# PRODEP

## APLICAÇÃO PARA WINDOWS

António José Esteves de Castro

7

6213(043)IEEEC1992ICAS

25/09/09

# AGRADECIMENTOS

Agradeço aos meus orientadores de estágio, Prof. Fernando Nunes Ferreira, orientador pela FEUP, e Prof. Raul Vidal, orientador pelo INESC, por todo o apoio prestado ao longo da realização do trabalho.

Pretendo também endereçar um agradecimento ao PRODEP que subsidiou esta acção.

Porto e FEUP, 2 de Novembro de 1993

António José Esteves de Castro

## Parecer

Confirmo tanto o empenho revelado no estágio pelo aluno de Engenharia Electrotécnica e de Computadores, da Faculdade de Engenharia da Universidade do Porto, **António José Esteves de Castro**, como a qualidade e o interesse científico do trabalho realizado e intitulado "Aplicação para Windows".

Considero assim o candidato merecedor da bolsa PRODEP.

Porto e FEUP, 2 de Novembro de 1993

Fernando Nunes Ferreira
Professor Catedrático da FEUP

## Parecer

Confirmo tanto o empenho revelado no estágio pelo aluno de Engenharia Electrotécnica e de Computadores, da Faculdade de Engenharia da Universidade do Porto, **António José Esteves de Castro**, como a qualidade técnica do trabalho realizado "Aplicação para Windows".

Considero assim o candidato merecedor da bolsa PRODEP.

Porto e INESC, 2 de Novembro de 1993

Raul Fernando Almeida Moreira Vidal
Professor Associado da FEUP

# ÍNDICE

# Introdução

Este relatório descreve uma aplicação em WINDOWS feita em linguagem C++ para WINDOWS. É feita uma breve descrição da linguagem, do sistema operativo, bem como da biblioteca ObjectWindows, largamente utilizada no programa.

A aplicação destina-se a ler dados de uma placa já existente e que, básicamente, faz a demodulação e a conversão analógico-digital de um sinal de tipo FAX,modulado em FM, contendo imagens metereológicas e que pode ser captado por aparelhagem de Rádio Amador.

A aplicação encarrega-se de adquirir o sinal digital vindo da placa e mostrá-lo no écran. Sempre que a placa é activada, interrompe o programa sempre que tiver no BUS um valor válido. Deste modo, a taxa de amostragem é independente da velocidade do programa, o que aconteceria caso fosse o programa a, periodicamente, ir ler o conteúdo do BUS.

A frequência a que a placa amostra o sinal pode ser alterada pelo utilizador

Além da aquisição, a aplicação pode, a pedido do utilizador, mostrar uma imagem já previamente adquirida e gravada em ficheiros com a extensão .MAP.(por nós criado e que consiste na totalidade dos bytes da imagem antecedidos do nº de pixeis por linha ) Pode também gravar a imagem presentemente no écran no mesmo formato (

Estas imagens, uma vez no écran, podem ser processadas de variadas formas descritas na secção seguinte.

# Manual do Programa

O programa é iniciado com a imagem seguinte


O programa apresenta um menu com diversas opções:

Ficheiro
Operações
Parâmetros
Aquisição
Stop
Help

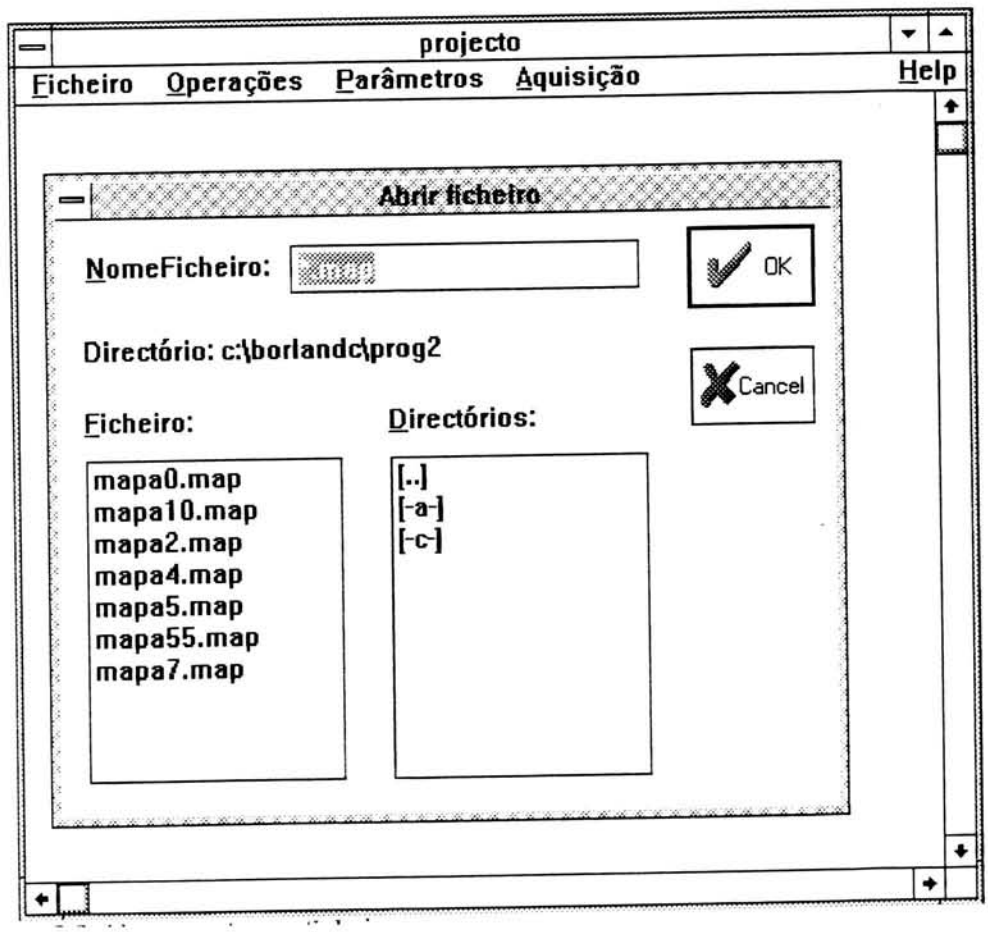
A opção Ficheiro possui vários items:

Abrir - serve para ler um ficheiro no formato map e desenhá-lo no écran;

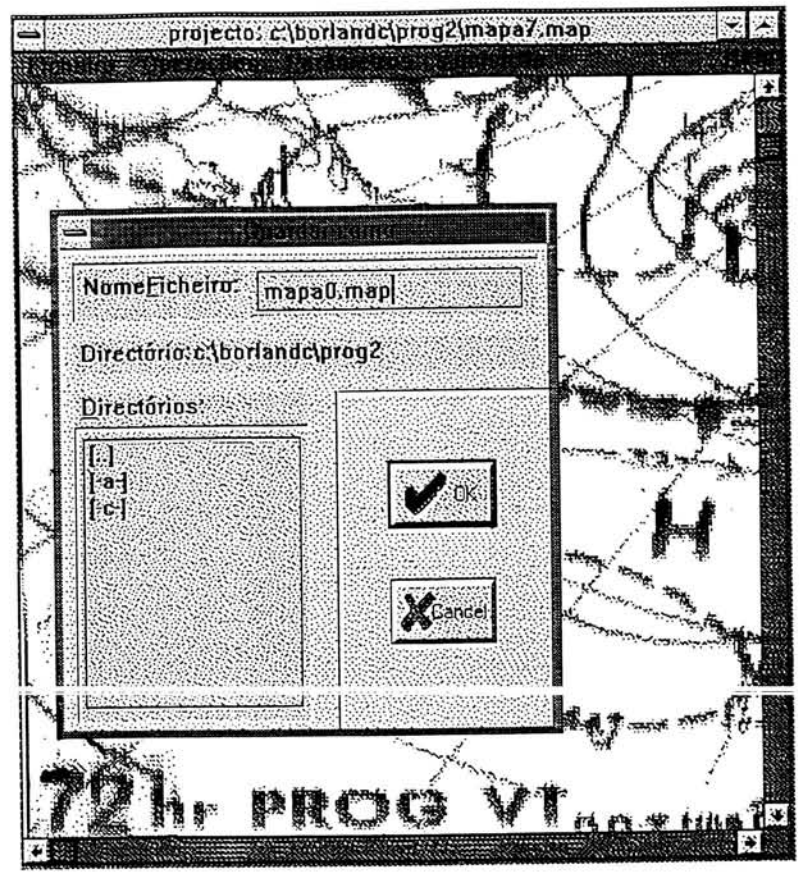
Guardar - serve para guardar no formato map a imagem presentemente no écran

Sair - termina a aplicação

A figura seguinte mostra o ambiente do programa quando se vai abrir um ficheiro




A figura seguinte mostra a janela que é utilizada na gravação de uma imagem




A opção Operações possui outros items:

Negativo - desenha o negativo da imagem no écran

Selecciona - Coloca no écran apenas a parte da imagem dentro do rectângulo previamente desenhado com o rato : caso o rectângulo não tenha sido desenhado, este item é inactivado

Copiar - copia para o ClipBoard a imagem dentro do rectângulo previamente desenhado com o rato

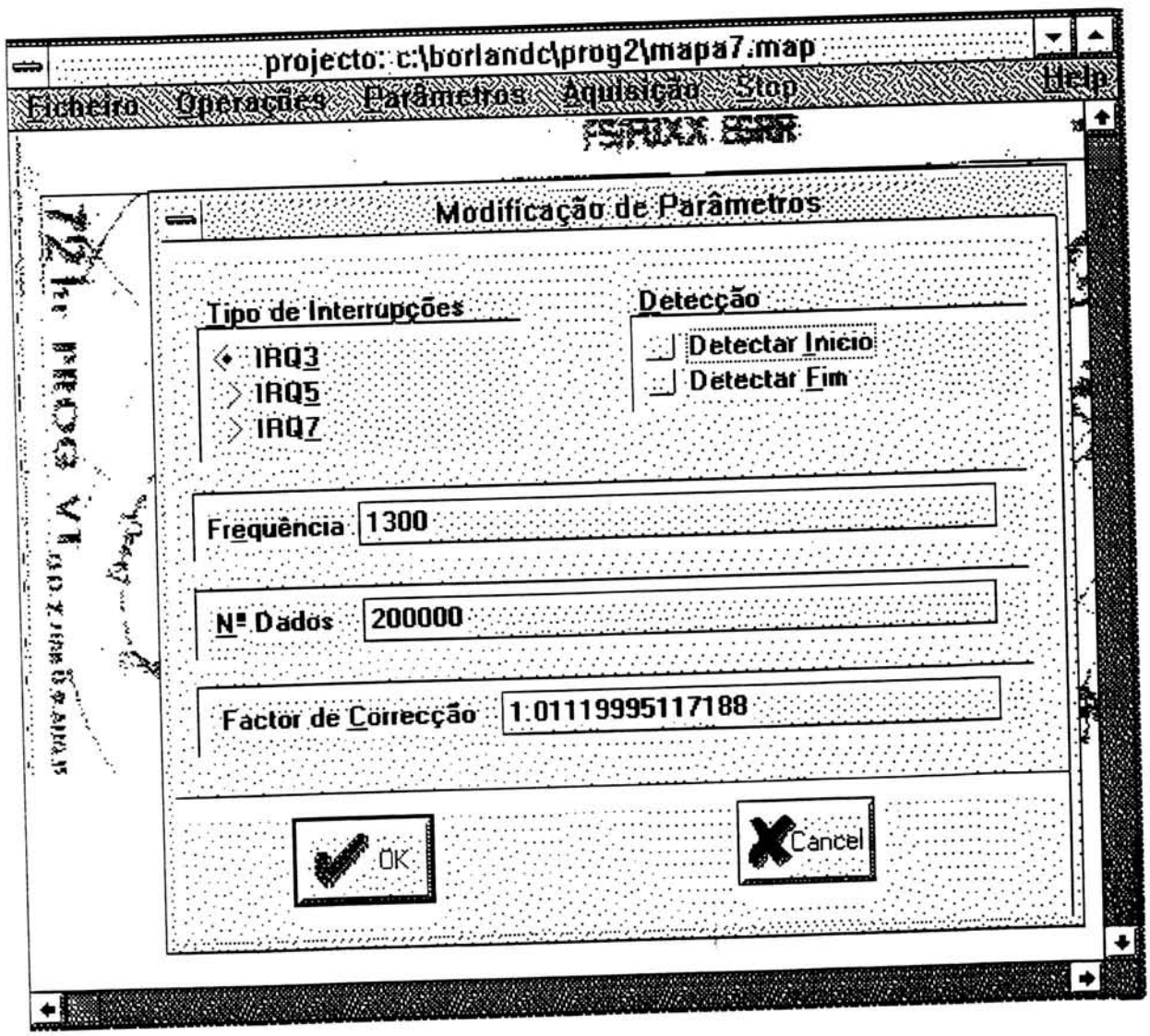
Zoom dobro

metade - fazem o zoom para metade ou o dobro da imagem dentro do rectângulo previamente desenhado com o rato

Espelho vertical
horizontal - faz o espelho vertical ou horizontal da imagem dentro do rectângulo previamente desenhado com o rato

Rodar esquerda
direita - faz a rodagem da imagem dentro do rectângulo previamente desenhado com o rato

A opção Parâmetros faz aparecer um diálogo onde se podem mudar a frequência de amostragem, o tipo de interrupção usada, o nº de dados a adquirir e um factor de correcção da imagem a adquirir da placa, como se pode ver na figura seguinte.



A opção Aquisição permite adquirir a imagem através da placa, de acordo com os parâmetros definidos pela opção anterior

A opção Stop permite terminar a aquisição dos dados mesmo que se não tenha ainda atingido o nº de dados especificados.

Finalmente a opção Help possui 2 items:

Acerca
Ajuda.

# Linguagem C++

C++ é uma melhoria do C tradicional, acrescentando-lhe o conceito de classe, a seguir explicado.

## Classes

Uma classe é uma extensão da ideia de estrutura usada no C tradicional. Através das classes é possivel implementar funções associadas e operadores e é também o mecanismo que permite representar por exemplo, dados tipo números complexos e *stacks* .

Uma estrutura permite ao programador agregar vários componentes numa ùnica variável. Os componentes da estrutura chamam-se membros e são acedidos dentro dela individualmente. Um aspecto que é crítico neste tipo de dados é o acesso aos membros da estrurura que é feito através do operador "." ou do apontador para a estrutura "->". Estes operadores conjuntamente com ( ) e [ ] tem maior prioridade.

O conceito de structure é alargado em C++ para permitir que funções sejam também membros numa estrutura. A declaração da função está incluída na declaração de estrutura e é chamada utilizando os diversos meios atrás referidos.

As funções membro são definidas dentro da estrutura e deverão ser inline, isto é, com uma só linha como por exemplo:

```
inline SQ( int x) { return (x*x);}
```

Para definir uma função membro fora da estrutura, deverá usar-se o operador *Scope Resolution* "::". Este operador permite que funções membro de diferentes estruturas tenham o mesmo nome.Além disso em C++ a estrutura que os membros sejam do tipo public e private. Dentro de uma estrutura o uso da palavra chave "private:", restringe o alcance dos membros que se lhe seguem. Os membros privados só podem ser usados em algumas categorias, cujos privilégios incluam o acesso a esses

membros. Os membros publicos podem ser acedidos por qualquer função.
Classes são um tipo especial de estrutura cujos membros são por defeito privados. As classes podem ser intercaladas e também derivadas de outras.

### Classes derivadas

*Inheritance* é o mecanismo de derivar uma nova classe a partir de uma já existente, isto é, a classe derivada pode adicionar membros ou alterar a classe existente.
Uma classe é derivada da seguinte forma :

```
class  nome-da-classe : (
publica/privada/protegida)opcionalnome-da-classe-
base

                    {
                        declarações dos membros;
                    };
```

A classe derivada tem os seus próprios construtores (falaremos deles mais adiante) que vão invocar os construtores da classe base. A sintaxe usada para passar argumentos da classe derivada para a classe base é a seguinte:

```
        function header : nome-da -classe-
baseopcional (lista de argumentos);
```

Uma classe derivada publicamente constitui um subtipo da classe base. Uma variável da classe base pode ser tratada como se pertencessse a uma classe base. Um operador para uma classe base pode apontar para objectos de uma classe derivada publicamente.
A palavra chave classe virtual é uma especificaçao da função que fornece um mecanismo para que o membro da função apropriado seja dinâmicamente seleccionado durante a execução de programa.

### Construtores e Destruidores

Algumas das classes usadas podem ter construtores e destruidores.
Um construtor é uma função membro cujo o nome é igual ao nome da classe e que constroi um objecto dessa classe tipo. Este processo pode envolver a inicialização de membros e a alocação de memória para esse objecto, usando para tal a instrução "new".
Um destruidor é uma função membro cujo nome é o nome da função precedido do caracter "~". É usado para destruir uma classe tipicamente usando a função "delete".

Apresenta-se de seguida um exemplo de uma classe com construtor e destruidor:

```
class stack {
     char *s;
     int max_comp;
     int top;
public :
     stack (int size) {(s= new
char[size];
                    max_comp= size;
                    top= 0;
                   };
      ~stack(void) { delete s; }
     ........
     };
```

**Breve referência a programação orientada por objectos**

A caracteristica principal da programação orientada por objectos (OOP) consiste no facto de um conjunto apropriado de tipos de dados e suas operações estarem encadeadas. A classe, com as suas funções membro e o operador de *overloading* fornecem as ferramentas de codificação apropriada. As classes permitem também a implementação de dados do tipo abstrato, tais como por exemplo n·meros complexos.

A segunda caracteristica é a possibilidade do codigo poder ser reutilizado através de um mecanismo de hereditariedade. Sem esse mecanismo, uma variação mesmo que pequena nos dados de tipo abstratos implicaria uma revisão do código. Este mecanismo permite ao programador construir um sistema hierárquico adequado para a resolução do seu problema.

A terceira caracteristica é a formação dinâmica de Types. A função virtual fornece a possibilidade da criação de tipos no decorrer da execução do programa mas é o objecto que determina como tal será feito. Por exemplo, considerando *shape* como uma classe base e *area* e *draw* como as funções virtuais dessa classe. Partindo do principio que existe um conjunto de classes derivadas tais como *rectangle*, *circle*, *pentagon* e *triangle*, cada uma delas sabe por si só desenhar-se ou computar a sua própria área. Um apontador para a classe *shape* pode ser usado para processar dinâmicamente objectos de qualquer um dos subtipos de *shape*. O codigo para desenhar formatos é mantido com facilidade porque é muito modular e novos formatos podem ser acrescentados com facilidade devido ao mecanismo de hereditariedade.

A quarta caracteristica é a possibilidade de tornar o acesso a dados condicionado. Através de um sistema de acesso onde existem previlégios consegue-se gerir qual o grupo de funções que podem aceder aos dados, que serão normalmente apenas aquelas que realmente necessitam deles

para realizar as suas implementações. Os modificadores de visibilidade, public , protected e private controlam o acesso aos objectos. Através deste controlo consegue-se um sistema modular e mais robusto.

A programação por objectos consegue soluções que devido à estrutura utilizada de encapsulamento obtém-se programas mais robustos, maior facilidade na introdução de alterações e também mais reutilizáveis. Por exemplo quando um codigo necessita de uma *stack* , pode utilizar com facilidade a *stack* de outro código já existente. Em liguagens tradicionais tal não é possível dado que esse tipo de estrutura de dados não pode ser exportada porque é caracteristica do respectivo algoritmo.

Este tipo de vantagens são muito importantes especialmente em projectos de grande dimensão, em termos de código, e que exijam normalmente uma grande coordenação entre muitos programadores.

## Sistema Operativo Windows

Ao contrário do que acontece na maioria das aplicações em DOS, que são time driven (isto é, sequenciais: as funções são chamadas umas atrás das outras de acordo com a sequencia em que foram postas pelo programador) , as aplicações em WINDOWS são event driven, isto é, a sequência por que são chamadas as diversas funções depende dos eventos ocorridos, que , neste caso, são mensagens enviadas quer pelo sistema operativo quer pelo utilizador, por exemplo, quando selecciona um item num menu. Assim, por exemplo, sempre que o utilizador mexe o rato, o sistema operativo envia uma mensagem WM_MOUSEMOVE e quando carrega no botão esquerdo do rato, envia uma mensagem WM_LBUTTONDOWN. De acordo com estas mensagens, assim deve agir o programa, chamando a função respectiva (por exemplo, um programa que desenhe pontos no ecran na posição onde estiver o rato, deve ser chamada sempre que houver uma mensagem WM_MOUSEMOVE, colocando um pixel na posição do rato). Neste sistema operativo, as mensagens estão sistematicamente a interromper os processos , sendo colocadas numa fila de espera que o programa vai ler quando não estiver ocupado com uma função específica (por exemplo, enquanto estiver num ciclo não lê as mensagens, excepto se for chamada uma função específica para tal :em C++ é *GetMessage()* ou *PeekMessage()* ).Este sistema operativo permite que, muitas vezes, o programa "não faça nada ", quando está a espera de uma acção da parte do utilizador.

Para lidar com a maior complexidade deste sistema eram necessários programas extremamente complicados, que, unicamente para abrir uma janela exigiam uma grande quantidade de código e, para

responder as mensagens ter-se ia que usar inúmeras condições *switch-case* com um *case* para cada uma das mensagens recebidas, o que se poderia tornar complexo se houvesse várias condições a cumprir e se se usassem muitas mensagens.

# Estrutura de um programa em Windows

## 1- Estrutura do Windows

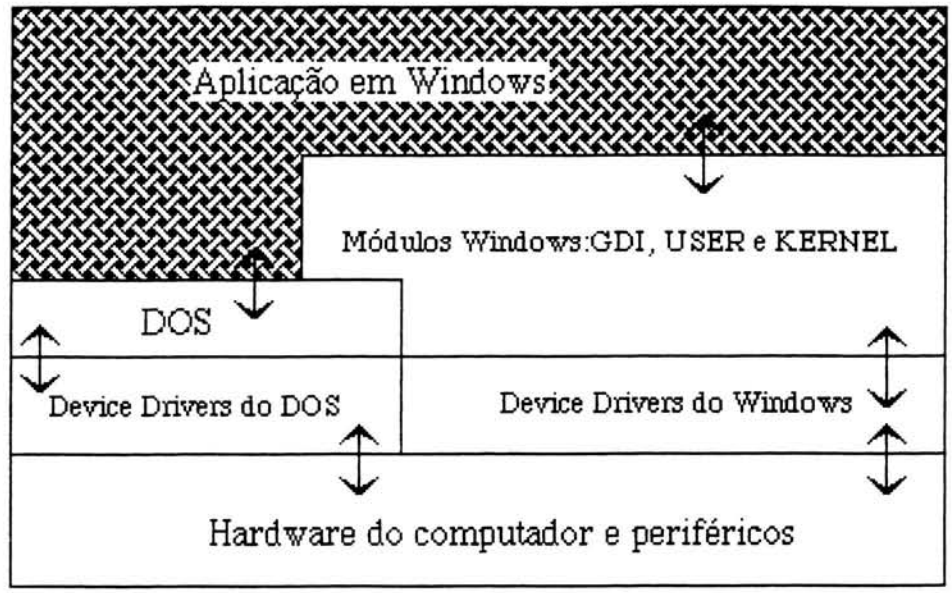
A funcionalidade do Windows e das suas API's (*Windows Application Programming*) reside em 3 bibliotecas externas chamadas pelas aplicações durante a respectiva execução e que são :

-KERNEL.EXE
Responsavél pela gestão da memória e recursos, pelo escalonamento e interacção com o DOS.
-GDI.EXE
Responsavel pela colocação no ecrâm e na impressora dos gráficos.
-USER.EXE
Responsável pela gestão das janelas e pela comunicação com o utilizador.

## 2-Interacção com o DOS e Windows

As aplicações em DOS não tem em consideração a contribuição do sistema operactivo para o sucesso da aplicação, apesar dos programas em DOS serem executados devido à interacção entre o código da aplicação e as facilidades concedidas pelo sistema operativo ( S.O.).

O mesmo não se pode dizer para qualquer programa em Windows, devido ao facto deste S.O. oferecer muito mais funcões do sistema é mais dificil menosprezar as interacções entre o S.O. e a aplicação. O programa é obrigado a interagir continuamente com o S.O. DOS+Windows. A figura seguinte ilustra como uma aplicação windows interage com o S.O. Windows e DOS .




### 3. *Application startup responsibilities*

No inicio da execução de uma aplicação o Windows fornece 4 parâmetros a essa aplicação. Num programa em ObjectWindows esses parâmetros são transferidos para o construtor do objecto da nossa aplicação, que deriva da classe TApplication. Esses parâmetros são também armazenados nos data members do objecto da aplicação de modo a poderem ser convenientemente acedidos pelo nosso código.

Esses *data members* são descritos de seguida :

*-hInstance*

Guarda o valor da instância da aplicação.

*-hPrevInstance*

Guarda o valor da instância anterior da mesma aplicação. Tem o valor zero se essa for a primeira instância de execução.

*-lpCmdLine*

Guarda uma string (LPSTR) representando a linha de comando do startup da aplicação, incluindo a opção nome_do_ficheiro. Por exemplo, "CALC.EXE/M" ou "WORDPROC.EXE LETTER1.DOC".

*-nCmdShow*

Guarda um inteiro que representa o valor inicial do modo do display para a janela principal. É usado nas chamadas às funções membro da classe show.

A primeira coisa que uma aplicação em ObjectWindows deve fazer é construir um objecto da aplicação a partir da classe TApplication. No exemplo é usado o THelloApp.

A partir da primeira instância de uma aplicação em ObjectWindows, a instrução Run chama InitApplication para realizar as inicializações necessárias para a aplicação. Run inicializa todas as instâncias ( incluindo a primeira ) ao chamar a função InitInstance. InitInstance chama de seguida a função membro InitMainWindow para construir e inicializar o objecto janela principal. Finalmente o Run retorna da aplicação apenas quando o programa terminar. O status correspondente ao terminus da aplicação fica armazenado num data member status, que deve retornar para o Windows tal como acontece com a função WinMain.

### 4. **As responsabilidades da janela principal**

A janela principal de uma aplicação é a janela que aparece em primeiro lugar quando a aplicação é iniciada. Nesta janela está representado uma lista de comandos que o utilizador pode usar ( o menu ). Durante o decorrer da aplicação a janela principal faz também a gestão da interface da aplicação e em muitos casos serve como a única área de trabalho dos programas.Outras aplicações mais complicadas podem ter muitas janelas a servir de área de trabalho.Finalmente, quando o utilizador fecha a janela principal, esta é responsável pela inicializacão do processo que vai fechar a aplicação.

## ObjectWindows

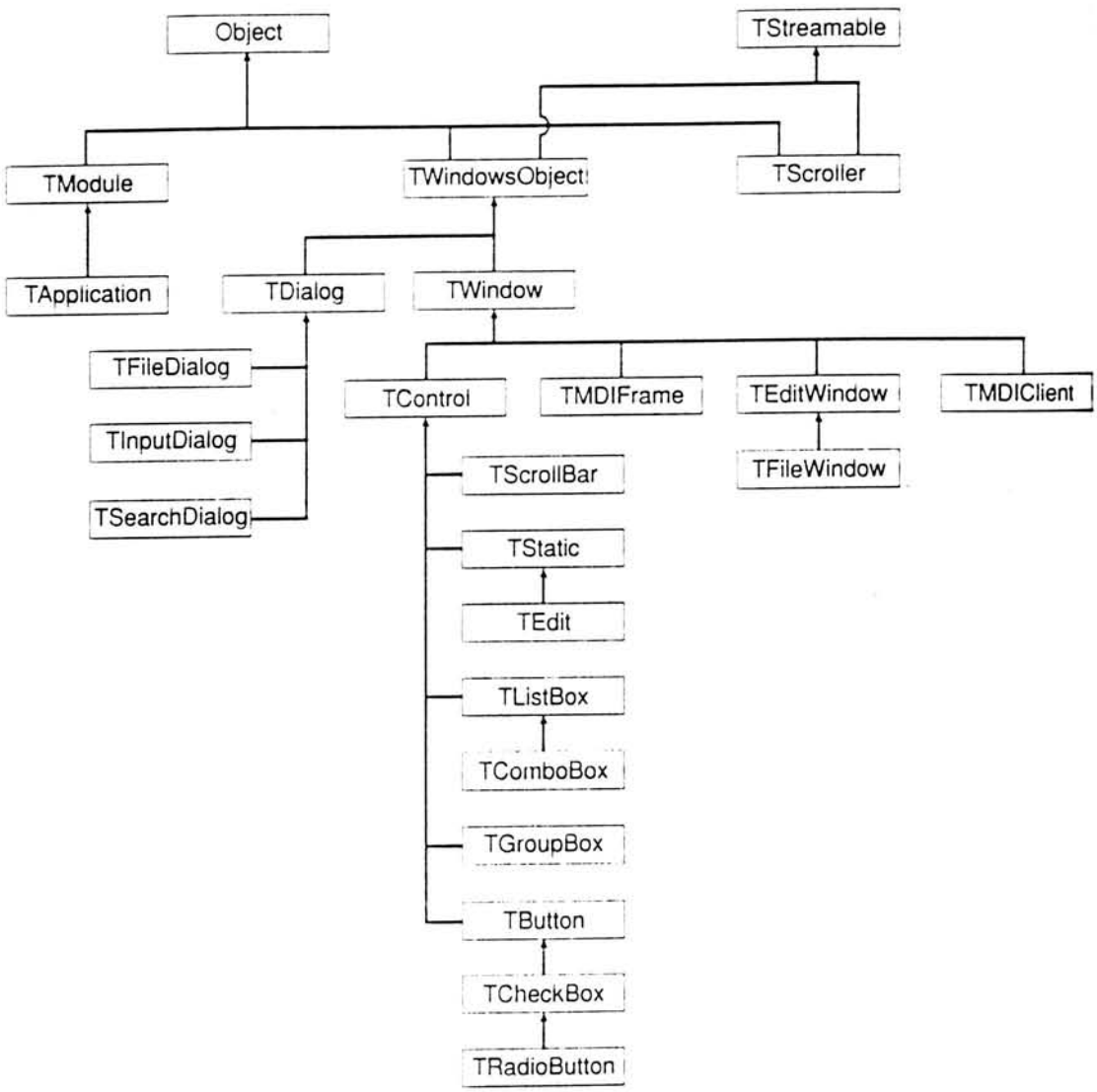
Para simplificar ligeiramente os programas em C++ para Windows foi criada uma Biblioteca ObjectWindows em <owl.h>, incluída no pacote C++ para WINDOWS, que permite lidar com problemas como a criação de janelas ou de diálogos e menus de forma mais simples . No entanto, é mesmo assim complicado criar um programa que simplesmente crie uma janela , como é o caso do programa seguinte:

```
    #include<owl.h>

// define-se a classe TOlaApp como derivada da classe pré
definida TApplication

class TOlaApp : public TApplication {
public:
    TOlaApp(LPSTR AName, HINSTANCE hInstance,
            HINSTANCE hPrevInstance, LPSTR lpCmd,
            int nCmdShow)
          : TApplication(AName, hInstance,
                         hPrevInstance, lpCmd, nCmdShow) {};
    virtual void InitMainWindow();
};
```

```
/*****inicializa-se a janela, como sendo TWindow, outra
classe pré definida e que se encarrega de gerir tudo o que
diz respeito a uma janela */

 void TOlaApp::InitMainWindow()
{
  MainWindow = new TWindow(NULL, "OLA");
}


/*****programa principal: em ObjectWindows para C++ usa-se
sempre este programa principal*********/

int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance,
             LPSTR lpCmd, int nCmdShow)
{
  TOlalApp OlaApp("OlaApp", hInstance, hPrevInstance,
          lpCmd, nCmdShow);
  OlaApp.Run();
  return OlaApp.Status;
}
```

A biblioteca OWL possui pré definidas diversas classes, na qual a principal é a classe Object, da qual todas as outras derivam, conforme ilustrado na figura da página seguinte

Destas, as mais usadas são TApplication, que serve para criar as aplicações em WINDOWS, para criação da janela principal, além de se encarregar do processamento das mensagens e TWindow, que se encarrega de gerir a janela chamada por TApplication.




## Descrição do programa[1]

Uma vez que no nosso programa só usamos uma janela, este apenas possui uma única classe derivada de TWindow, que é a classe Janela, representativa da janela principal, que é declarada do seguinte modo:

```
_CLASSDEF(Janela)
class Janela : public TWindow
{
public:

        BOOL ButtonDown;
        TPParametro Parametro;
```

[1] A listagem completa do programa encontra-se em anexo

```
        // funções já definidas em TWindow
      Janela( PTWindowsObject, LPSTR );
      virtual ~Janela();
      virtual LPSTR GetClassName();
      virtual void Paint( HDC, PAINTSTRUCT _FAR & );
      virtual void GetWindowClass( WNDCLASS& WndClass );

         // declaração de todos os items dos menus
      virtual void CMExit( RTMessage ) = [CM_FIRST +
CM_EXIT];
      virtual void CMUFileOpen(RTMessage)=
[CM_FIRST+CM_U_FILEOPEN];
      virtual void CMUHelpAbout( RTMessage ) =
[CM_FIRST+CM_U_HELPABOUT];
      virtual void CMGuarda( RTMessage ) =
[CM_FIRST+CM_U_GUARDA];
      virtual void CMAjuda(RTMessage)=[CM_FIRST+CM_AJUDA];


      virtual void CMSel(RTMessage
)=[CM_FIRST+CM_SELECCIONA];
      virtual void CMNega(RTMessage )=[CM_FIRST+CM_NEGA];
      virtual void CMMetade(RTMessage )=[CM_FIRST+CM_METADE];
      virtual void CMDobro(RTMessage )=[CM_FIRST+CM_DOBRO];
      virtual void CMEsphor(RTMessage )=[CM_FIRST+CM_ESPHOR];
      virtual void CMEspver(RTMessage )=[CM_FIRST+CM_ESPVER];
      virtual void
CMRodadir(RTMessage)=[CM_FIRST+CM_RODADIR];
      virtual void
CMRodaesq(RTMessage)=[CM_FIRST+CM_RODAESQ];
      virtual void CMParametro(RTMessage)=
[CM_FIRST+CM_PARAMETRO];
      virtual void CMaquisicao(RTMessage)=
[CM_FIRST+CM_AQUISICAO];
      virtual void CMpara(RTMessage )=[CM_FIRST+CM_PARA];
      virtual void CMCopia(RTMessage )=[CM_FIRST+CM_copia]



      virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];

      virtual void WMLButtonDown(RTMessage Msg)
     = [WM_FIRST + WM_LBUTTONDOWN];
      virtual void WMLButtonUp(RTMessage Msg)
     = [WM_FIRST + WM_LBUTTONUP];
       virtual void WMMouseMove(RTMessage Msg)
     = [WM_FIRST + WM_MOUSEMOVE];


      int X1, Y1, X2, Y2, larg,alt,posx,posy;
      void AdjustScroller();
       void Desactiva();
       void ActivaMenu();
         void verifica();
      char szName[256];
           HDC HandleDC;
      HPEN ACaneta;
```

```
    HCURSOR Cur;
    LPLOGPALETTE MyLogPalette;


};
```

Esta classe Janela possui todas as variáveis e funções usadas, como por exemplo as funções chamadas quando o utilizador escolhe um item do menu associado a janela. Por exemplo, como se pode ver, uma função associada a esta classe Janela é assim declarada :

```
    virtual void CMSel(RTMessage
)=[CM_FIRST+CM_SELECCIONA];
```

em que, entre parêntesis rectos está a mensagem a qual esta função deve responder, que neste caso é a mensagem enviada pelo sistema operativo sempre que se selecciona o item Seleccionar, da opção Operações.

Esta função, que será a seguir explicitada, envolve também a manipulação de *Bitmaps* , que são porções reservadas de memória onde se guardam os bytes que indicam a cor de cada pixel( o nº de bytes por ponto depende do nº de cores usadas);Estes *Bitmaps* podem ser criados usando a seguinte função:

```
    CreateCompatibleBitmap(DC, larg,alt);
```

Os *Bitmaps* podem ser transferidos quer para outros Bitmaps quer para o écran

Apresenta-se agora a funçÃo atrás mencionada:

```
void Janela::CMSel(RTMessage Msg)
{

 // coloca no bitmap principal apenas o que foi seleccionado
com o rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;

     // adquire o contexto da janela activa
   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

     // coloca o bitmap a branco
   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);
```

```
     /* trasfere para nbitmap a parte de MeuBitMap
seleccionada, entre (X1,Y1) e (X1+larg,Y1+alt)  */
    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


     PatBlt(MemDC, 0, 0,
BitMapWidth,BitMapHeight,WHITENESS);

   // transfere a imagem em nbitmap para MeuBitMap
    BitBlt(MemDC, 0, 0,BitMapWidth, BitMapHeight, Mem2DC, 0,
0,SRCCOPY);
     // transfere a imagem de MeuBitMap para o écran
    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

    X1= Y1= X2= Y2= larg=alt=0;


}
```

Além desta função *CMSel()* ,todas as outras funções são definidas no programa, como o construtor e o destruidor (Janela::Janela e Janela::~Janela) que são chamadas, respectivamente quando se cria e quando se destrói a janela. É no construtor que é feita a inicialização de variáveis e funções. É também aqui que é feita a inicialização da *Palette* usada: essa *palette* consiste num nº de tons de cinzento dado por *nColors*

A classe TWindow tem já pré-definidas as funções *GetClassName()*, *GetWindowClass()* e *Paint()*. Todas elas são chamadas directamente pelo Sistema Operativo: por exemplo, *Paint()* é chamado sempre que o S.O. detecta alterações na janela activa, redesenhando-a; ésta função é por nós redefinida do seguinte modo:

```
void Janela::Paint( HDC HandleDC, PAINTSTRUCT & PaintInfo )
{


     HDC MemDC;

          if (larg>1) ActivaMenu();
          else Desactiva();

      MemDC = CreateCompatibleDC(PaintInfo.hdc);
      SelectObject(MemDC, MeuBitMap);
      BitBlt(PaintInfo.hdc,0,0,BitMapWidth,
BitMapHeight,MemDC, 0, 0, modo);
      DeleteDC(MemDC);

}
```

Como se vê, a função limita-se a colocar o *Bitmap* no écran

Outra das classes OWL usadas no programa é a classe TDialog, que é usada para gerir todas as janelas de diálogo usadas. No nosso programa temos uma classe, TParametro , derivada de TDialog, a seguir declarada:

```
class TParametro : public TDialog {
public:
  TParametro(PTWindowsObject AParent, int ResourceId);
};
```

O respectivo constructor é

```
TParametro::TParametro(PTWindowsObject AParent,int
ResourceId)              : TDialog(AParent, ResourceId)
{
  new TBCheckBox(this, ID_INICIO, NULL);
  new TBCheckBox(this, ID_FIM, NULL);

  new TBRadioButton(this,ID_INTER3,NULL);
  new TBRadioButton(this,ID_INTER5,NULL);
  new TBRadioButton(this,ID_INTER7,NULL);

  new TEdit(this, ID_FREQ,
         sizeof(((Janela *)Parent)->Parametro.frequencia));
  new TEdit(this, ID_NDADOS,
         sizeof(((Janela *)Parent)->Parametro.n_dados));
  new TEdit(this, ID_FACT,
         sizeof(((Janela *)Parent)->Parametro.factor));

  TransferBuffer = (void far*)&(((Janela *)Parent)-
>Parametro);
}
```

No construtor de TParametro são inicializadas outras classes pré-definidas, através, por exemplo, da expressão:

```
          new TEdit(this, ID_FREQ,sizeof(((Dialogo *)
Parent)->Parametro.frequencia));
```

que declara que este diálogo possui uma janela onde o utilizador pode editar valores, modificando-os (neste caso, pode modificar a variável Parametro.frequencia). A classe TEdit, como se pode ver na figura 2, é derivada de TControl. Juntamente com esta são declaradas outras classes de controle: TBCheckBox e TBRadioButton, para o utilizador poder escolher, respectivamente, o tipo de detecção e o tip°o de interrupções a usar na aquisição da imagem( neste último caso é necessária mútua exclusão entre as opções, razão pela qual se usa um *Radio Button* )

A expressão *new* é usada para iniciar qualquer tipo de classe (no programa mostrado inicialmente é usada para iniciar a classe TWindow, de modo a aparecer a janela).

Esta classe TParametro controla uma janela de diálogo onde o utilizador pode modificar algumas variáveis usadas no programa. Para chamar este diálogo usa-se a seguinte função, chamada quando o utilizador escolhe a opção Parâmetros do menu:

```
void Janela::CMParametro( RTMessage Msg )
{


 char ALabel[255];

    //converte os valores para string

    ltoa(n_dados,Parametro.n_dados,10);
    itoa(frequencia,Parametro.frequencia,10);
    gcvt(paramet,15,Parametro.factor);


   // chama o diálogo TParametro, definido em dial.rc

  if ( GetModule()->ExecDialog(new TParametro(this,
ID_DIALOGO)) == IDOK )
  {

   frequencia= atof( Parametro.frequencia);
   n_dados= atol( Parametro.n_dados);
   paramet=atof(Parametro.factor);
   if ( Parametro.INTER3 ) INTR=11;
    else if ( Parametro.INTER5) INTR=13;
    else INTR=15;
}
}
```

Existem, outras classes ,derivadas de TDialog , usadas para aplicações mais especificas, como por exemplo TFileDialog usada quando os diálogos são para fazer uma listagem de ficheiros e que no nosso programa é usado para os diálogos Abrir e Fechar, da opção Ficheiro do menu

Finalmente uma outra classe OWL usada é a classe TScroller, usada para colocar e gerir tudo o que se relaciona com as 2 *Scrollbars* usadas

Finalmente descreveremos uma das mais importantes funçoes por nós usada, que é a que se encarrega de adquirir dados a partir da placa.

Como já anteriormente descrito, o sistema de aquisição funciona por interrupções, isto é, sempre que a placa tem dados válidos no BUS, interrompe o programa. Estas interrupções só são activadas quando o

utilizador escolher a opção Aquisição do menu, que também indica à placa qual a fruquência a que deve amostrar o sinal. Uma vez activadas as interrupções, a rotina chamada quando o CPU recebe a interrupção coloca o valor recebido numa posição de memória. A função *CMaquisição()* vai entretanto lendo os valores escritos na memória desenhando no écran e num *Bitmap* o pixel com a cor correspondente ao valor que veio da placa. Uma vez que a placa envia dados a uma cadência mais alta do que a de leitura dos dados, o nº de dados a adquirir é limitado pela memória disponível no computador. Em seguida apresentam-se extratos dessa função:

```
     void Janela::CMaquisicao( RTMessage )
{
 .
 .
 .
    p=(byte huge *) farmalloc(n_dados+3000);
    if (p==NULL) MessageBox(HWindow,"\n não há memória
suficiente  \n" , "Mensagem", MB_OK);

     n_dados_gravados=0;
     g=0;

   // Seleccao da frequencia de amostragem (frequencia
minima 160 Hz)

       if (frequencia>160) valor=(long)
(7600000/frequencia);

       outportb(771,255); outportb(772,0x36);  // control
word para o C0
       // com dois bytes a serem escritos no contador

       // primeiro manda-se o byte menos significativo
       outportb(771,63);  outportb(772,(byte)(valor % 256));

       // em seguida o byte mais significativo
       outportb(771,63);  outportb(772,(byte)(valor / 256));


  // INICIALIZA A LEITURA POR INTERRUPÇÕES

   /* save the old interrupt vector */
   oldhandler = getvect(INTR);
   /* install the new interrupt handler */
   setvect(INTR, handler);
   //activação de interrupções do IRQ3 do 8259
   outportb(0x21,0);

   // Escolha do canal de entrada do AD
   outportb(769,0);

   .
   .
   .
```

```
    // enable de interrupções
   outportb(771,12);

    ///**********************


  //INICIO DA LEITURA E DESENHO DA IMAGEM NO ECRAN
   while(g<n_dados)
    {

  fp=fopen(ficheiro_dados,"wb");
  n_dados_gravados=0;

  med=(long) frequencia/paramet;

   .
   .
   .

   inicio_imagem=h-k;
 // h fica a apontar para o "desvio" do inicio da imagem
relativamente a p

   h=inicio_imagem-50;
   coluna=linha=1;


    while (h<(long)n_dados)
      { if (coluna>=(int) (med/salta)) { coluna=0; linha++;
}
     coluna++;
     int cor= (byte)  NumColors* (*(p+h)) / 255 ;
     SetPixel(TheDC,coluna,linha,PALETTEINDEX(cor));
     SetPixel(MemDC,coluna,linha,PALETTEINDEX(cor));
```

/* a função *verifica()* destina-se a verificar se o Sistema Operativo enviou alguma mensagem e, em caso afirmativo, executar as funções inerentes a essa mensagem. O sinal continuará a ser adquirido, mesmo que se mude de aplicação. Se esta função não existisse, não se sairia da mesma janela enquanto não terminasse a aquisição de todos os dados. Graças a esta função é possivel terminar a aquisição, chamando a função *CMpara()* que activa a *flag* para */

```
     verifica();
```

/* se para foi activada, termina a aquisição, desactivando as interrupções*/

```
     if (para) { n_dados=g;
               n_dados=h;
               /* reset the interrupt handler */
                outportb(771,255);
                setvect(INTR, oldhandler);
}
   .
   .
   .
```

```
}
```

O nosso programa está distribuído por diversos ficheiros, que são depois compilados e ligados pelo compilador. Esses ficheiros são:

Prodep.cpp
Dial.rc
bwcc.lib
owl.def

O primeiro é onde está o programa escrito em C++ que foi anteriormente descrito

Os 2 últimos ficheiros estão incluídos no pacote C++ :*bwcc.lib* , que está no directório borlandc\lib serve apenas para que o ficheiro executável funcione quando o compilador não está activo pois o programa usa recursos próprios do C++ que não seriam compreensíveis pelo sistema operativo caso não se inclua esta biblioteca; *owl.def* , que está no directório borlandc\owl\include define algumas variáveis do sistema, como por exemplo o tamanho da *stack* a usar.

O segundo é uma *resource file* onde estão definidos, em linguagem própria, o menu, os diversos diálogos e os ícones utilizados. Estes recursos podem ser facilmente visualizados e alterados usando o programa *Resourse Workshop* incluído no pacote C++; todos estes recursos têm de ser declarados e chamados a partir do ficheiro Prodep.cpp ; por exemplo, na definição do menu, faz-se:

```
CMDLGAPMENU MENU
BEGIN
  .
  .
  .
     POPUP "&Operações"
     BEGIN
          MENUITEM "&Negativo", CM_NEGA
          MENUITEM SEPARATOR
          MENUITEM "S&eleccionar", CM_SELECCIONA, GRAYED
          MENUITEM "&Copiar", CM_COPIAR, GRAYED
    .
    .
    .
END
```

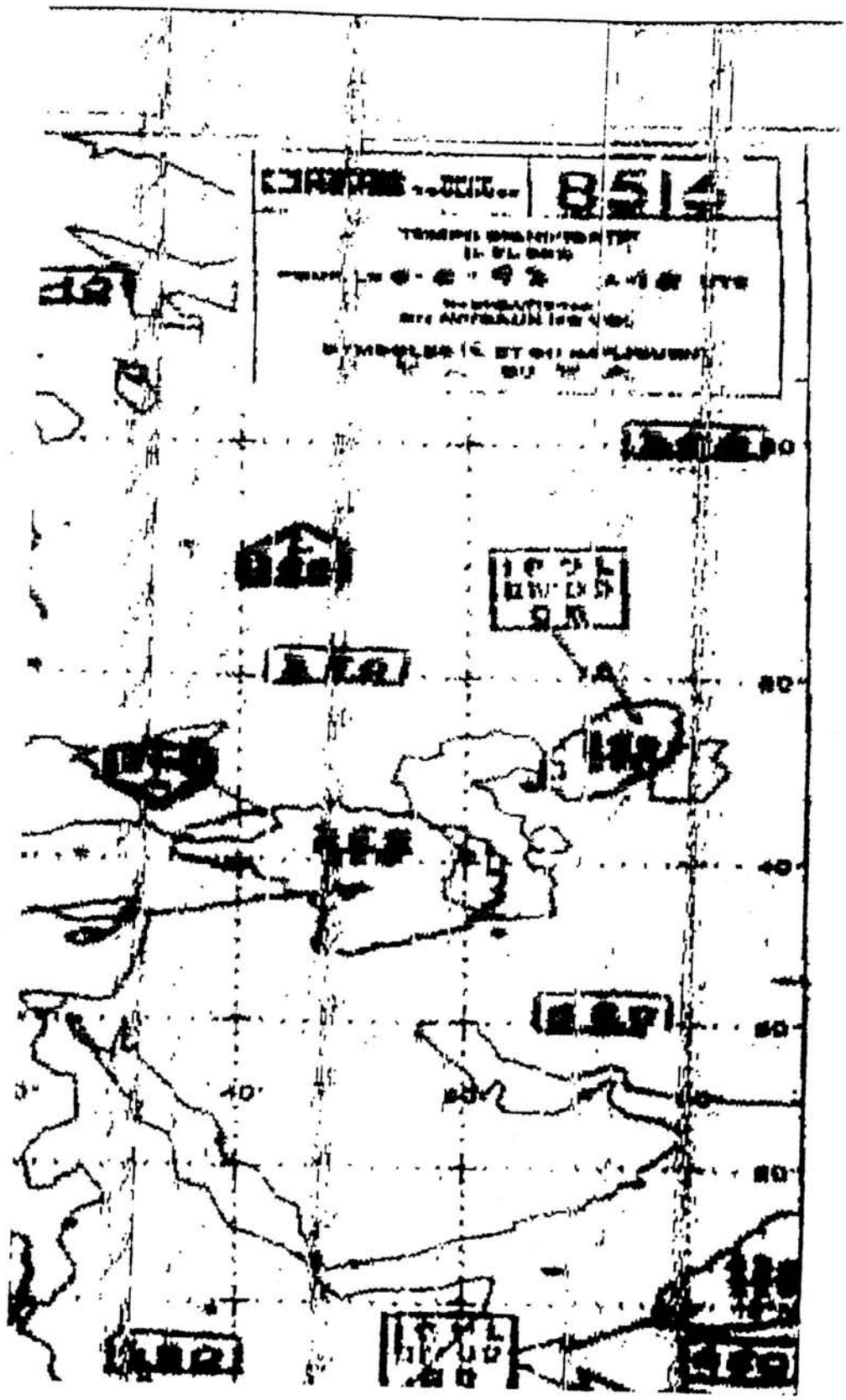
Sempre que se escolhe a opção Seleccionar, é enviada uma mensagem CM_SELECCIONA, que, como já vimos, é respondida pela função *CMSelecciona()*

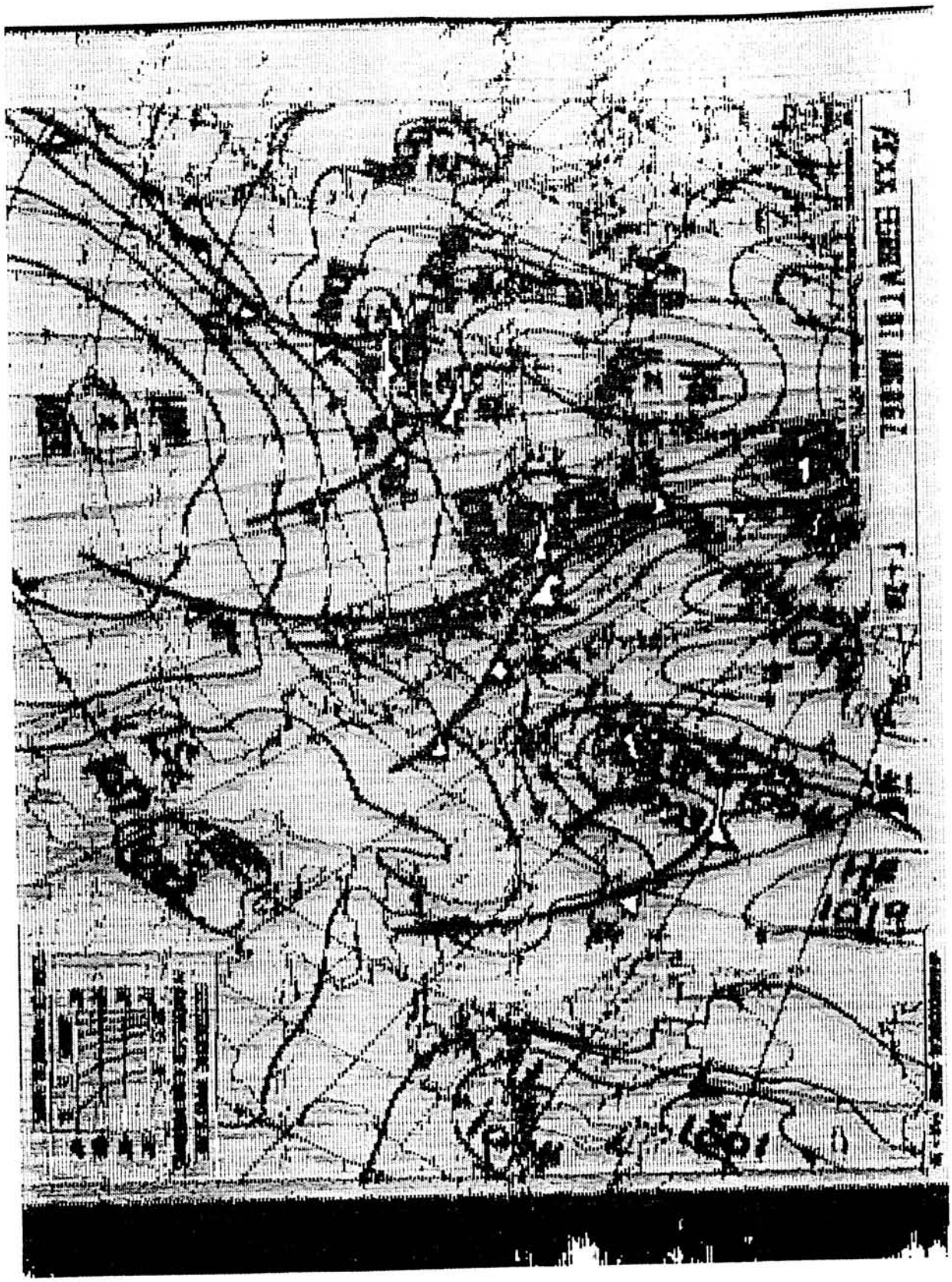
No programa Prodep.cpp, este menu é chamado da seguinte forma:

```
void Janela::GetWindowClass( WNDCLASS& WndClass )
{
     TWindow::GetWindowClass( WndClass );
        // chama o menu e o icone declarados em dial.rc
     WndClass.lpszMenuName = "CMDLGAPMENU";
          WndClass.hIcon = LoadIcon( GetApplication()-
>hInstance, "Projecto" );
};
```

## Algumas imagens...

Para concluir, apresentam-se alguns exemplos de imagens adquiridas através da placa





# ANEXO

```
// ObjectWindows - (C) Copyright 1993 b//

// ***************PROGRAMA PRODEP.CPP************

#include <owl.h>
#include <dialog.h>
#include <windobj.h>
#include <stdio.h>
#include <string.h>

#include <bchkbox.h>
#include <bradio.h>

#include <filedial.h>
#include <edit.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <dos.h>
#include <bwcc.h>


#include <alloc.h>

#include "cmdlgapr.h"



#define limiar_sync 150
#define salta 1

#define NumColors 16
#define NOME_JAN "projecto"


#define ficheiro_dados "dados"

typedef unsigned char byte;

//n_dados é o nº de amostras que se vão recolher

long n_dados=200000;


// tamanho do bitmap inicial
int BitMapWidth  = 800;
int BitMapHeight = 610;


/*
 paramet é um valor que se utiliza para corrigir a imagem
que vem distorcida
  do gravador podendo  ser alterada pelo utilizador, através
do diálogo PARÂMETROS
  */
float paramet=1.0112;
```

```
// INTR é o tipo de interrupção recebida da placa e que pode
ser alterada pelo utilizador
int INTR=11;
/* frequencia é a frequência de amostragem do sinal
analógico que entra na placa*/
int frequencia=1300;

int i,j,linha,coluna,ch,x;
FILE *fp;
byte b;
long impulsos,med,inicio,inicio_imagem,valor;
BOOL para=FALSE;
byte huge *p,valor_8259;
long g,n_dados_gravados,h,k;

HBITMAP   MeuBitMap,ClipboardBitmap;
DWORD modo=SRCCOPY;


//TParametro contém todos os valores que podem ser
modificados
// através do diálogo PARÂMETROS
struct TPParametro {
  BOOL inicio;
  BOOL fim;
  BOOL INTER3;
   BOOL INTER5;
   BOOL INTER7;
  char frequencia[30];
  char n_dados[30];
  char factor[30];

};




#ifdef __cplusplus
    #define __CPPARGS ...
#else
    #define __CPPARGS
#endif



/************************************************************
*************/

void interrupt (*oldhandler)(__CPPARGS);

//*********      Rotina de atendimento de uma interrupcao

void interrupt handler(__CPPARGS) /* if C++, need the the
ellipsis */
{
```

```
/* disable interrupts during the handling of the interrupt
*/
   disable();
/* increase the global counter */
    *(p+g)=inportb(768);
    g++;
 //  eoi=inportb(0x20);
   if (g>n_dados)
     {
        /* reset the old interrupt handler */
        outportb(771,255);
     setvect(INTR, oldhandler);

      }
/* reenable interrupts at the end of the handler */
   outportb(0x20,0x20);
   enable();
/* call the old routine */
   oldhandler();
}


/*************************************************************
*******************/



/**********declaração da aplicação, classe derivada de
TApplication************/

_CLASSDEF(AplicJanela)
class AplicJanela : public TApplication
{
public:
     AplicJanela( LPSTR, HINSTANCE, HINSTANCE, LPSTR, int );
     ~AplicJanela();
     virtual void InitMainWindow();
};



/*************************************************************
************************
* declaração da classe Janela, derivada de TWindow que
inclui todos                *
*       os elementos necessários ao funcionamento da janela
principal do programa  *

*************************************************************
************************/


_CLASSDEF(Janela)
class Janela : public TWindow
{
public:
```

```
      BOOL ButtonDown;
      TPParametro Parametro;
        // funções já definidas em TWindow
    Janela( PTWindowsObject, LPSTR );
    virtual ~Janela();
    virtual LPSTR GetClassName();
    virtual void Paint( HDC, PAINTSTRUCT _FAR & );
         virtual void GetWindowClass( WNDCLASS& WndClass );

       // declaração de todos os items dos menus
    virtual void CMExit( RTMessage ) = [CM_FIRST +
CM_EXIT];
    virtual void CMUFileOpen( RTMessage ) = [CM_FIRST +
CM_U_FILEOPEN];
    virtual void CMUHelpAbout( RTMessage ) = [CM_FIRST +
CM_U_HELPABOUT];
    virtual void CMGuarda( RTMessage ) = [CM_FIRST +
CM_U_GUARDA];
       virtual void CMAjuda( RTMessage ) = [CM_FIRST +
CM_AJUDA];


    virtual void CMSel(RTMessage
)=[CM_FIRST+CM_SELECCIONA];
    virtual void CMNega(RTMessage )=[CM_FIRST+CM_NEGA];
    virtual void CMMetade(RTMessage )=[CM_FIRST+CM_METADE];
    virtual void CMDobro(RTMessage )=[CM_FIRST+CM_DOBRO];
    virtual void CMEsphor(RTMessage )=[CM_FIRST+CM_ESPHOR];
    virtual void CMEspver(RTMessage )=[CM_FIRST+CM_ESPVER];
    virtual void CMRodadir(RTMessage
)=[CM_FIRST+CM_RODADIR];
    virtual void CMRodaesq(RTMessage
)=[CM_FIRST+CM_RODAESQ];
    virtual void CMParametro(RTMessage
)=[CM_FIRST+CM_PARAMETRO];
    virtual void CMaquisicao(RTMessage
)=[CM_FIRST+CM_AQUISICAO];
    virtual void CMpara(RTMessage )=[CM_FIRST+CM_PARA];



    virtual void WMSize(TMessage&) = [WM_FIRST + WM_SIZE];

    virtual void WMLButtonDown(RTMessage Msg)
   = [WM_FIRST + WM_LBUTTONDOWN];
    virtual void WMLButtonUp(RTMessage Msg)
   = [WM_FIRST + WM_LBUTTONUP];
     virtual void WMMouseMove(RTMessage Msg)
   = [WM_FIRST + WM_MOUSEMOVE];


    int X1, Y1, X2, Y2, larg,alt,posx,posy;
    void AdjustScroller();
     void Desactiva();
     void ActivaMenu();
       void verifica();
```

```
      char szName[256];
           HDC HandleDC;
      HPEN ACaneta;
      HCURSOR Cur;
      LPLOGPALETTE MyLogPalette;


};


// declaração de TParametro, derivada de TDialog
class TParametro : public TDialog {
public:
  TParametro(PTWindowsObject AParent, int ResourceId);
};


/*********************FIM DA DECLARAÇÃO DE VARIÁVEIS
*****************************/



AplicJanela::AplicJanela( LPSTR lpszNam, HINSTANCE hInst,
HINSTANCE hPrevInst,
      LPSTR lpszCmdLn, int nCmdShw )
      : TApplication( lpszNam, hInst, hPrevInst, lpszCmdLn,
nCmdShw )
{
};

AplicJanela::~AplicJanela()
{
};


// inicializa a janela Janela
void AplicJanela::InitMainWindow()
{
      MainWindow = new Janela( NULL, NOME_JAN );
};

 // construtor do diÁlogo TParametro
TParametro::TParametro(PTWindowsObject AParent, int
ResourceId)
                  : TDialog(AParent, ResourceId)
{
  new TBCheckBox(this, ID_INICIO, NULL);
  new TBCheckBox(this, ID_FIM, NULL);

  new TBRadioButton(this,ID_INTER3,NULL);
  new TBRadioButton(this,ID_INTER5,NULL);
  new TBRadioButton(this,ID_INTER7,NULL);

  new TEdit(this, ID_FREQ,
         sizeof(((Janela *)Parent)->Parametro.frequencia));
  new TEdit(this, ID_NDADOS,
         sizeof(((Janela *)Parent)->Parametro.n_dados));
  new TEdit(this, ID_FACT,
```

```
        sizeof(((Janela *)Parent)->Parametro.factor));


  TransferBuffer = (void far*)&(((Janela *)Parent)-
>Parametro);
}


// construtor de Janela: função chamada quando se abre a
janela, no inicio do programa
Janela::Janela( PTWindowsObject AParent, LPSTR ATitle )
     : TWindow( AParent, ATitle )
{

     HDC MemDC,DC;
     BYTE RedVals[NumColors], GreenVals[NumColors],
BlueVals[NumColors];



     strcpy( szName, "" );            // Empty the file name
string

     Attr.Style |=WS_MAXIMIZE | WS_VSCROLL | WS_HSCROLL;
     Scroller = new TScroller(this, 8, 15, 80, 60);
// inicialização da Palette usada,cujo nº de tons de
cinzento é dado por NumColors
 for (int i=0; i<=NumColors; i++)
  {
   RedVals[i] = i*(int)(255/(NumColors-1));
   GreenVals[i]= i*(int)(255/(NumColors-1));
   BlueVals[i] = i*(int)(255/(NumColors-1));
  }

       MyLogPalette = (LPLOGPALETTE)
  farmalloc(sizeof(LOGPALETTE) + sizeof(PALETTEENTRY) *
NumColors);
  MyLogPalette->palVersion = 0x300;
  MyLogPalette->palNumEntries = NumColors;
  for (i = 0; i < NumColors; ++i)
  {
    MyLogPalette->palPalEntry[i].peRed = RedVals[i];
    MyLogPalette->palPalEntry[i].peGreen = GreenVals[i];
    MyLogPalette->palPalEntry[i].peBlue = BlueVals[i];
    MyLogPalette->palPalEntry[i].peFlags = PC_RESERVED;
  }

  // fim do prenchimento da palette usada


Parametro.inicio=Parametro.fim=Parametro.INTER5=Parametro.IN
TER7=FALSE;
       Parametro.INTER3=TRUE;
```

```
      ButtonDown = FALSE;
      X1= Y1= X2= Y2= larg=alt=0;

      ACaneta = CreatePen(PS_DASH, 1, 100);


      DC = GetDC(0);
         // criação do bitmap
      MeuBitMap = CreateCompatibleBitmap(DC, BitMapWidth,
BitMapHeight);
      MemDC = CreateCompatibleDC(DC);
      SelectObject(MemDC,MeuBitMap);
      PatBlt(MemDC, 0, 0, BitMapWidth, BitMapHeight,
WHITENESS);
      DeleteDC(MemDC);
      ReleaseDC(0, DC);

         // chama o dialogo "About"
      GetApplication()->ExecDialog(new TDialog(this,
"About"));

};



 // destructor de Janela
Janela::~Janela()
{
  DeleteObject(MeuBitMap);
  DeleteObject(ACaneta);
  farfree(MyLogPalette);
   // desactiva interrupcao
    outportb(771,255);
    setvect(INTR, oldhandler);

};

LPSTR Janela::GetClassName()
{
      return NOME_JAN;
};


void Janela::GetWindowClass( WNDCLASS& WndClass )
{
      TWindow::GetWindowClass( WndClass );
         // chama o menu e o icone declarados em dial.rc
      WndClass.lpszMenuName = "CMDLGAPMENU";
           WndClass.hIcon = LoadIcon( GetApplication()-
>hInstance, "Projecto" );
};


/*************************************************************
******************
      função chamada sempre que se altera o tamanho da
janela ou se carrega na
```

```
   ScrollBar e que chama AdjustScroller() para mudar a
posição do bitmap em relação à
   janela visível, de modo a que se possa ver a totalidade
do bitmap, mesmo quando este
    é maior do que a janela

**************************************************************
*****************/
 void Janela::WMSize(TMessage& Msg)
{
  TWindow::WMSize(Msg);
  if ( (Msg.WParam != SIZEICONIC) )
    AdjustScroller();
};

void Janela::AdjustScroller()
{
  RECT ClientRect;

  GetClientRect(HWindow, &ClientRect);
  Scroller->SetRange((BitMapWidth-ClientRect.right -
ClientRect.left),
      (BitMapHeight-ClientRect.bottom - ClientRect.top));
  Scroller->ScrollTo(0, 0);

  InvalidateRect(HWindow, NULL, TRUE);
};




 //****************ROTINAS DO RATO ************************

  void Janela::WMLButtonDown(RTMessage Msg)
{

     // muda o cursor
  Cur = SetCursor(LoadCursor(NULL, IDC_CROSS));

  InvalidateRect(HWindow, NULL, TRUE);
  if ( !ButtonDown )
  {


    ButtonDown = TRUE;
    SetCapture(HWindow);
    HandleDC = GetDC(HWindow);
    SelectObject(HandleDC, ACaneta);

  X1 = Msg.LP.Lo;
  Y1 = Msg.LP.Hi;
  X2 = Msg.LP.Lo;
  Y2 = Msg.LP.Hi;

  SelectObject(HandleDC, GetStockObject(NULL_BRUSH));
  SetROP2(HandleDC, R2_NOT);
```

```
    Rectangle(HandleDC, X1, Y1, X2, Y2);
   }
};

void Janela::WMMouseMove(RTMessage Msg)
{
  if ( ButtonDown )
  {

   Rectangle(HandleDC, X1, Y1, X2, Y2);
  X2 =  Msg.LP.Lo;
  Y2 = Msg.LP.Hi;
  Rectangle(HandleDC, X1, Y1, X2, Y2);


    }
}

void Janela::WMLButtonUp(RTMessage)
{
  if ( ButtonDown )
  {

  Rectangle(HandleDC, X1, Y1, X2, Y2);
  SetROP2(HandleDC, R2_COPYPEN);

  Rectangle(HandleDC, X1, Y1, X2, Y2);

    larg=abs(X2-X1);
    alt=abs(Y2-Y1);
    // altera-se X1 e Y1,de modo a apontarem sempre o canto
superior esquerdo
    if (X1>X2) X1=X2;
    if (Y1>Y2) Y1=Y2;

    ButtonDown = FALSE;
    ReleaseCapture();
    ReleaseDC(HWindow, HandleDC);

          if (larg>1) ActivaMenu();

     else Desactiva();




      // regressa ao cursor anterior
     SetCursor(Cur);


  }
}

//**********************FIM DAS ROTINAS DO RATO
*******************
```

```
//************* ROTINA QUE TRATA DA AQUISIÇAO DE DADOS DA
PLACA *************
void Janela::CMaquisicao( RTMessage )
{



    para=FALSE;
    p=(byte huge *) farmalloc(n_dados+3000);
    if (p==NULL) MessageBox(HWindow,"\n não há memória
suficiente  \n" , "Mensagem", MB_OK);

     n_dados_gravados=0;
     g=0;

   // Seleccao da frequencia de amostragem (frequencia
minima 160 Hz)

       if (frequencia>160) valor=(long)
(7600000/frequencia);

       outportb(771,255); outportb(772,0x36);  // control
word para o C0
       // com dois bytes a serem escritos no contador

       // primeiro manda-se o byte menos significativo
       outportb(771,63);  outportb(772,(byte)(valor % 256));

       // em seguida o byte mais significativo
       outportb(771,63);  outportb(772,(byte)(valor / 256));


  // INICIALIZA A LEITURA POR INTERRUPÇÕES

   /* save the old interrupt vector */
   oldhandler = getvect(INTR);
   /* install the new interrupt handler */
   setvect(INTR, handler);
   //activação de interrupções do IRQ3 do 8259
   outportb(0x21,0);

   // Escolha do canal de entrada do AD
   outportb(769,0);

    HPALETTE ThePalette, OldPalette,ThePalette2,
OldPalette2;
 HDC TheDC,MemDC;

    /* criação de 2 palettes: uma para a janela onde se vão
colocar os pixeis
     e outra para o bitmap que se usar; vão-se colocar
pixeis simultâneamente
     na janela e no bitmap, de modo a que se possa ver a
imagem a ser desenhada
```

```
     enquanto se adquire o sinal e não apenas no final da
aquisição, o que aconteceria caso
     se desenhasse unicamente no bitmap .Deste modo, caso a
janela seja reinicializada,
     desaparece da janela tudo o que já se tinha desenhado,
mas que se pode ver quando
     acabar a aquisição e se visualizar o bitmap, fazendo
BitBlt(); */
    ThePalette2 = CreatePalette(MyLogPalette);

    TheDC = GetDC(HWindow);
    MemDC = CreateCompatibleDC(TheDC);
    SelectObject(MemDC,MeuBitMap);
    OldPalette2 = SelectPalette(MemDC, ThePalette2, FALSE);
    RealizePalette(MemDC);


  ThePalette = CreatePalette(MyLogPalette);
  OldPalette = SelectPalette(TheDC, ThePalette, FALSE);
  RealizePalette(TheDC);



    PatBlt(MemDC, 0, 0, BitMapWidth, BitMapHeight,
WHITENESS);

    BitBlt(TheDC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0,
0, modo);



   BWCCMessageBox(HWindow,"Ligue o gravador" , "Mensagem",
MB_OK);

    EnableMenuItem(GetMenu( HWindow ),CM_AQUISICAO  ,
MF_BYCOMMAND | MF_GRAYED );
    EnableMenuItem(GetMenu( HWindow ),CM_PARA  ,
MF_BYCOMMAND | MF_ENABLED );
     EnableMenuItem(GetMenu( HWindow ),CM_PARAMETRO  ,
MF_BYCOMMAND | MF_GRAYED );
     Desactiva();
    DrawMenuBar(HWindow);

     // enable de interrupt÷es
   outportb(771,12);

    ///**********************


  //INICIO DA LEITURA E DESENHO DA IMAGEM NO ECRAN
   while(g<n_dados)
    {

  fp=fopen(ficheiro_dados,"wb");
  n_dados_gravados=0;
```

```
  med=(long) frequencia/paramet;    // 1.012
 // vai ser encontrado o inicio da imagem
 // o bit B0 ( B0=1) da porta 770 corresponde ao fim de
conversao

    k=0;
    h=0;              // não se pode ultarpassar g
    while (k<100) { while(h>=g); if ( *(p+h)>30) {k++;h++;}
else { k=0; h++;}}
   inicio_imagem=h-k;
 // h fica a apontar para o "desvio" do inicio da imagem
relativamente a p



   h=inicio_imagem-50;
   coluna=linha=1;


    while (h<(long)n_dados)
      { if (coluna>=(int) (med/salta)) { coluna=0; linha++;
}
     coluna++;                          // 1 9 7 15
     int cor= (byte)  NumColors* (*(p+h)) / 255 ;
     SetPixel(TheDC,coluna,linha,PALETTEINDEX(cor));
     SetPixel(MemDC,coluna,linha,PALETTEINDEX(cor));

     verifica();
     if (para) { n_dados=g;
              n_dados=h;
              /* reset the interrupt handler */
               outportb(771,255);
               setvect(INTR, oldhandler);

                  }
     h+=salta;
     while(h>=g)  { }
            if(n_dados_gravados<g)
                    { fputc((int) *(p+n_dados_gravados),fp);
                      n_dados_gravados++;
                    }
      }



 } // fim de leitura e escrita da imagem no ecran



   fclose(fp);

   farfree(p);

  SelectPalette(TheDC, OldPalette, FALSE);
  SelectPalette(MemDC, OldPalette2, FALSE);
```

```
  ReleaseDC(HWindow, TheDC);
  DeleteObject(ThePalette);
  DeleteObject(ThePalette2);

  DeleteDC(MemDC);

    EnableMenuItem(GetMenu( HWindow ),CM_AQUISICAO  ,
MF_BYCOMMAND | MF_ENABLED );
     EnableMenuItem(GetMenu( HWindow ),CM_PARA  ,
MF_BYCOMMAND | MF_GRAYED );
     EnableMenuItem(GetMenu( HWindow ),CM_PARAMETRO  ,
MF_BYCOMMAND | MF_ENABLED );
     ActivaMenu();
    DrawMenuBar(HWindow);

     BWCCMessageBox(HWindow,"Fim de aquisição" , "Mensagem",
MB_OK);

}
//************************FIM DA ROTINA DE AQUISIÇÃO
************************



// função chamada quando se carrega no item Stop e que faz
terminar a aquisição
void Janela::CMpara( RTMessage Msg )
{
    BWCCMessageBox(HWindow," Stop" , "Mensagem", MB_OK);
  /* reset the interrupt handler */
        outportb(771,255);
     setvect(INTR, oldhandler);

   para=TRUE;
}


/* função que é chamada durante a aquisição e que  vai
verificar quais as
  mensagens enviadas pelo sistema operativo, agindo de
acordo com o recebido.
  é esta função que permite que se possa parar a
aquisição,caso a mensagem seja CM_PARA
  ou, sem terminar a aquisição, correr outros programas
*/
void Janela::verifica()
{
MSG msg;
BOOL fRetVal = TRUE;

while (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
 {

    if (msg.message == CM_PARA) para=TRUE;
```

```
    TranslateMessage(&msg);
    DispatchMessage(&msg);
 }

}


//  desactiva menus quando não são necessários
void Janela::Desactiva()
        {
       EnableMenuItem(GetMenu( HWindow ),CM_SELECCIONA   ,
MF_BYCOMMAND | MF_GRAYED );
         EnableMenuItem(GetMenu( HWindow ),CM_DOBRO   ,
MF_BYCOMMAND | MF_GRAYED );
       EnableMenuItem(GetMenu( HWindow ),CM_METADE   ,
MF_BYCOMMAND | MF_GRAYED );
       EnableMenuItem(GetMenu( HWindow ),CM_ESPHOR   ,
MF_BYCOMMAND | MF_GRAYED );
       EnableMenuItem(GetMenu( HWindow ),CM_ESPVER   ,
MF_BYCOMMAND | MF_GRAYED );
       EnableMenuItem(GetMenu( HWindow ),CM_RODADIR   ,
MF_BYCOMMAND | MF_GRAYED );
       EnableMenuItem(GetMenu( HWindow ),CM_RODAESQ   ,
MF_BYCOMMAND | MF_GRAYED );

     DrawMenuBar( HWindow );

     }

//reactiva os menus
void Janela::ActivaMenu()
        {
       EnableMenuItem(GetMenu( HWindow ),CM_SELECCIONA   ,
MF_BYCOMMAND | MF_ENABLED );
         EnableMenuItem(GetMenu( HWindow ),CM_DOBRO   ,
MF_BYCOMMAND | MF_ENABLED );
       EnableMenuItem(GetMenu( HWindow ),CM_METADE   ,
MF_BYCOMMAND | MF_ENABLED );
       EnableMenuItem(GetMenu( HWindow ),CM_ESPHOR   ,
MF_BYCOMMAND | MF_ENABLED );
       EnableMenuItem(GetMenu( HWindow ),CM_ESPVER   ,
MF_BYCOMMAND | MF_ENABLED );
       EnableMenuItem(GetMenu( HWindow ),CM_RODADIR   ,
MF_BYCOMMAND | MF_ENABLED );
       EnableMenuItem(GetMenu( HWindow ),CM_RODAESQ   ,
MF_BYCOMMAND | MF_ENABLED );



     DrawMenuBar( HWindow );

     }
```

```
    // esta função vem já predefinida na classe TWindow e é
chamada sempre
    // que o sistema operativo detecta mudanças na janela
activa

void Janela::Paint( HDC HandleDC, PAINTSTRUCT & PaintInfo )
{


     HDC MemDC;


          if (larg>1) ActivaMenu();

          else Desactiva();

       MemDC = CreateCompatibleDC(PaintInfo.hdc);
       SelectObject(MemDC, MeuBitMap);
       BitBlt(PaintInfo.hdc, 0, 0,BitMapWidth, BitMapHeight,
MemDC, 0, 0, modo);
       DeleteDC(MemDC);


}


 //**************** ROTINAS DO MENU ************************



 // as seguintes rotinas são chamadas sempre que se
selecciona um item do
 //menu Operações


void Janela::CMSel(RTMessage Msg)
{

 // coloca no bitmap principal apenas o que foi seleccionado
com o rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;


   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);
```

```
   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(MemDC, 0, 0,BitMapWidth, BitMapHeight, Mem2DC, 0,
0,SRCCOPY);
    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

    X1= Y1= X2= Y2= larg=alt=0;
    Desactiva();

}


void Janela::CMCopia(RTMessage Msg)
{

    HDC MemDC,DC,Mem2DC;

   HPALETTE ThePalette,OldPalette;


   DC=GetDC(HWindow);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(Mem2DC,MeuBitMap);



    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight,
MemDC,X1,Y1,SRCCOPY );

MemDC=CreateCompatibleDC(DC);

ClipboardBitmap=CreateCompatibleBitmap(DC,larg,alt)
Mem2DC=CreateCompatibleDC(DC);
SelectObject(MemDC,ClipboardBitmap);
ThePalette = CreatePalette(MyLogPalette);
OldPalette=SelectPalette(MemDC,ThePalette,FALSE);
RealizePalette(MemDC);

BitBlt(MemDC, 0, 0,larg,alt, Mem2DC,X1,Y1,SRCCOPY );


    //  poe no clip
    OpenClipboard(HWindow);
    EmptyClipboard();
```

```
    SetClipboardData(CF_BITMAP,ClipboardBitmap);
    CloseClipboard();

    DeleteDC(Mem2DC);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

    X1= Y1= X2= Y2= larg=alt=0;
    Desactiva();

}


void Janela::CMDobro( RTMessage Msg )
{
// faz o zoom para o dobro do rectangulo seleccionado com o
rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;



   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);


    StretchBlt(MemDC,0, 0,2*larg,2*alt, Mem2DC, 0,
0,larg,alt,SRCCOPY);


    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

     X1= Y1= X2= Y2= larg=alt=0;
```

```
      Desactiva();
}

  void Janela::CMMetade( RTMessage Msg )
{

  // faz o zoom para metade do rectangulo seleccionado com o
rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;




   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);


    StretchBlt(MemDC,0, 0,(int)larg/2,(int) alt/2, Mem2DC,
0, 0,larg,alt,SRCCOPY);


    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

     X1= Y1= X2= Y2= larg=alt=0;
     Desactiva();
}


   void Janela::CMNega( RTMessage Msg )
{

//faz o negativo da imagem, modificando o modo como o bitmap
 passado para a janela
```

```
   HDC MemDC,DC;


    if (modo==NOTSRCCOPY) modo=SRCCOPY ;
      else modo=NOTSRCCOPY;

   DC=GetDC(HWindow);

   MemDC=CreateCompatibleDC(DC);


   SelectObject(MemDC, MeuBitMap);

    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

    Desactiva();
}




void Janela::CMEsphor( RTMessage Msg )
{

   // faz o espelho horizontal do rectangulo seleccionado
com o rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;



   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);


    StretchBlt(MemDC,larg, 0,-larg,alt, Mem2DC, 0,
0,larg,alt,SRCCOPY);
```

```
    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

     X1= Y1= X2= Y2= larg=alt=0;

     Desactiva();
}

  void Janela::CMEspver( RTMessage Msg )
{

// faz o espelho vertical do rectangulo seleccionado com o
rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;



   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);


    StretchBlt(MemDC,0, alt,larg,-alt, Mem2DC, 0,
0,larg,alt,SRCCOPY);


    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

     X1= Y1= X2= Y2= larg=alt=0;
```

```
     Desactiva();
}

 void Janela::CMRodadir( RTMessage Msg )
{

 // roda para a direita o rectangulo seleccionado com o rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;
   int i,j;
   COLORREF cor;

    // muda o cursor
     Cur = SetCursor(LoadCursor(NULL, IDC_WAIT));


   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);
    for (i=0;i<larg;i++)
    for (j=0;j<alt;j++)
    {
     cor=GetPixel(Mem2DC,i,j);
     SetPixel(MemDC,alt-j,i,cor);
     }

   // StretchBlt(MemDC,larg, alt,-larg,-alt, Mem2DC, 0,
0,larg,alt,SRCCOPY);

   // BitBlt(MemDC, 0, 0,BitMapWidth, BitMapHeight, Mem2DC,
0, 0,SRCCOPY);
    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

     X1= Y1= X2= Y2= larg=alt=0;
     Desactiva();
```

```
      // regressa ao cursor anterior
     SetCursor(Cur);
}

void Janela::CMRodaesq( RTMessage Msg )
{

  // roda para a esquerda o rectangulo seleccionado com o
rato

   HDC MemDC,DC,Mem2DC;

   HBITMAP nbitmap;
   int i,j;
   COLORREF cor;

    // muda o cursor
     Cur = SetCursor(LoadCursor(NULL, IDC_WAIT));



   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC, larg,alt);
   MemDC=CreateCompatibleDC(DC);
   Mem2DC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);
   SelectObject(Mem2DC, nbitmap);

   PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);

    BitBlt(Mem2DC, 0, 0,larg,alt, MemDC,X1,Y1,SRCCOPY );


    PatBlt(MemDC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);
    for (i=0;i<larg;i++)
    for (j=0;j<alt;j++)
    {
     cor=GetPixel(Mem2DC,i,j);
     SetPixel(MemDC,j,larg-i,cor);
     }


    BitBlt(DC, 0, 0,BitMapWidth, BitMapHeight, MemDC, 0, 0,
modo);

    DeleteDC(Mem2DC);
    DeleteObject(nbitmap);
    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );


    X1= Y1= X2= Y2= larg=alt=0;
    Desactiva();
```

```
      // regressa ao cursor anterior
     SetCursor(Cur);
}

 //***fim das funções inerentes aos items do menu Operações


 // função chamada quando se escolhe o item Parâmetros
void Janela::CMParametro( RTMessage Msg )
{


 char ALabel[255];

    //converte os valores para string

    ltoa(n_dados,Parametro.n_dados,10);
    itoa(frequencia,Parametro.frequencia,10);
    gcvt(paramet,15,Parametro.factor);


   // chama o diálogo TParametro, definido em dial.rc

  if ( GetModule()->ExecDialog(new TParametro(this,
ID_DIALOGO)) == IDOK )
  {

   frequencia= atof( Parametro.frequencia);
   n_dados= atol( Parametro.n_dados);
   paramet=atof(Parametro.factor);
   if ( Parametro.INTER3 ) INTR=11;
    else if ( Parametro.INTER5) INTR=13;
    else INTR=15;

  }

}

void Janela::CMExit( RTMessage Msg )
{
     TWindow::CMExit( Msg );
}



// função que permite seleccionar um ficheiro para abrir
// e que, posteriormente o lê e desenha no bitmap
void Janela::CMUFileOpen( RTMessage )
{

     int i,j,x,cor;
     FILE *fp;
        char ch;
     HDC MemDC,DC,Mem2DC;
     HPALETTE ThePalette, OldPalette;
      HBITMAP nbitmap;
```

```
      char FileName[40];

     char CaptionBuffer [/*MAXPATH+*/ 12 /* NOME_JAN*/ + 2
/*" "*/ + 1 /*'\0'*/ ];

      // chama dialogo definido em dial.rc, do tipo
TFileDialog
        if (GetApplication()->ExecDialog(new
TFileDialog(this,
      SD_FILEOPEN, strcpy(FileName, "*.map"))) == IDOK)
      {

    //***************** Lê FICHEIRO
************************

     InvalidateRect( HWindow, NULL, TRUE );
    fp=fopen(FileName,"rb");



      strcpy(CaptionBuffer, NOME_JAN);
      strcat(CaptionBuffer, ": ");
      strcat(CaptionBuffer, strlwr(FileName));
     SetWindowText(HWindow, CaptionBuffer);


    // muda o cursor
     Cur = SetCursor(LoadCursor(NULL, IDC_WAIT));



   DC=GetDC(HWindow);
   nbitmap = CreateCompatibleBitmap(DC,BitMapWidth,
BitMapHeight);

   Mem2DC=CreateCompatibleDC(DC);


        i=j=1;

       // selecciona palette

     ThePalette = CreatePalette(MyLogPalette);

     SelectObject(Mem2DC, nbitmap);

          OldPalette = SelectPalette(Mem2DC, ThePalette,
FALSE);
           RealizePalette(Mem2DC);
```

```
       PatBlt(Mem2DC, 0, 0, BitMapWidth,
BitMapHeight,WHITENESS);



          int npl;
        fscanf(fp,"%d",&npl);
        int xx;

         do { xx= fgetc(fp);
           if (i>=npl) { i=0; j++; }
              i++;
           cor= (int) NumColors* (xx) / 255 ;
           SetPixel(Mem2DC,i,j,PALETTEINDEX(cor));
            } while (xx!=EOF);


      fclose(fp);




       DeleteObject(ThePalette);


       MeuBitMap = CreateCompatibleBitmap(DC, BitMapWidth,
BitMapHeight);
       MemDC=CreateCompatibleDC(DC);
       SelectObject(MemDC, MeuBitMap);

     BitBlt(MemDC, 0, 0,BitMapWidth, BitMapHeight, Mem2DC,
0, 0, modo);
     DeleteDC(Mem2DC);
     DeleteDC(MemDC);
     ReleaseDC(HWindow, DC );


      DeleteObject(nbitmap);

        // regressa ao cursor anterior
     SetCursor(Cur);

     }; //if


     X1= Y1= X2= Y2= larg=alt=0;

}


// função que permite seleccionar um ficheiro onde se vai
posteriormente colocar
// os dados referentes à imagem presentemente na janela
activa
void Janela::CMGuarda( RTMessage msg)
{
char FileName[30];
```

```
HDC MemDC,DC;


   int i,j;
   byte cor;


   // chama dialogo definido em dial.rc, do tipo TFileDialog
if (GetApplication()->ExecDialog(new TFileDialog(this,
      SD_FILESAVE, strcpy(FileName, "*.MAP")))
         == IDOK)
      {


      DC=GetDC(HWindow);

   MemDC=CreateCompatibleDC(DC);

   SelectObject(MemDC, MeuBitMap);

      fp = fopen ( FileName,"wb" );


    // muda o cursor
     Cur = SetCursor(LoadCursor(NULL, IDC_WAIT));


    fprintf(fp,"%d",600);
    for (i=1;i<=600;i++)
    for (j=1;j<=600;j++)
    {
     cor = GetRValue( GetPixel(MemDC,j,i));
     fputc((int) cor,fp);
     }

    fclose(fp);

    DeleteDC(MemDC);
    ReleaseDC(HWindow, DC );

  } // if

    X1= Y1= X2= Y2= larg=alt=0;
    Desactiva();

      // regressa ao cursor anterior
     SetCursor(Cur);


}
```

```
// função chamada quando se selecciona o item "Acerca" do
manu "Help"
void Janela::CMUHelpAbout( RTMessage )
{

    GetApplication()->ExecDialog(new TDialog(this,
"About"));
}
// função chamada quando se selecciona o item "Ajuda" do
manu "Help"
 void Janela::CMAjuda( RTMessage )
{

    GetApplication()->ExecDialog(new TDialog(this, Ajuda));
}


   //********************** PROGRAMA PRINCIPAL
********************


int PASCAL WinMain( HINSTANCE hInst, HINSTANCE hPrevInst,
LPSTR lpszCmdLn,
     int nCmdShw )
{

     AplicJanela App( "dialogo", hInst, hPrevInst,
lpszCmdLn, nCmdShw );

     App.Run();
     return App.Status;
}

// **********************FIM DO PROGRAMA****************
```

```
// ObjectWindows - (C) Copyright 1993

// cmdlgapr.h


#define ID_GRAVADOR      600
#define ID_INTER3    43
#define ID_INTER5    44
#define ID_INTER7    45
#define ID_FACT      47
#define Ajuda   0x250
#define CM_PARA      292
#define CM_AQUISICAO     291
#define ID_DIALOGO  0x150
#define ID_FREQ      206
#define ID_NDADOS    207
#define ID_INTER     208
#define ID_INICIO    106
#define ID_FIM  107
#include <owlrc.h>

#define CM_U_FILEOPEN          0x100
#define    CM_U_GUARDA           0X101

#define CM_SELECCIONA          0x105

#define CM_NEGA              0x106
#define CM_DOBRO             0x107
#define CM_METADE             0x108
#define CM_ESPHOR            0x109
#define CM_ESPVER            0x110
#define CM_COLAR             0x111
#define CM_RODAESQ            0x120
#define CM_RODADIR            0x121
#define CM_PARAMETRO            0x122
#define CM_COPIAR            0x112



#define CM_U_HELPABOUT          0x200
#define CM_AJUDA            0x201
```

```
// ObjectWindows - (C) Copyright 1993
//
// FICHEIRO dial.rc

#include <windows.h>
#include "cmdlgapr.h"

CMDLGAPMENU MENU
BEGIN
     POPUP "&Ficheiro"
     BEGIN
          MENUITEM "&Abrir", CM_U_FILEOPEN
          MENUITEM "&Guardar", CM_U_GUARDA
          MENUITEM SEPARATOR
          MENUITEM "&Sair", CM_EXIT
     END

     POPUP "&Operações"
     BEGIN
          MENUITEM "&Negativo", CM_NEGA
          MENUITEM SEPARATOR
          MENUITEM "S&eleccionar", CM_SELECCIONA, GRAYED
          MENUITEM "&Copiar", CM_COPIAR, GRAYED
          POPUP "&Zoom"
          BEGIN
               MENUITEM "&Dobro", CM_DOBRO
               MENUITEM "&Metade", CM_METADE
          END

          POPUP "&Espelho"
          BEGIN
               MENUITEM "&Horizontal", CM_ESPHOR
               MENUITEM "&Vertical", CM_ESPVER
          END

          POPUP "&Rodar"
          BEGIN
               MENUITEM "E&squerda", CM_RODAESQ
               MENUITEM "D&ireita", CM_RODADIR
          END

     END

     MENUITEM "&Parâmetros", CM_PARAMETRO
     MENUITEM "&Aquisição", CM_AQUISICAO
     MENUITEM "&Stop", CM_PARA, GRAYED
     POPUP "\a&Help"
     BEGIN
          MENUITEM "A&juda", CM_AJUDA
          MENUITEM SEPARATOR
          MENUITEM "A&cerca", CM_U_HELPABOUT
     END

END

Projecto ICON "projec2.ico"
```

```
// ************DIALOGO  "About"*****************************
About DIALOG 8, 20, 255, 263
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
WS_SYSMENU
CLASS "BorDlg"
CAPTION "Acerca do Projecto"
FONT 8, "Helv"
BEGIN
     CONTROL "", 102, "BorShade", 1 | WS_CHILD | WS_VISIBLE,
7, 6, 245, 166
     CTEXT "Programa Projecto", 101, 71, 187, 111, 8,
WS_CHILD | WS_VISIBLE | WS_GROUP
     CTEXT "Copyright \251 1992/93", -1, 37, 198, 175, 8,
WS_CHILD | WS_VISIBLE | WS_GROUP
     CONTROL "", -1, "BorShade", 2 | WS_CHILD | WS_VISIBLE,
0, 230, 255, 2
     CONTROL "", 1, "BorBtn", 1 | WS_CHILD | WS_VISIBLE |
WS_GROUP | WS_TABSTOP, 109, 235, 36, 25
     CONTROL "", -1, "BorShade", BSS_GROUP | WS_CHILD |
WS_VISIBLE, 5, 183, 245, 42
     CONTROL "&i", 103, "BUTTON", BS_PUSHBUTTON | WS_CHILD |
WS_VISIBLE, -10, -10, 5, 5
     CTEXT " António José Esteves de Castro ", -1, 23, 211,
202, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
     CONTROL "", 111, "BorBtn", BBS_BITMAP | WS_CHILD |
WS_VISIBLE, 29, 10, 199, 156
END



//bitmap que aparece nos dialogos About e Acerca
1111 BITMAP "boneco.bmp"


// dialogo chamado quando se quer guardar imagem
SD_FILESAVE DIALOG 11, 25, 146, 144
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | DS_MODALFRAME
CLASS "BorDlg"
CAPTION "Guardar como"
BEGIN
     CONTROL "", -1, "BorShade", 1, 4, 53, 70, 89
     CONTROL "", -1, "BorShade", 1, 4, 4, 139, 18
     CONTROL "", -1, "BorShade", BSS_VDIP, 82, 43, 1, 101
     CONTROL "", -1, "BorShade", BSS_HDIP, 83, 43, 64, 1
     LTEXT " Nome&Ficheiro:", -1, 5, 7, 50, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", ID_FNAME, "EDIT", WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP | ES_AUTOHSCROLL, 59, 7, 79, 12
     LTEXT " Direct≤rio:", -1, 4, 27, 36, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     LTEXT "", ID_FPATH, 40, 27, 102, 10
     LTEXT " &Direct≤rios:", -1, 4, 43, 69, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", ID_DLIST, "LISTBOX", WS_CHILD | WS_VISIBLE
| WS_TABSTOP | LBS_STANDARD, 7, 57, 64, 82
```

```
     CONTROL "Button", IDOK, "BorBtn", BS_DEFPUSHBUTTON |
WS_TABSTOP, 97, 64, 36, 24
     CONTROL "&Cancel", IDCANCEL, "BorBtn", BS_PUSHBUTTON |
WS_CHILD | WS_VISIBLE | WS_TABSTOP, 97, 99, 34, 20
END


// dialogo chamado quando se quer abrir ficheiro com  imagem
SD_FILEOPEN DIALOG 9, 29, 197, 150
STYLE WS_POPUP | WS_CAPTION | WS_SYSMENU | DS_MODALFRAME
CLASS "BorDlg"
CAPTION "Abrir ficheiro"
BEGIN
     CONTROL "", -1, "BorShade", 1, 5, 4, 148, 18
     CONTROL "", -1, "BorShade", 1, 5, 57, 70, 89
     CONTROL "", -1, "BorShade", 1, 82, 57, 70, 89
     CONTROL "", -1, "BorShade", BSS_VDIP, 156, 0, 1, 150
     LTEXT " &NomeFicheiro:", -1, 6, 8, 50, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", ID_FNAME, "EDIT", WS_CHILD | WS_VISIBLE |
WS_BORDER | WS_TABSTOP | ES_AUTOHSCROLL, 60, 7, 88, 12
     LTEXT " Directório:", -1, 5, 29, 37, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     LTEXT "", ID_FPATH, 42, 29, 110, 10
     LTEXT " &Ficheiro:", -1, 5, 47, 69, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", ID_FLIST, "LISTBOX", WS_CHILD | WS_VISIBLE
| WS_TABSTOP | LBS_STANDARD, 8, 61, 64, 82
     LTEXT " &Directórios:", -1, 82, 47, 69, 10, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", ID_DLIST, "LISTBOX", WS_CHILD | WS_VISIBLE
| WS_TABSTOP | LBS_STANDARD, 85, 61, 64, 82
     CONTROL "&Ok", IDOK, "BorBtn", BS_DEFPUSHBUTTON |
WS_TABSTOP, 160, 4, 33, 21
     CONTROL "&Cancel", IDCANCEL, "BorBtn", BS_PUSHBUTTON |
WS_TABSTOP, 160, 34, 33, 20
END


// dialogo Ajuda
Ajuda DIALOG 8, 20, 255, 263
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION |
WS_SYSMENU
CLASS "BorDlg"
CAPTION "Acerca do Projecto"
FONT 8, "Helv"
BEGIN
     CONTROL "", 102, "BorShade", 1 | WS_CHILD | WS_VISIBLE,
7, 6, 245, 166
     CTEXT "ATENÇÃO", 101, 71, 187, 111, 8, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CTEXT "Para realizar todas as Operações excepto
Negativo", -1, 37, 198, 175, 8, WS_CHILD | WS_VISIBLE |
WS_GROUP
     CONTROL "", -1, "BorShade", 2 | WS_CHILD | WS_VISIBLE,
0, 230, 255, 2
```

```
     CONTROL "", 1, "BorBtn", 1 | WS_CHILD | WS_VISIBLE |
WS_GROUP | WS_TABSTOP, 109, 235, 36, 25
     CONTROL "", -1, "BorShade", BSS_GROUP | WS_CHILD |
WS_VISIBLE, 5, 183, 245, 42
     CONTROL "&i", 103, "BUTTON", BS_PUSHBUTTON | WS_CHILD |
WS_VISIBLE, -10, -10, 5, 5
     CTEXT "é necessário seleccionar previamente com o
rato", -1, 23, 211, 202, 8, WS_CHILD | WS_VISIBLE | WS_GROUP
     CONTROL "", 111, "BorBtn", BBS_BITMAP | WS_CHILD |
WS_VISIBLE, 29, 10, 199, 156
END


// dialogo usado para a modificação de parâmetros
ID_DIALOGO DIALOG 20, 20, 234, 196
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CLASS "BorDlg"
CAPTION "Modificação de Parâmetros"
FONT 8, "Helv"
BEGIN
     CONTROL "", -1, "borshade", 2 | WS_CHILD | WS_VISIBLE |
WS_GROUP, 0, 154, 234, 2
     LTEXT " &Detecção", -1, 122, 16, 104, 8, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", -1, "borshade", 1 | WS_CHILD | WS_VISIBLE |
WS_GROUP, 122, 24, 104, 28
     CONTROL "Detectar &Inicio", ID_INICIO, "borcheck",
BS_AUTOCHECKBOX | BBS_PARENTNOTIFY | WS_CHILD | WS_VISIBLE |
WS_GROUP | WS_TABSTOP, 126, 28, 96, 10
     CONTROL "Detectar &Fim", ID_FIM, "borcheck",
BS_AUTOCHECKBOX | BBS_PARENTNOTIFY | WS_CHILD | WS_VISIBLE,
126, 38, 96, 10
     LTEXT " &Tipo de Interrupções", -1, 9, 16, 85, 8,
WS_CHILD | WS_VISIBLE | WS_GROUP
     CONTROL "", -1, "borshade", 1 | WS_CHILD | WS_VISIBLE |
WS_GROUP, 9, 24, 85, 38
     CONTROL "IRQ&3", ID_INTER3, "borradio",
BS_AUTORADIOBUTTON | BBS_PARENTNOTIFY | WS_CHILD |
WS_VISIBLE | WS_GROUP | WS_TABSTOP, 13, 28, 73, 10
     CONTROL "IRQ&5", ID_INTER5, "borradio",
BS_AUTORADIOBUTTON | BBS_PARENTNOTIFY | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 13, 38, 74, 10
     CONTROL "IRQ&7", ID_INTER7, "borradio",
BS_AUTORADIOBUTTON | BBS_PARENTNOTIFY | WS_CHILD |
WS_VISIBLE | WS_TABSTOP, 13, 48, 74, 10
     CONTROL "", -1, "borshade", 1 | WS_CHILD | WS_VISIBLE |
WS_GROUP, 7, 70, 218, 20
     LTEXT " Fr&equência", -1, 8, 76, 40, 9, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", 1, "borbtn", 8193 | WS_CHILD | WS_VISIBLE |
WS_GROUP | WS_TABSTOP, 31, 162, 37, 25
     CONTROL "", 2, "borbtn", 8192 | WS_CHILD | WS_VISIBLE |
WS_TABSTOP, 146, 160, 37, 25
     EDITTEXT ID_FREQ, 50, 74, 166, 12, ES_LEFT | WS_CHILD |
WS_VISIBLE | WS_BORDER | WS_TABSTOP
     CONTROL "", 41, "BorShade", 1 | WS_CHILD | WS_VISIBLE,
7, 97, 218, 20
```

```
     EDITTEXT ID_NDADOS, 51, 101, 165, 12, ES_LEFT |
WS_CHILD | WS_VISIBLE | WS_BORDER | WS_TABSTOP
     LTEXT "&Nº Dados", -1, 12, 103, 35, 8, WS_CHILD |
WS_VISIBLE | WS_GROUP
     CONTROL "", 46, "BorShade", 1 | WS_CHILD | WS_VISIBLE,
7, 124, 219, 20
     LTEXT "Factor de &Correcção", -1, 13, 130, 70, 8,
WS_CHILD | WS_VISIBLE | WS_GROUP
     EDITTEXT ID_FACT, 86, 128, 130, 12, ES_LEFT | WS_CHILD
| WS_VISIBLE | WS_BORDER | WS_TABSTOP
END


  // **************FIM DE dial.rc
```

# Bibliografia

Ira Pohl, "C++ for C programmers",*The Benjamin/Cumming Publishing Company, Inc*

"Objectwindows for C++ Users guide", *Borland International, Inc*