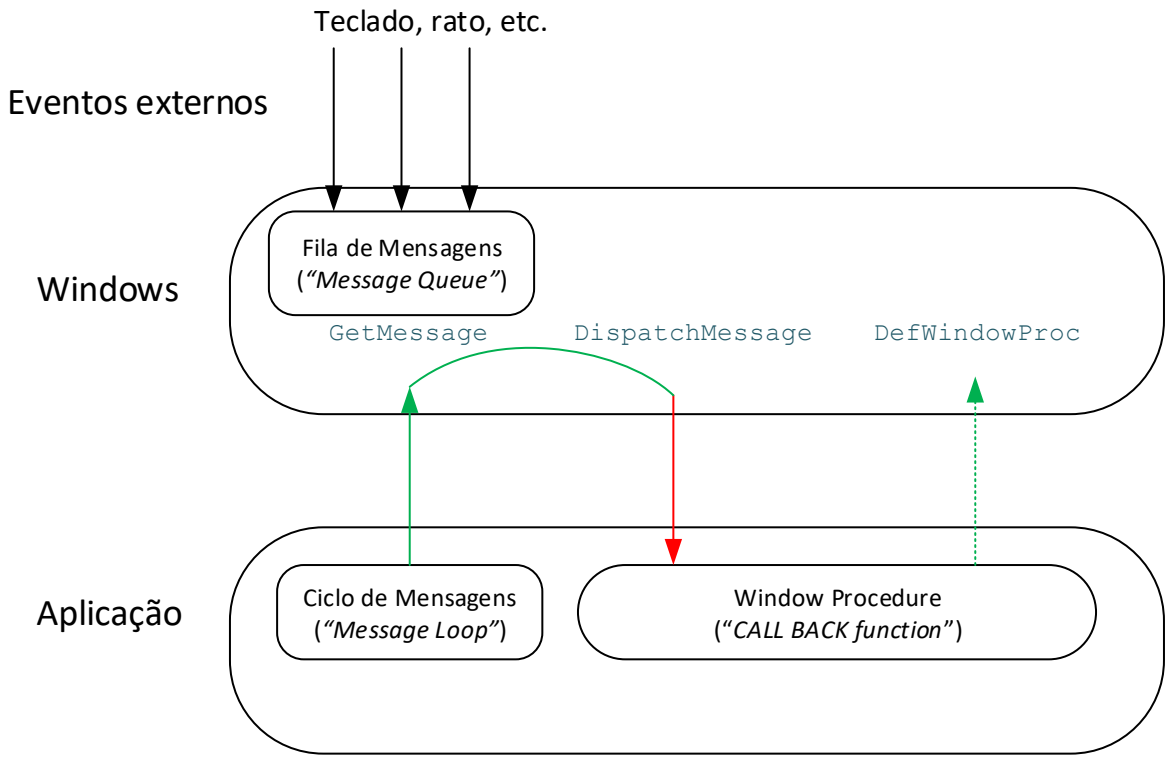
ISEL Instituto Superior de Engenharia de Lisboa	ESTRUTURAS DE DADOS E ALGORITMOS
Sumário: Ficha 4	Duração: 2 Aulas

INTRODUÇÃO

Neste trabalho pretende-se que o resultado do programa seja mostrado numa janela do Windows. Para isso há que esclarecer primeiramente alguns aspetos relativos à programação com janelas. Um programa do Windows® ou de outro sistema operativo semelhante é baseado em eventos (*"event-driven"*), isto quer dizer que o programa deve responder a eventos que acontecem de forma assíncrona, como por exemplo *clicks* ou movimentos no botão do rato, teclas pressionadas, etc.

A figura seguinte ilustra esquematicamente o que se pretende afirmar.



O Windows® apanha os vários eventos do teclado, do rato, etc., cada evento é convertido para uma mensagem sendo esta despachada de seguida para a janela apropriada. Por exemplo todas as mensagens do teclado vão diretamente para a janela que tem o **foco** (janela ativa). As mensagens do rato são despachadas de acordo com a posição do cursor, normalmente estas mensagens vão para a janela que está diretamente debaixo do cursor, etc.

Uma consequência desta situação é a obrigatoriedade na função *main* de cada aplicação existir um ciclo de mensagens:

```
int main()
{
. . .
      MSG msg ;
      while( GetMessage( &msg, 0, 0, 0 ) > 0 )
      {
         DispatchMessage(&msg);
         . . .
      }
}
```

A outra parte importante de cada aplicação em Windows é a função "CALLBACK" que se define e que o sistema operativo chama quando está a despachar as mensagens que chegam e que é responsável por parte de cada aplicação de tratar apenas as mensagens que pretender.

```
LRESULT CALLBACK WindowProcedure
    (HWND hwnd, unsigned int message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
        case WM_DESTROY:
            PostQuitMessage (0);
            return 0;
    }
    return DefWindowProc (hwnd, message, wParam, lParam );
}
```

Que como se verifica por este exemplo a aplicação apenas terminará quando um utilizador clicar na cruz da janela, iniciando uma mensagem do tipo WM_DESTROY e que provocará que esta função devolva o valor zero o que fará com que o ciclo de mensagens da função *main* termine.

PARTE 1

Pretende-se implementar um programa que desenhe um retângulo com uma cor na nova janela criada e que altera a sua cor assim que o utilizador *click* dentro ou fora do retângulo. Para isso são definidas as seguintes classes:

Classe Ponto – representa um ponto num plano XY

Ficheiro Ponto.h

```
#pragma once
class Ponto{
private:

	int x, y; // coordenadas do ponto

public:

	Ponto(); // construtor por omissão
	Ponto(int x0, int y0); //construtor para atribuição

	void AtribuirXY(int novo_x, int novo_y); // novas coordenadas
	float ObterDistancia(Ponto p2); // calcula a distancia a um ponto p2

	int ObterX(); // devolve a coordenada x
	int ObterY(); // devolve a coordenada y
};
```

Classe Rectangulo - representa um retângulo num plano XY. Definido por 2 pontos (canto superior esquerdo e inferior direito).

Ficheiro Rectangulo.h

```
#pragma once
#include <windows.h>
#include "Ponto.h"

class Rectangulo{
private:
	Ponto top_left, bottom_right;		// Cantos do rectângulo
public:
	Rectangulo(Ponto tl, Ponto br);
	Rectangulo(int x1, int y1, int x2, int y2);
	void Desenhar(HWND janelaId, long cor);  // Identificador da janela onde se
                                             // desenha o rectângulo
	bool Contem(Ponto p);    // Verifica se o ponto p está contido no rectangulo
                             // corrente
	bool Contem(Rectangulo r);  // Verifica se o rectângulo r está contido no
                                // rectangulo corrente
	double ObterArea();
	int ObterPerimetro();
};
```

A implementação da função **`Desenhar(HWND janelaId, long cor)`** da classe Rectangulo é dada a seguir e utiliza os métodos da biblioteca do Windows®.

Ficheiro Rectangulo.cpp

```
. . .
void Rectangulo::Desenhar(HWND janelaId, long cor)
{
	Ponto top_right(bottom_right. ObterX(), top_left. ObterY());
	Ponto bottom_left(top_left. ObterX(), bottom_right. ObterY());

	HDC DrawHDC = GetDC(janelaId);
	HBRUSH hOldBrush;
	HBRUSH hNewBrush;

	hNewBrush = CreateSolidBrush(cor);
	hOldBrush = (HBRUSH)SelectObject(DrawHDC, hNewBrush);
	RECT rect;
	rect.bottom = bottom_right.ObterY();
	rect.left = top_left. ObterX();
	rect.right = bottom_right. ObterX();
	rect.top = top_left. ObterY();
	FillRect(DrawHDC, &rect, hNewBrush)
	DeleteObject(SelectObject(DrawHDC, hOldBrush));
	ReleaseDC(janelaId, DrawHDC);
}
```

Classe Janela – representa uma janela do S.O. Windows®

Ficheiro Janela.h

```
#pragma once
#include <Windows.h>
#include "Ponto.h"

class Janela {
private:
	HWND janelaId;
	Ponto cur_coord;
	bool clickou;
	static Janela* objecto;
	static Janela* ObterObjecto();

public:
	Janela();
	~Janela() { };
	bool Criar(char* sTitulo);
	static LRESULT CALLBACK DespacharMensagens(HWND janId,
                                               unsigned int msg,
                                               WPARAM wp, LPARAM lp);
	HWND ObterId()      { return janelaId; }
	bool Click()        { return clickou; }
	Ponto ObterPonto()  { return cur_coord; }
};
```

Na declaração desta classe verifica-se que existe uma função que é declarada como estática ("*static*") que é a forma possível de uma função duma classe poder ser chamada como uma função *CALLBACK* por parte do sistema operativo (`static LRESULT CALLBACK DespacharMensagens`).

As propriedades desta classe são apenas aquelas que são necessárias para a execução do programa proposto, i.e.:

- janelaId – É o identificador da janela que se vai criar.
- cur_coord – Representa as coordenadas do cursor quando existir um click do rato em cima da janela.
- clickou – Variável booleana que indica se existiu ou não um click do rato em cima da janela.

A razão de existir um apontador para a classe Janela e ter sido declarado como estático tem a ver a função *DespacharMensagem* e o poder alterar valores do próprio objeto.

Ficheiro Janela.cpp

```
#include "Janela.h"
Janela* Janela::objecto = NULL;  // Objecto estático da classe Janela que é utilizado
                                 // dentro da função CALLBACK

Janela::Janela()
{
       janelaId = NULL;
       objecto = this;    // Uma vez criado o objecto estático é o próprio objecto
       clickou = false;
}
Janela* Janela::ObterObjecto()
{
       return objecto;
}
bool Janela::Criar(char* sTitulo)
{
    WNDCLASSEX wndclass = { sizeof(WNDCLASSEX), CS_DBLCLKS, DespacharMensagens,
                            0, 0, GetModuleHandle(0), LoadIcon(0,IDI_APPLICATION),
                            LoadCursor(0,IDC_ARROW), HBRUSH(CTLCOLOR_STATIC+1),
                            0, "minhaclasse", LoadIcon(0,IDI_APPLICATION) } ;
    if( RegisterClassEx(&wndclass) )
    { // Função do SO Windows que cria janelas
        janelaId = CreateWindowEx( 0, "minhaclasse", sTitulo,
                   WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT,
                   CW_USEDEFAULT, CW_USEDEFAULT, 0, 0, GetModuleHandle(0), 0 );
        if(janelaId)
        {
           ShowWindow( janelaId, SW_SHOWDEFAULT );
           return true;
        }
     }
     return false;
}
```

A função *DespacharMensagens* que é passada para o SO.

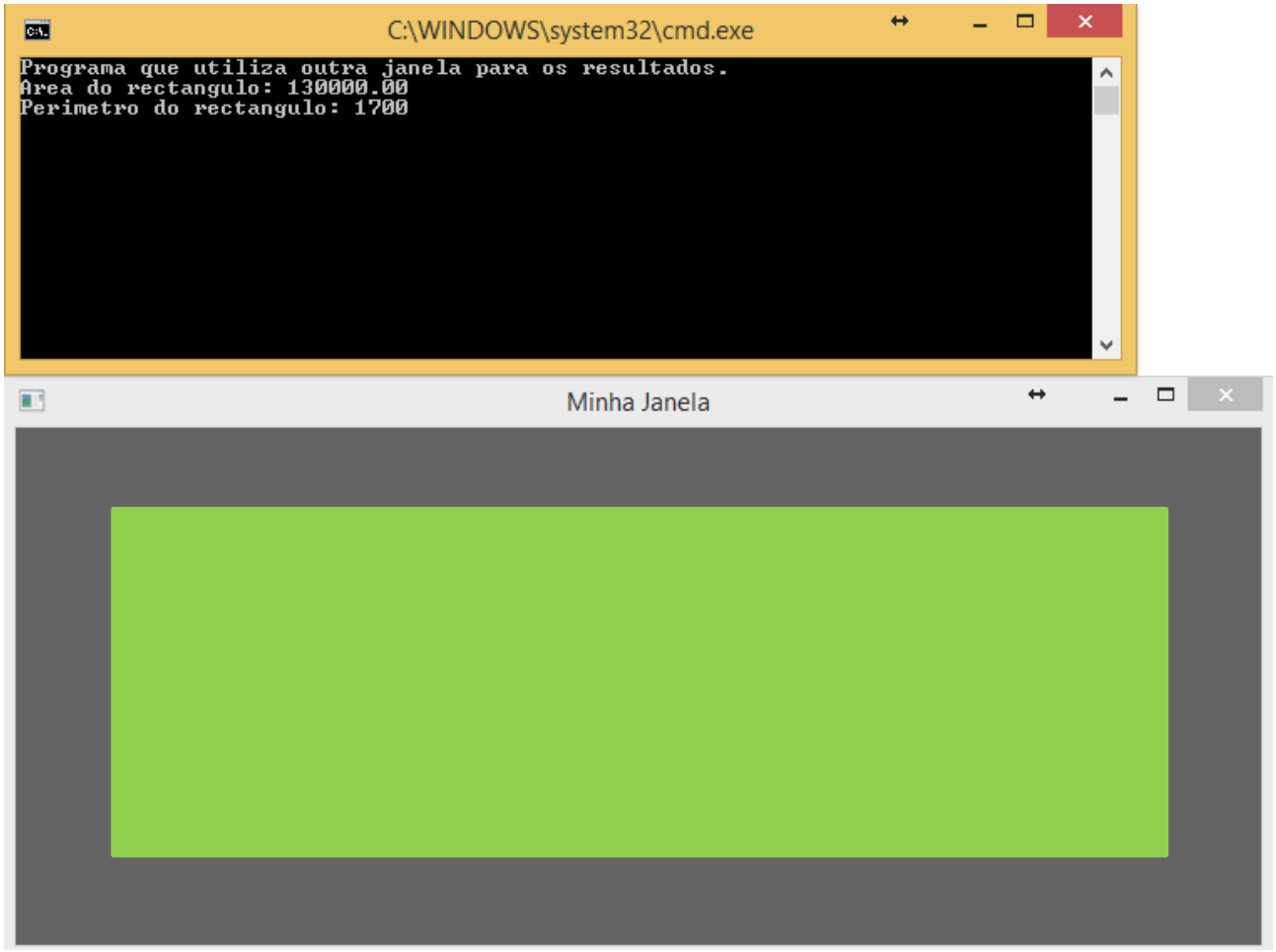
```
LRESULT CALLBACK Janela::DespacharMensagens(HWND janId, unsigned int msg,
                                            WPARAM wp, LPARAM lp)
{
    Janela* janela = ObterObjecto();
    janela->clickou = false;
    switch(msg) {
        case WM_DESTROY:
            PostQuitMessage(0) ;
            return 0;
         case WM_KEYDOWN:
            if (wp == VK_ESCAPE) {
                PostQuitMessage(0) ;
                return 0;
            }
             break;
        case WM_LBUTTONDOWN:
            janela->clickou = true;
            janela->cur_coord = Ponto(LOWORD(lp), HIWORD(lp));
            break;
        default:
            break;
    }
    return DefWindowProc(janId, msg, wp, lp);
}
```

A função *main* é a que se indica de seguida:

```
#include<stdio.h>
#include"Rectangulo.h"
#include"Janela.h"
#define VERDE RGB(0,255,0)
#define AZUL RGB(0,0,255)
#define VERMELHO RGB(255,0,0)

int main() {
   Janela janela;
   HWND janelaId;
   printf("Programa que utiliza outra janela para os resultados.\n");
   if (janela.Criar("Minha Janela")) {
      if ((janelaId=janela.ObterId()) != NULL) {
         Rectangulo r(242, 177, 892, 377, janelaId);
         printf("\nArea do rectangulo: %0.2f\n", r.obterArea());
         printf("\nPerimetro do rectangulo: %d\n", r.obterPerimetro());
         r.desenhar(janelaId, VERDE);
         MSG msg ;
         while( GetMessage( &msg, 0, 0, 0 ) ) {
            DispatchMessage(&msg);
            if (janela.Click()){
               if(r.contem(Ponto(janela.ObterPonto().ObterX(),
                           janela.ObterPonto().ObterY())) ==true)
                  r.desenhar(janelaId, VERDE);
               else
                  r.desenhar(janelaId, VERMELHO);
            }
         }
      }
   }
}
```

O resultado deste programa deve ser algo de semelhante ao que se indica na figura seguinte.



PARTE 2

Pretende-se implementar um programa que efetue os cálculos estatísticos já realizados na Ficha Nº 1 e acrescente a mediana, mas agora desenhe numa janela do Windows um gráfico de barras, adaptado ao tamanho da janela.
Deve criar uma nova classe que contenha esses dados e os desenhe graficamente sob a forma de gráfico de barras, criando os métodos que achar conveniente.

Para desenhar de forma proporcional o gráfico de barras existem os seguintes novos métodos na classe janela:

```
class Janela
{
 . . .
public:
. . .
       int ObterLargura();     // Devolve a largura da janela em pixeis
       int ObterAltura();     // Devolve a altura da janela em pixeis
       Rectangulo ObterJanelaRect (); // Devolve um objecto da classe Rectangulo com as
                                      // dimensões da janela
};
```

A função `ObterJanelaRect()` da classe Janela utiliza métodos da biblioteca do Windows® e a sua implementação pode ser a seguinte:

```
Rectangulo Janela::ObterJanelaRect()
{
       Rectangulo rect;
       RECT r;
       if (GetWindowRect(janelaId, &r))
              rect = Rectangulo(r.left, r.top, r.right, r.bottom);
       return rect;
}
```

Nota 1: Nesta ficha deve-se criar 2 projetos dentro da mesma solução do Visual Studio. Relembrar também que nas propriedades do projeto dentro das configurações gerais deve estar selecionada a seguinte opção:

- **Character Set: Not Set**

Nota 2: Para desenhar um histograma deve utilizar a biblioteca '**Histograma.dll**' que se encontra nos ficheiros anexos.

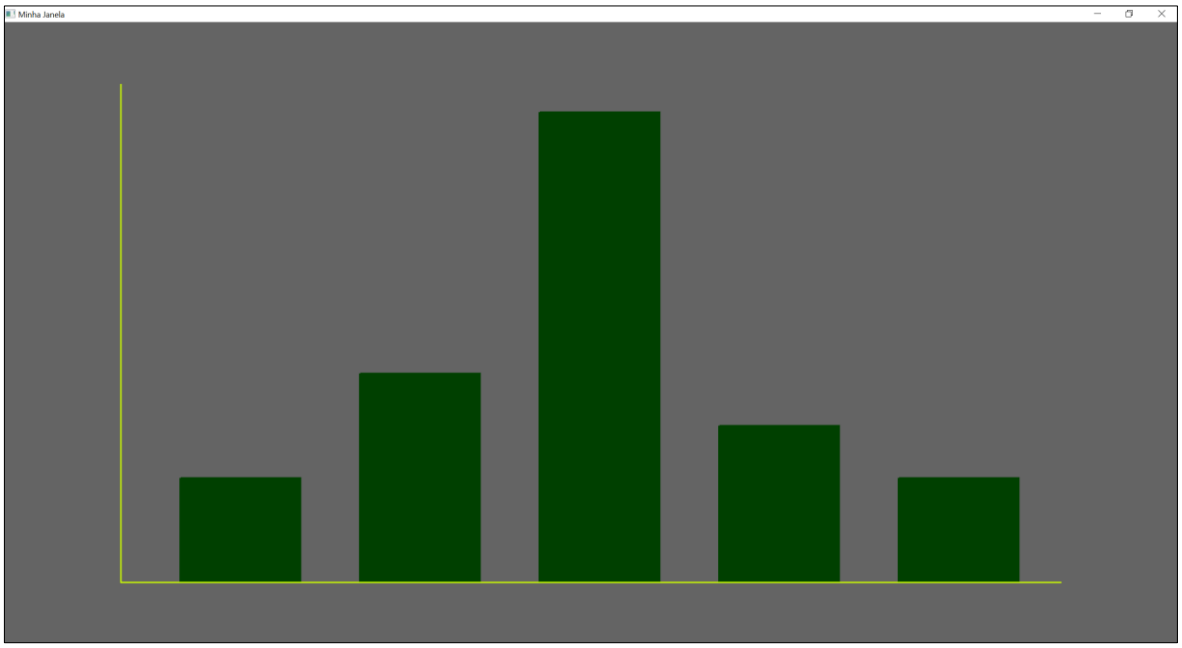
Exemplo de um programa que utiliza essa biblioteca:

```
#include "Janela.h"
#include "Hist.h"
#include<stdio.h>

int main()
{
	Janela janela;
	HWND janelaId;
	int bins[] = { 2, 4, 9, 3, 2 };
	int nbins = 5;
	Hist h(bins, nbins);

	h.SetCorBins(RGB(0, 64, 0));
	if (janela.Criar("Minha Janela")) {
		if ((janelaId = janela.ObterId()) != NULL) {
			h.Desenhar(&janela);

			MSG msg;
			while (GetMessage(&msg, 0, 0, 0)) {
				DispatchMessage(&msg);
			}
			printf("O programa terminou\n");
		}
	}
}
```



Existem 3 ficheiros anexos que são:

- **Histograma.lib**: Ficheiro que se tem de incluir no projeto para efetuar a *linkagem*. Deve-se primeiramente copiar para a pasta onde estão todos os restantes ficheiros do projeto. Seguidamente nas propriedades do projeto deve selecionar a opção Linker: Input e no campo de Additional Dependencies deve acrescentar o nome do ficheiro Histograma.Lib.
- **Histograma.Dll**: Ficheiro que possui as funções de biblioteca que implementam a classe Hist. Este ficheiro deve ser copiado para a pasta onde se encontra o executável do projeto.
- **Hist.h**: Ficheiro que tem a definição da classe Hist que queremos utilizar no nosso programa. Este ficheiro também deve ser copiado para a pasta onde existem os restantes ficheiros do projeto.

Ficheiro Hist.h

```
#pragma once
#include "Janela.h"

class Hist
{
private:
	int* pBins;
	int nBins;
	int vmax;
	long cor_eixos;
	long cor_bins;

public:
	Hist();
	Hist(int* pbins, int nbins);
	~Hist();

	void Desenhar(Janela* janela);
	void SetCorEixos(long cor);
	void SetCorBins(long cor);
};
```

A classe Hist apenas tem as funções de desenhar um conjunto de *bins* e alterar as cores de defeito dos *bins* e dos eixos.

Nota 3: Não é referido expressamente, mas é necessário definir uma nova classe que lê e guarda a amostra de dados que está no ficheiro e que define funções para as operações de leitura, cálculo da média, cálculo da mediana, desvio padrão, etc.

Nota 4: Para o cálculo da mediana deve implementar um dos algoritmos de ordenação que foram referidos nas aulas teóricas.