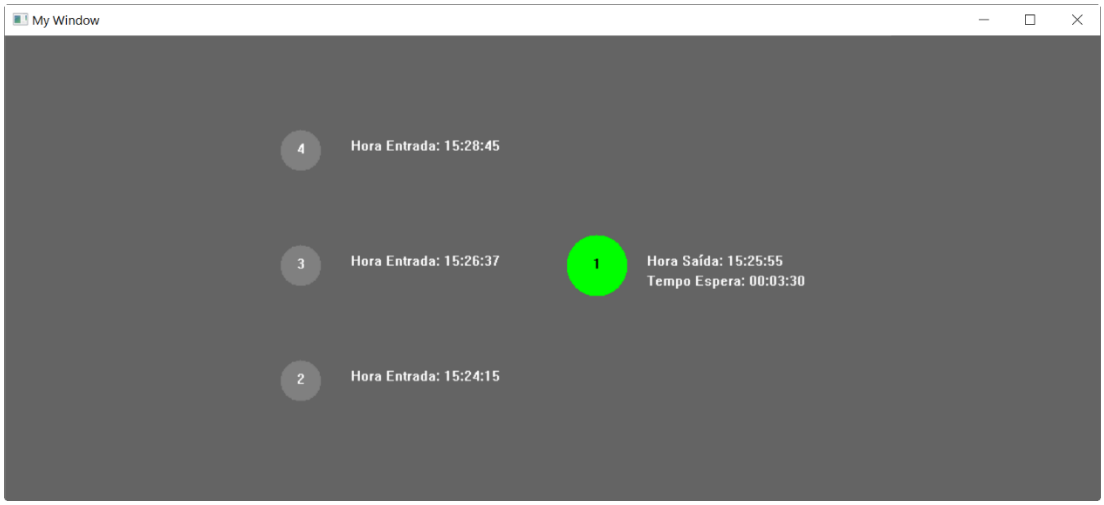
ISEL Instituto Superior de Engenharia de Lisboa	**ESTRUTURAS DE DADOS E ALGORITMOS**
Sumário: Ficha 4	Duração: 2 Semanas

Esta ficha consiste na implementação de um programa, que utilize a classe Fila que está definida nos slides da disciplina. Para isso deve implementar um programa que tenha as seguintes funcionalidades:

1. O programa deve funcionar em Windows com as classes definidas na ficha anterior e deve ter o seguintes aspeto:



2. Sempre que o utilizador clicar com o rato na janela o programa deve retirar o primeiro elemento, aquele que está há mais tempo e que está no circulo central pintado a verde e colocar o segundo como primeiro elemento indicando a hora de saída e o tempo que demorou a ser tratado.

3. Para adicionar um novo elemento à Fila o utilizador deve carregar na tecla <RETURN>. É automaticamente adicionado uma unidade ao ultimo elemento que está na Fila.

4. O programa mostra apenas os 4 primeiros elementos guardados na fila, nos círculos da janela. Se a fila tiver mais elementos guardados, estes ficam por mostrar e só aparecem à medida que se vai eliminando os primeiros elementos da fila. Enquanto não tiver 4 elementos deve mostrar os círculos vazios em valor e sem texto acessório.

Uma vez que neste programa a janela tem mais funcionalidade que a ficha anterior é necessário efetuar algumas alterações à classe Window.

Ficheiro Window.h

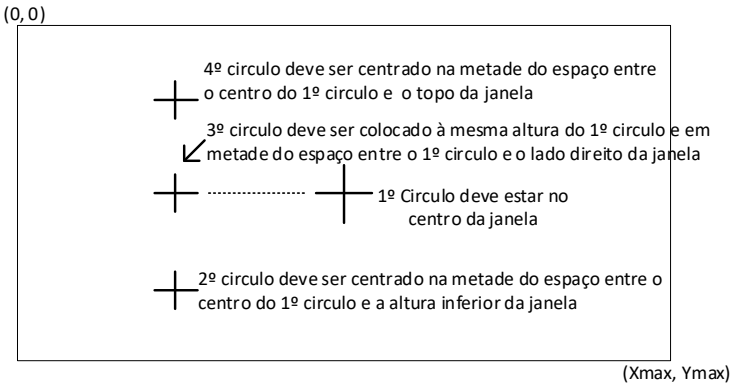
```
#pragma once
#include <Windows.h>
#include "Point.h"

class Window {
private:
	HWND windowId;
	Point cur_coord;
	bool bClick;
	bool bEnter;
	static Window* object;
	static Window* GetObject();

public:
	Window();
	~Window() { };
	bool Create(const char* sTitle);
	static LRESULT CALLBACK WindowProc (HWND wndId,
	                                    unsigned int msg,
	                                    WPARAM wp, LPARAM lp);
	void Clean();
	HWND GetWindowId()    { return windowId; }
	bool HasClicked()     { return bClick; }
	Point GetPoint()      { return cur_coord; }
	Point GetBottomRight();
	void ResetEnterKey() { GetObject()->bEnter = false; }
	bool HasPressEnter() { return bEnter; }
};
```

Ficheiro Window.cpp

```
. . .
LRESULT CALLBACK Window:: WindowProc (HWND wndId, unsigned int msg,
                                      WPARAM wp, LPARAM lp)
{
    Window* window = GetObject();
    window->bClick = false;
    switch(msg) {
        case WM_DESTROY:
            PostQuitMessage(0) ;
            return 0;
         case WM_KEYDOWN:
            if (wp == VK_ESCAPE) {
                PostQuitMessage(0) ;
                return 0;
            }
            else if (wp == VK_RETURN)
                window->bEnter = true;
             break;
        case WM_LBUTTONDOWN:
            window->bClick= true;
            window->cur_coord = Point(LOWORD(lp), HIWORD(lp));
            break;
        default:
            break;
    }
    return DefWindowProc(wndId, msg, wp, lp);
}
```

```
Point Window::GetBottomRight()
{
    Point bottom_right;
    Window* window = GetObject();
    RECT rec;
    GetClientRect(window->windowId, &rec);
    bottom_right = Point(rec.right, rec.bottom);
    return bottom_right;
}
void Window::Clean()
{
    RECT rec;
    Window* window = GetObject();
    GetClientRect(window->windowId, &rec);
    InvalidateRect(window->windowId, &rec, TRUE);
    UpdateWindow(windowId);
}
```

Os círculos devem ser colocados nos seguintes pontos:



Para saber qual é a dimensão da janela foi adicionado à classe Window uma nova função GetBottomRight() que devolve um Point com as coordenadas do canto inferior direito da janela (Xmax, Ymax).

Existe mais uma variável, bool bEnter, da classe Window que dá conta quando se carrega na tecla <RETURN>. A função de CALLBACK é alterada de forma a tratar a situação quando se carrega na tecla <RETURN>. Para apagar o conteúdo da janela antes de escrever nova informação, existe uma nova função na classe Window, `void Clean()`, que é responsável por apagar todo o conteúdo da janela gráfica.

Outra classe que é necessário introduzir, tem a ver com o texto que se quer escrever na janela gráfica, para isso definiu-se a classe StaticText, que escreve o texto na janela a partir de um dado ponto de referência e com uma dada cor, como indicado na função `Draw(HWND wndId, Point pos, long color)`.

Ficheiro StaticText.h

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

class StaticText
{
private:
	char* sText;

public:
	StaticText(const char* text);
	~StaticText();
	void Draw(HWND wndId, Point pos, long color);
};
```

Ficheiro StaticText.cpp

```
#include <string.h>
#include "StaticText.h"

StaticText::StaticText(const char* text)
{
  int len = strlen(text);
  sText = new char[len+1];
  strcpy_s(sText, len+1, text);
}
StaticText::~StaticText()
{
  delete[] sText;
}
void StaticText::Draw(HWND wndId, Point pos, long color)
{
  HDC drawDC = GetDC(wndId);
  SetBkMode(drawDC, TRANSPARENT);
  SetTextColor(drawDC, color);
  SetTextAlign(drawDC, TA_LEFT | TA_BASELINE);
  TextOut(drawDC, pos.GetX(), pos.GetY(), sText, strlen(sText));
}
```

Para poder escrever texto no centro dos círculos é necessário também efetuar algumas mudanças na classe Circle da ficha anterior:

Ficheiro Circle.h

```
class Circle {

. . .
public:
. . .
  void Draw(HWND wndId, long color); // Desenha graficamente o circulo
  void Draw(HWND wndId, long color, char* text, long text_color);
. . .
};
```

Acrescentou-se outra função Draw e que agora não só desenha o circulo com uma cor, mas também desenha o texto que se quer colocar dentro do circulo com outra cor.

Ficheiro Circle.cpp

```
void Circle::Draw(HWND wndId, long color, char* text, long text_color)
{
	if (wndId != NULL)
	{
		HDC DrawHDC = GetDC(wndId);
		// penstyle, width, color
		HPEN hNPen = CreatePen(PS_SOLID, 2, color);
		HPEN hOPen = (HPEN)SelectObject(DrawHDC, hNPen);
		HBRUSH hOldBrush;
		HBRUSH hNewBrush;
		hNewBrush = CreateSolidBrush(color);
		hOldBrush = (HBRUSH)SelectObject(DrawHDC, hNewBrush);
		Ellipse(DrawHDC, center.GetX() - radius, center.GetY() + radius,
			center.GetX() + radius, center.GetY() - radius);
		SetBkMode(DrawHDC, TRANSPARENT);
		SetTextColor(DrawHDC, text_color);
		SetTextAlign(DrawHDC, TA_CENTER | TA_BASELINE);
		TextOut(DrawHDC, center.GetX(), center.GetY() + 3, text, strlen(text));
		DeleteObject(SelectObject(DrawHDC, hOPen));
		DeleteObject(SelectObject(DrawHDC, hOldBrush));
		ReleaseDC(wndId, DrawHDC);
	}
}
```

Para a implementação do programa deve definir uma nova classe Elemento que faz parte da Fila que contenha a seguinte informação:

1. Número inteiro da ordem que corresponde à sua seleção, este número deve ser sempre incrementado sempre que se seleciona um novo elemento.
2. Hora do dia em que foi selecionado este novo elemento.

Exemplo de cálculo da hora do dia e do tempo decorrido:

```
#include <time.h>
. . .
time_t start, end;
struct tm* pGmt;
start = time(NULL);
// Execution time
end = time(NULL);
end = end - start;  // Elapsed time
pGmt = gmtime(&end);
printf("%02d:%02d:%02d\n", pGmt->tm_hour, pGmt->tm_min, pGmt->tm_sec);
. . .
```