,* int *y,* int *tamanho*)

Cria um vetor de imagens.

Parâmetros

in	*imagens*	Vetor de strings contendo as imagens.
in	*x*	Posição no eixo X da imagem.
in	*y*	Posição no eixo Y da imagem.
in	*tamanho*	Quantidade de imagens no vetor.

Retorna

Um vector de Imagens com as imagens criadas.

4.2.3.6 vector<Imagem> crialmagens (const vector< string > *imagens,* int *x,* int *y*)

Cria um vetor de imagens.

Parâmetros

in	*imagens*	Vector de strings contendo as imagens.
in	*x*	Posição no eixo X da imagem.
in	*y*	Posição no eixo Y da imagem.

Retorna

Um vector de Imagens com as imagens criadas.

4.2.3.7 vector<Imagem> crialmagens (const string *imagem,* int *x,* int *y*)

Cria um vetor com uma imagem.

Parâmetros

in	*imagem*	Nome do arquivo com as imagens. Deve ser usado o separador padrão da função retornaArquivoSprites().
in	*x*	Posição no eixo X da imagem.
in	*y*	Posição no eixo Y da imagem.

Retorna

Um vector de Imagem com a imagem criada.

4.2.3.8 void desligaCursor (bool *Liga*)

Altera a visibilidade do cursor do teclado.

Parâmetros

in	*Liga*	Se true, seta o cursor como invisivel, se false, seta como visível.

4.2.3.9 void espera (long int *tempo*)

Pausa a execucao do programa.

Parâmetros

in	*tempo*	Tempo, em milissegundos, que o programa sera pausado.

4.2.3.10 string geraLetraRandomico (int *qtd*)

Gera letras randômicas.

Parâmetros

in	*qtd*	Quantidade de letras a serem geradas.

Retorna

Uma string com as letras geradas.

4.2.3.11 string geraLetraRandomicoMaiscula (int *qtd*)

Gera letras maiúsculas randômicas.

Parâmetros

in	*qtd*	Quantidade de letras a serem geradas.

Retorna

Uma string com as letras geradas.

4.2.3.12 string geraLetraRandomicoMinuscula (int *qtd*)

Gera letras minúsculas randômicas.

Parâmetros

in	*qtd*	Quantidade de letras a serem geradas.

Retorna

Uma string com as letras geradas.

4.2.3.13 int getch ( )

Verifica se alguma tecla foi pressionada. Espera por uma operacao de I/O, parando o programa.

Veja também

kbhit()
verificaKB()

Retorna

A tecla pressionada.

4.2.3.14 void gotoXY (int *x*, int *y*)

Seta a posicao do cursor do teclado.

Parâmetros

in	*x*	Posicao do cursor no eixo X (coluna).
in	*y*	Posicao do cursor no eixo Y (linha).

4.2.3.15 void imprimeSprite (string *sprite*, int *x*, int *y*)

Imprime um sprite na tela O sprite é um arquivo de texto simples contendo strings de caracteres formando imagens.

Parâmetros

in	*sprite*	String
in	*x*	Coordenada horizontal de início do sprite. Se não for informado seu valor padrão é um.
in	*y*	Coordenada vertical de início do sprite. Se não for informado seu valor padrão é um.

Veja também

animaSprites(vector<string> sprites, int x, int y, int tempo)
retornaConteudoArquivo(string nomeArquivo)

4.2.3.16 int kbhit ( )

Verifica se alguma tecla foi pressionada. Nao para a execucao de um programa, pois nao aguarda uma operacao de I/O.

Veja também

getch( )
verificaKB()

Retorna

True se qualquer tecla foi pressionada, caso contrário retorna false.

4.2.3.17 void limpaArea (int *xInicial*, int *yInicial*, int *xFinal*, int *yFinal*)

Limpa uma área da tela.

Parâmetros

in	*xInicial*	Coordenada x do canto superior esquerdo da área a ser apagada. Se não for informado seu valor padrão é um - Windows - .
in	*yInicial*	Coordenada y do canto superior esquerdo da área a ser apagada. Se não for informado seu valor padrão é um.

in	*xFinal*	Coordenada x do canto inferior direito da área a ser apagada. Se não for informado seu valor padrão é 79.
in	*yFinal*	Coordenada y do canto inferior direito da área a ser apagada. Se não for informado seu valor padrão é 24.

Anotações

Caso nenhum parâmetro seja informado ele apaga a tela inteira, se essa estiver com o tamanho padrão.

4.2.3.18 void limpaEfeito ( )

Limpa efeitos do texto. Reseta qualquer alteracao da cor do texto e fundo, feitas anteriormente pelas funcoes mudaCor().

Veja também

mudaCor(COR NumeroCorLetra, COR NumeroCorFundo)
mudaCor(COR NumeroCorLetra)

4.2.3.19 Imagem modificaCorPontos (Imagem *aColorir,* Imagem *referencia*)

Colore uma image, permite mais de uma cor na mesma imagem.

Parâmetros

in	*aColorir*	Imagem base para ser colorida
in	*referencia*	Imagem, igual ao parâmetro aColorir, porém ao invés dos caracteres desejados, insere-se números de 1(um) a 8(oito), referentes a cor desejada para aquele ponto, conforme o enumerador COR

Retorna

Imagem com a cor dos pontos já setadas

4.2.3.20 int mostraMenuH (int *x,* int *y,* string *opcoes[],* int *qtd,* COR *ativo,* COR *inativo*)

Mostra um menu horizontal na tela.

Parâmetros

in	*x*	Coordenada horizontal do menu.
in	*y*	Coordenada vertical do menu.
in	*opcoes*	Vetor de strings contendo as opções do menu.
in	*qtd*	Quantidade de opções do menu.
in	*ativo*	Cor da opção atualmente selecionada. Caso seja omitido o valor padrão é azul.
in	*inativo*	Cor das opções não selecionadas. Caso seja omitido o valor padrão é branco.

Retorna

Um inteiro contendo a posição da opção escolhida.

4.2.3.21 int mostraMenuV (int *x*, int *y*, string *opcoes[]*, int *qtd*, COR *ativo*, COR *inativo*)

Mostra um menu vertical na tela.

Parâmetros

in	*x*	Coordenada horizontal do menu.
in	*y*	Coordenada vertical do menu.
in	*opcoes*	Vetor de strings contendo as opções do menu.
in	*qtd*	Quantidade de opções do menu.
in	*ativo*	Cor da opção atualmente selecionada. Caso seja omitido o valor padrão é azul.
in	*inativo*	Cor das opções não selecionadas. Caso seja omitido o valor padrão é branco.

Retorna

Um inteiro contendo a posição da opção escolhida.

4.2.3.22 void mudaCor (COR *NumeroCorLetra*)

Muda a cor do texto. Altera a cor de qualquer texto que venha depois da chamada dessa funcao, a nao ser que ela seja chamada novamente ou os efeitos de cor sejam limpados.

Parâmetros

in	*NumeroCorLetra*	Cor desejada para o texto, de acordo com o enumerador COR.

Veja também

mudaCor(COR NumeroCorLetra, COR NumeroCorFundo)
limpaEfeito()

4.2.3.23 void mudaCor (COR *NumeroCorLetra*, COR *NumeroCorFundo*)

Muda a cor do texto e do fundo. Altera a cor de qualquer texto e fundo que venham depois da chamada dessa funcao, a nao ser que ela seja chamada novamente ou os efeitos de cor sejam limpados.

Parâmetros

in	*NumeroCorLetra*	Cor desejada para o texto, de acordo com o enumerador COR.
in	*NumeroCorFundo*	Cor desejada para o fundo, de acordo com o enumerador COR.

Veja também

mudaCor(COR NumeroCorLetra)
limpaEfeito()

4.2.3.24 void mudaTamanhoTerminal (int *x*, int *y*)

Muda o tamanho da janela do programa.

Parâmetros

in	*x*	Novo numero de colunas da janela.
in	*y*	Novo numero de linhas da janela.

4.2.3.25 void noecho (bool *liga*)

Retira o echo do output.

Parâmetros

in	*liga*	Se true, seta para nao mostrar as teclas pressionadas durante a execucao do programa. O contrario e obtido com false.

Anotações

No sistema Windows nao e necessario o uso dessa funcao.

4.2.3.26 string numeroToString (int *valor*)

Converte um numero para string.

Parâmetros

in	*valor*	Inteiro a ser convertido.

Retorna

Uma string contendo o valor convertido.

4.2.3.27 string numeroToString (double *valor*)

Converte um numero para string.

Parâmetros

in	*valor*	Double a ser convertido.

Retorna

Uma string contendo o valor convertido.

4.2.3.28 string numeroToString (float *valor*)

Converte um numero para string.

Parâmetros

in	*valor*	Float a ser convertido.

Retorna

Uma string contendo o valor convertido.

4.2.3.29 string palavraAleatoria (string *palavras*)

Randomiza uma palavra de um arquivo.

Parâmetros

in	*palavras*	String contendo as palavras a serem randomizadas, separadas por um caractere de quebra de linha (’ ’).

Retorna

Uma string contendo a palavra aleatória.

4.2.3.30 long randomico (int *inicial,* int *final*)

Gera um numero randomicamente.

Parâmetros

in	*inicial*	Numero minimo a ser retornado.
in	*final*	Numero maximo a ser retornado. Se for omitido, nao ha limite maximo.

Anotações

Se ambos os parametros forem omitidos, entao o intervalo de numeros que podem ser retornados e o proprio limite da variavel.

Retorna

Um long int com o numero gerado randomicamente.

4.2.3.31 bool readBool (string *mensagem*)

Le um valor booleano do teclado. Le outro bool ate que o mesmo seja valido.

Parâmetros

in	*mensagem*	Mensagem a ser mostrada na tela antes da leitura.

Retorna

O bool lido.

4.2.3.32 char readChar (string *mensagem*)

Le um caractere do teclado. Le outro caractere ate que o mesmo seja valido.

Parâmetros

in	*mensagem*	Mensagem a ser mostrada na tela antes da leitura.

Retorna

O caractere lido.

4.2.3.33 double readDouble (string *mensagem*)

Le um double do teclado. Le outro valor ate que o mesmo seja valido.

Parâmetros

in	*mensagem*	Mensagem a ser mostrada na tela antes da leitura.

Retorna

O double lido.

4.2.3.34 float readFloat (string *mensagem*)

Le um float do teclado. Le outro float ate que o mesmo seja valido.

Parâmetros

in	*mensagem*	Mensagem a ser mostrada na tela antes da leitura.

Retorna

O float lido.

4.2.3.35 int readInt (string *mensagem*)

Le um inteiro do teclado. Le outro inteiro ate que o mesmo seja valido.

Parâmetros

in	*mensagem*	Mensagem a ser mostrada na tela antes da leitura.

Retorna

O inteiro lido.

4.2.3.36 string readString (string *mensagem*)

Le uma string do teclado. Le outra string ate que a mesma seja valida.

Parâmetros

in	*mensagem*	Mensagem a ser mostrada na tela antes da leitura.

Retorna

A string lida.

4.2.3.37 vector<string> retornaArquivoSprites (string *nomeArquivo,* string *separador*)

Retorna as sprites contidas em um arquivo de texto simples.

Parâmetros

in	*nomeArquivo*	String contendo o caminho e nome do arquivo que possui os sprites.
in	*separador*	String que funcionará como separador de sprites. Toda vez que houver esse separador a função irá considerar um novo sprite. O padrão é "*????????*"

Retorna

Um vetor de strings contendo os sprites.

Veja também

retornaConteudoArquivo(string nomeArquivo)
retornaImagens(string nomeArquivo, string separador)

4.2.3.38 string retornaConteudoArquivo (string *nomeArquivo*)

Retorna o conteudo de um arquivo. Podem acontecer falhas de acordo com o tipo do arquivo.

Parâmetros

in	*nomeArquivo*	String contendo o caminho e nome do arquivo.

Retorna

Uma string contendo todas as linhas do arquivo.

Veja também

retornaArquivoSprites(string nomeArquivo, string separador)

4.2.3.39 vector<Imagem> retornaImagens (string *nomeArquivo,* string *separador*)

Retorna as imagens criadas a partir de um arquivo de texto simples.

Parâmetros

in	*nomeArquivo*	String contendo o caminho e nome do arquivo que possui os sprites.
in	*separador*	String que funcionará como separador de imagens. Toda vez que houver esse separador a função irá considerar um novo imagens. O padrão é "*????????*"

Retorna

Um vetor de strings contendo as imagens.

Anotações

Possui a mesma utilidade que a função retornaArquivoSprites(string nomeArquivo, string separador), porém retorna um vetor do tipo Imagem e não do tipo string

4.2.3.40 time_t tempoDecorrido (time_t *Entrada*)

Calcula o tempo que se passou. Inicia uma contagem de tempo em segundos ou retorna o tempo que se passou.

Parâmetros

in	*Entrada*	Opcional, se nao for passado a funcao ira iniciar a contagem (ou reseta-la). Se for passado calculará o tempo decorrido desde a chamada que iniciou a contagem.

Retorna

O tempo decorrido, em segundos, desde a chamada que iniciou a contagem.

4.2.3.41 clock_t tempoInicio ( )

Usada para iniciar a contagem do tempo.

Retorna

Uma variável do tipo TEMPO que armazena o momento em que a contagem foi iniciada. É usada como parâmetro na função tempoPassado(clock_t inicio).

4.2.3.42 int tempoPassado (clock_t *inicio*)

Calcula quanto tempo se passou desde o início da contagem.

Parâmetros

in	*inicio*	Variável do tipo tempo que armazena quando a contagem foi iniciada. Deve ser inicializada com a função tempoInicio()

Retorna

O tempo que se passou, sendo que o retorno igual a 100 representa um segundo.

4.2.3.43 int verificaKB (char & *tecla*)

Verifica se alguma tecla foi pressionada. Para a execução do programa. Útil quando você deve esperar que o usário digite algo, economizando processamento.

Parâmetros

out	*tecla*	Tecla pressionada. Caso nenhuma tecla for pressionada seu valor nao e alterado.

Veja também

getch( )
kbhit()

Retorna

true se alguma tecla foi pressionada ou false, caso contrario.

Índice Remissivo