3. Introducció al Borland C++ Builder.

3.1. Introducció.

En la realització del present projecte s'ha utilitzat com a llenguatge de programació el **C++**, i com a entorn d'aquest llenguatge s'ha fet ús de Borland C++ Builder, una ferramenta visual adaptada al sistema operatiu **Windows**.

En aquest capítol no s'intentarà donar al lector una visió del llenguatge de programació C++ ni tampoc com utilitzar l'entorn de programació **Borland Builder**, ja què aquesta informació queda fora de l'abast d'aquest projecte.

El que si que es va a fer, és donar una idea al lector de l'entorn de programació i les ferramentes més utilitzades al llarg del present projecte.

3.2. Llenguatge C++.

El llenguatge C++ es caracteritza per ser un llenguatge de programació orientat a objectes (**POO**), és a dir, així com el llenguatge **C** és un llenguatge orientat a funcions (amb les seues variables locals), açò és, un llenguatge amb parts reutilitzables per part del programador; al C++ podem "crear" a més d'aquestes funcions, una mena de nodes anomenats **classes** que a més de contindre variables, poden contindre com a propietats una sèrie de funciones definides dins del seu cos.

Aquestes classes a més, introdueixen el concepte de privaticitat, podent compartir en part o totalment les seues propietats. Així també s'introdueix el concepte d'herència, com a la capacitat que té una classe de compartir les propietats de les demés classes (**objectes**), així com també compartir amb les demés classes les propietats d'ell mateix.

Un exemple del que s'està dient ho podem veure a continuació, on es pot observar el contingut d'una classe, amb les seues variables, components visuals (propis d'aquest entorn Borland C++ Builder) i funcions, que després seran ben explicades.

```
//---------------------------------------------------------------------------
#ifndef mipsH
#define mipsH
//---------------------------------------------------------------------------
#include <vcl\Classes.hpp>
```

```
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\ComCtrls.hpp>
#include <vcl\ExtCtrls.hpp>
#include <vcl\Menus.hpp>
#include "registres.h"
#include "memoria.h"
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published: // IDE-managed Components
        TLabel *LInstruccio;
        TEdit *EInstruccio;
        TButton *BSeguent;
        TButton *BAceptar;
        TLabel *Label3;
        TLabel *Lop;
        TLabel *Lrs;
        TLabel *Lrt;
        TLabel *Lrd;
        TLabel *Lshamt;
        TLabel *Lfunct;
        TEdit *Eop;
        TEdit *Ers;
        TEdit *Ert;
        TEdit *Erd;
        TEdit *Eshamt;
        TEdit *Efunct;
        TLabel *LInmediat;
        TEdit *EInmediat;
        TLabel *LAdreca;
        TEdit *EAdreca;
        TButton *BCancelar;
        TMainMenu *MainMenu1;
        TMenuItem *Instr1;
        TMenuItem *Registres1;
        TMenuItem *Memoria1;
        TMenuItem *N1;
        TMenuItem *Eixir1;
        TMenuItem *Novainstrucci1;
        TMenuItem *N2;
        void __fastcall FormPaint(TObject *Sender);
        void __fastcall EInstruccioKeyPress(TObject *Sender, char &Key);
        void __fastcall BAceptarClick(TObject *Sender);
        void __fastcall BSeguentClick(TObject *Sender);
```

```
        void __fastcall BCancelarClick(TObject *Sender);
        void __fastcall Eixir1Click(TObject *Sender);
        void __fastcall Novainstrucci1Click(TObject *Sender);
        void __fastcall Registres1Click(TObject *Sender);
        void __fastcall Memoria1Click(TObject *Sender);


private:        // User declarations
public:                 // User declarations
   void Invertir (void);
   char TipusInstruccio (void);
   void DescompondreInstruccio (void);
   void Pas1 (void);
   void CrearBaseDades (void);
   void Activar (void);
   int BinariDecimal (int tamany,int bitMSB);
   void RenovaRegistres (void);
   void BorraRegistres (void);
   void NetejaText (void);
   int AcoplarJump (void);

        __fastcall TForm1(TComponent* Owner);

   int a,b;
   char IR[33],IR_usuari[33];
   int codiop;
   char tipus;
   int pas;
   int rs,rt,rd,shamt,funct;
   int inmediat,adreca_j;
   int ALUCero;

   char LControl[5][12]; //Fi de cadena.  [operacio][linees de control]
   char Linees[5][5][32]; //[operacio][pasos-1(el primer pas és comú)][Linees]

   POINT PC[4],ALUPC[7],MemIns[4],Registres[4],ALU[7],
        MemDad[4],ALUBot[7];
   POINT Desp1[4],Desp2[4],Control[4],ControlALU[4],ExtSigne[4];
   POINT Mux1[4],Mux2[4],Mux3[4],Mux4[4],Mux5[4],AND[2];
   POINT LMux5_PC[811],LPC_MemIns[16],LPC_ALUPC[236],L4[21];
   POINT LMemIns_Desp1[286],LMemIns_Control[136],
        LMemIns_RegLec1[136];
   POINT LMemIns_RegLec2[116],LMemIns_Mux1_0[91],
        LMemIns_Mux1_1[81];
   POINT LMemIns_ExtSigne[181],LMemIns_ControlALU[306];
   POINT LMux1[26],LMux3_Registres[516],LALUPC_ALUBot[221];
```

```
  POINT LALUPC_Mux4[316],LALUBot_Mux4[51],LMux4_Mux5[31];
  POINT LDesp1_Mux5[414],LDesp2_ALUBot[16],LExtSigne_Desp2[274];
  POINT LDadLleg1_ALU[56],LDadLleg2_Mux2[31],
      LDadLleg2_MemDad[171];
  POINT LExtSigne_Mux2[71],LMux2_ALU[16],LALU_MemDad[36];
  POINT LALU_Mux3[196],LMemDad_Mux3[26];
  POINT LCEscrReg[45],LCFuenteALU[163],LCEscrMem[275],
      LCALUOp[325];
  POINT LCMemaReg[368],LCLeerMem[575],LCSaltoCond[185];
  POINT SaltoIncond[435],LCRegDest[371],LCCero[120],LCAND_Mux4[82];
  POINT LCControlALU_ALU[68];

  TRect Area;
  POINT Diagonal[10];

  //Per al contingut de les línies
  TRect Contingut[12];
  int valor[10];
  char eALU[3],ALUc[4];

  TForm2 *Reg;
  TForm3 *Mem;

  //Contingut dels registres
  int r[32];
  int rpc;

  bool alerta;
};
//---------------------------------------------------------------------------
extern TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Com a principals estructures, i per no endinsar-se'n molt podem explicar una sèrie de característiques dels blocs principals:

- **Bloc d'includes**: On podem incloure les llibreries que seran necessàries per a l'elaboració del nostre programa. Una llibreria en C++, significa un arxiu que conté funcions ja definides i que podem utilitzar, sense tindre que realitzar-les nosaltres, amb el clar estalvi de temps que suposa. Tal i com es veu hi ha dos tipus d*'includes*, les pròpies llibreries de l'entorn *Borland Builder*, així com també llibreries pròpies del programador, a l'exemple es distingeixen aquests dos tipus de llibreries pels signes que acompanyen a la llibreria, així, per a llibreries (o *includes*) pròpies de l'entorn en el que estem treballant,

s'utilitzen els símbols **< >**, mentre que per a llibreries creades pel programador, s'utilitzen els símbols " ", açò ho podem veure a continuació:

```
#include <vcl\Classes.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Forms.hpp>
#include <vcl\ComCtrls.hpp>
#include <vcl\ExtCtrls.hpp>
#include <vcl\Menus.hpp>
#include "registres.h"
#include "memoria.h"
```

- **Definició de la classe**, especifiquem el nom que tindrà d'ara endavant l'objecte del que declararem certes variables, components visuals i funcions o procediments, com ja s'ha esmentat. El nom de la classe ve precedit de la **paraula reservada**: **class**, que significa que es va a declarar una classe, un exemple potser el següent:

  ```
  class TForm1 : public TForm
  ```

  on també s'afegeix la paraula reservada **public** , indicant així que aquesta classe Tform1 pot accedir lliurement als valors declarats com a públics a la classe TForm, que per fer-se'n una idea conté els procediments bàsics per realitzar una finestra.

- **Tipus de components de la classe**, que com s'observen són tres, **__published** que indica que aquestes variables i funcions van a ser publicats gràficament o que estan relacionades amb algun d'aquests components gràfics. **Private**, el contingut d'aquest "sector" indica que les funcions i variables esmentades dintre no podran ser utilitzades per cap classe o funció independent d'aquesta classe, a menys què es declare a aquesta classe (TForm1 al exemple) que hi ha classes o funcions germanes (**friends**) o l'altra opció és declarar classes mitjançant l'anomenada herència, açò és, de la mateixa manera que s'ha declarat TForm1 com a pública de la classe TForm, la podríem haver declarat com a,

  ```
  class TForm1 : private TForm
  ```

  i així compartir també les variables i procediments privats d'aquesta classe. Per últim, es troben els components declarats com a **public**, la

qual cosa ens indica que aquests elements els podem compartir amb qualsevol altra classe o cap altra funció on s'haja inclòs aquesta classe que conté les entitats públiques, així com també per qualsevol variable del tipus *TForm*, en aquest cas..

Al mateix temps, podem observar que dins d'aquests tres sectors que s'inclouen dins de la classe, existeix la possibilitat de crear noves variables i funcions, que com ja hem dit seran de tipus *published*, *private* o *public*.

Amb açò i un poc més que resultaria prou extens per a l'objectiu d'aquest projecte, no resulta molt difícil ja endinsar-nos en l'entorn del Borland C++ Builder.

3.3. Aspectes importants del Borland C++ Builder

El Borland Builder, com a entorn d'aquest tipus de llenguatge, segueix amb aquesta filosofia i a més com ja s'ha dit amb anterioritat és un entorn de finestres, preparat per a suportar el sistema operatiu Windows.

A la figura 4.1 s'observa la finestra principal d'aquest llenguatge. On podem observar tres parts(finestres) ben diferenciades. Aquestes tres parts com podem llegir són:

- La finestra principal, on ens trobem amb tots els menús emergents propis dels programes que funcionen baix el sistema operatiu Windows. Aquests menús emergents són els típics File, Edit ... , i a més al ser un programa orientat a introduir de manera gràfica components, ens trobem amb els menús emergents Component, Tools a més dels dedicats a la producció i execució del programa escrit en llenguatge C++, com ara són: Project, Run, Options. A aquesta mateixa finestra podem trobar-se amb la paleta de components de la figura que es mostra a continuació, capaç de localitzar més ràpidament els components a introduir, així com la seua execució, l'obertura de nous programes, etc.
  Aquesta paleta de components és:

En ella podem veure els distints botons d'accés directe que suporta aquest entorn de programació.

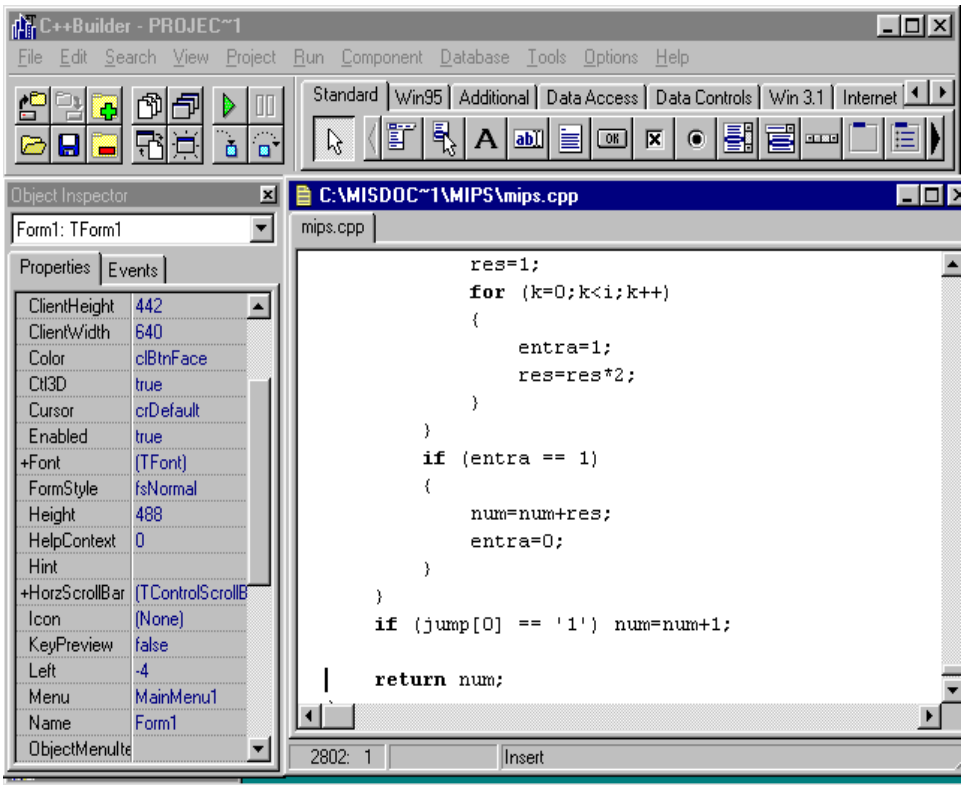
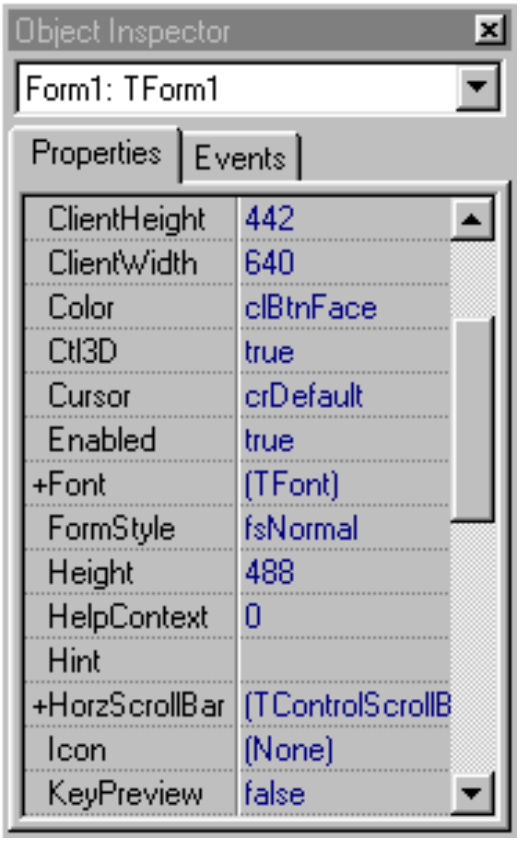


Figura 4.1.

- La segona d'aquestes finestres que apareixen inicialment al executar el programa, és l'**object inspector**, on ens trobem amb dues pestanyes, una en la que s'especifiquen les propietats de cada component, així com el color de fons, color de lletra, dimensions, posició, etc. Cal fer menció que les propietats es corresponen amb el component seleccionat, i l'altra de les pestanyes es correspon amb els **events** del component seleccionat, els events són els procediments que ha d'executar el programa en cas que s'active algun d'aquests mateix events. Possibles events d'un component, són ara, fer click en un dels components, un doble click, al escriure, intentar fer-lo més gran (cas de les finestres), etc.

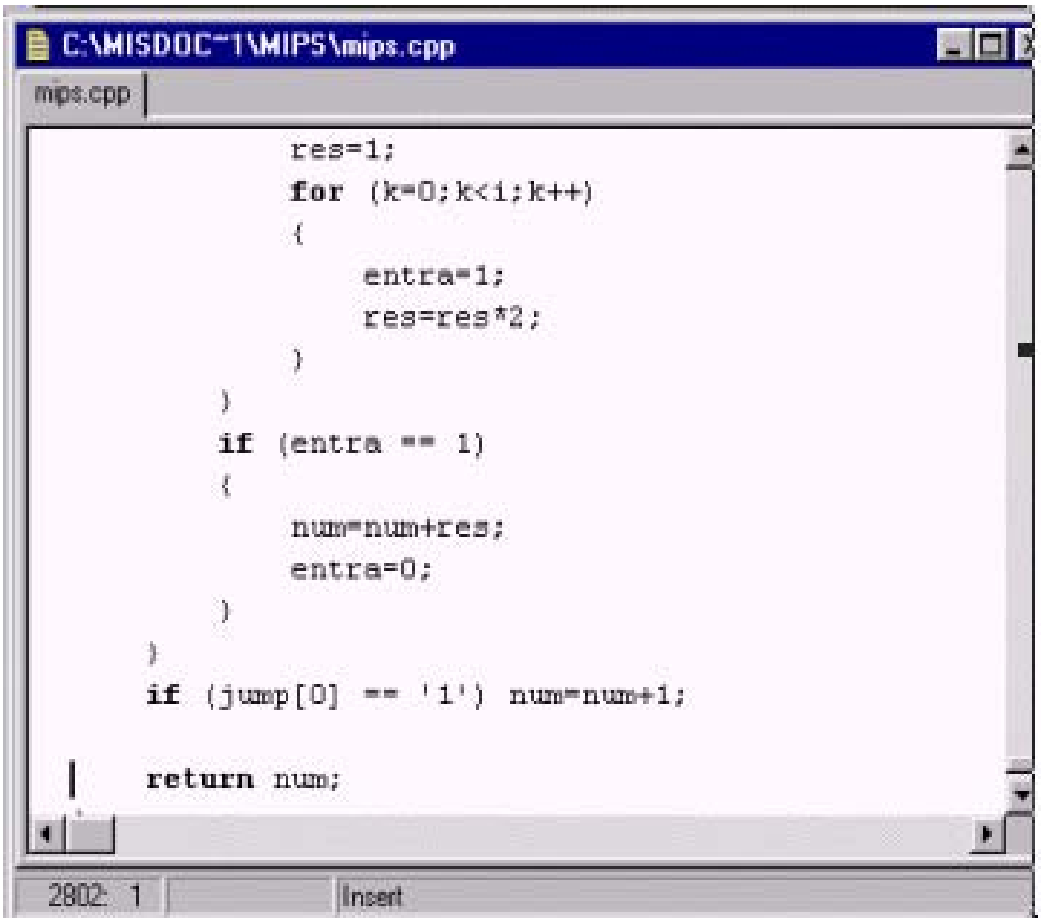
La forma de l'object inspector és mostrat a continuació:



Imatge de l'object inspector

- L'última d'aquestes finestres principals, es correspon , com és lògic al tractar-se d'un llenguatge de programació, a l'editor de programes, ací és on podem realitzar el codi dels nostres programes, no tan sols podem obrir els programes del projecte que es troba actualment en funcionament, sinó també, podem visualitzar-ne d'altres. L'únic que cal dir, és que al prémer la tecla per executar el nostre codi (compilar-lo i després runnejar-lo), el que farà el Borland C++ Builder és executar el projecte que es troba actualment en ús, no qualsevol programa editat a l'editor de programes.

  La finestra corresponent al editor de programes, la podem trobar a la figura la figura de la pàgina següent.



Una vegada explicat aquest inici del Borland Builder, es passa a explicar els diferents components utilitzats al llarg de la realització del present projecte.

Entre aquests components es troba **Form**: Paraula anglesa que podem traduir, per a que siga més comprensible, per fitxa, l'entorn Borland Builder crea per defecte una Form només començar, anomenada Form1, aquest objecte pot contindre'n d'altres que veure'm més endavant. Aquestes forms sintetitzen sempre la seua classe d'una manera automàtica simplement precedint el nom de la fitxa per la lletra T. Així per exemple si creem una fitxa amb el nom Form1, automàticament es crea la classe TForm1, editable des de la finestra de la figura anterior. Així que cada classe d'aquest tipus crea a més de la seua fitxa, una classe per poder tindre una sèrie de propietats, com són variables i funcions. L'aspecte d'una Form el trobem a la figura 4.2. Com veiem, aquesta fitxa és la que apareixerà després, a la execució del programa com a la típica finestra Windows.

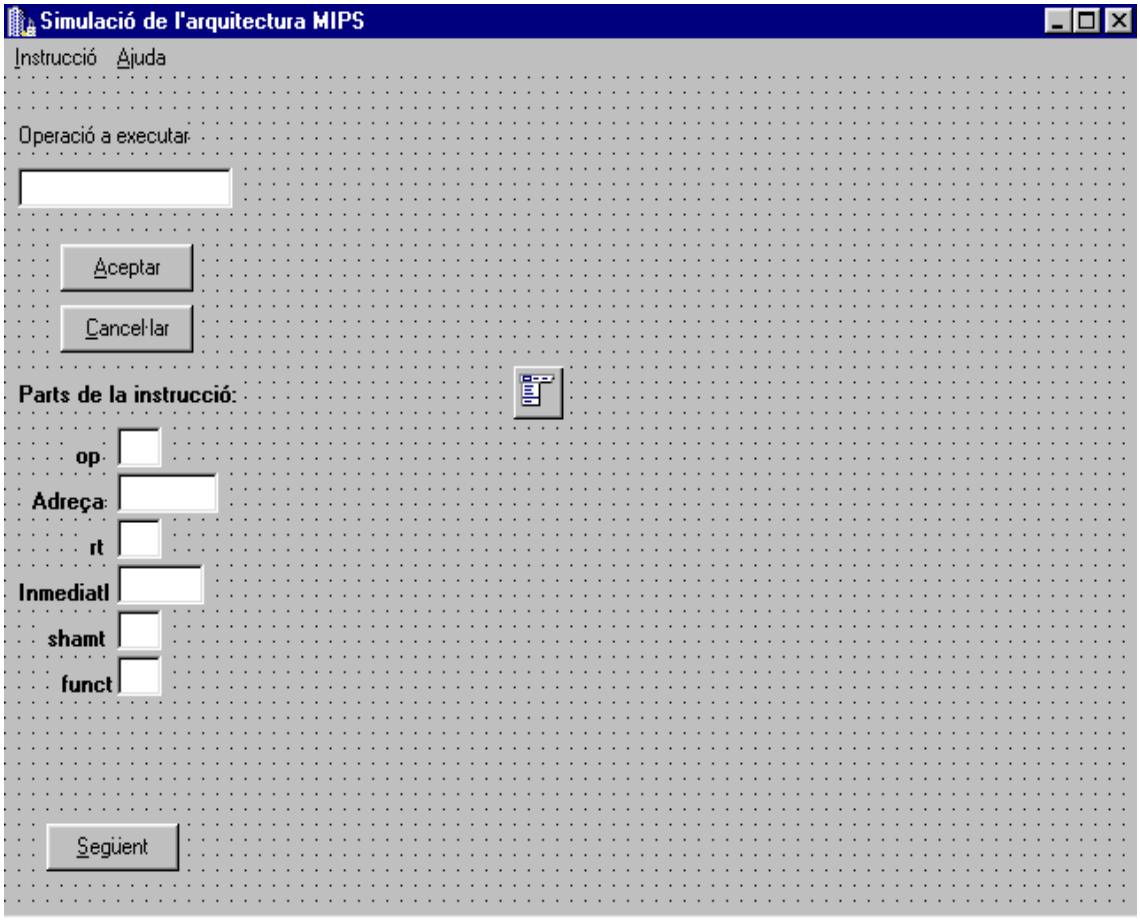


Figura 4.2. Aspecte d'una Form.

En aquesta figura podem observar que la fitxa conté diversos components, com es pot arribar a pensar aquests components formen part de la classe TForm1, de fet ho són, i es corresponen amb les variables declarades com a *published*, tal com es veu a continuació:

```
__published:
        TLabel *LInstruccio;
        TEdit *EInstruccio;
        TButton *BSeguent;
        TButton *BAceptar;
        TLabel *Label3;
        TLabel *Lop;
        TLabel *Lrs;
        TLabel *Lrt;
        TLabel *Lrd;
        TLabel *Lshamt;
        TLabel *Lfunct;
        TEdit *Eop;
```

```
TEdit *Ers;
TEdit *Ert;
TEdit *Erd;
TEdit *Eshamt;
TEdit *Efunct;
TLabel *LInmediat;
TEdit *EInmediat;
TLabel *LAdreca;
TEdit *EAdreca;
TButton *BCancelar;
TMainMenu *MainMenu1;
TMenuItem *Instr1;
TMenuItem *Registres1;
TMenuItem *Memoria1;
TMenuItem *N1;
TMenuItem *Eixir1;
TMenuItem *Novainstrucci1;
TMenuItem *N2;
```

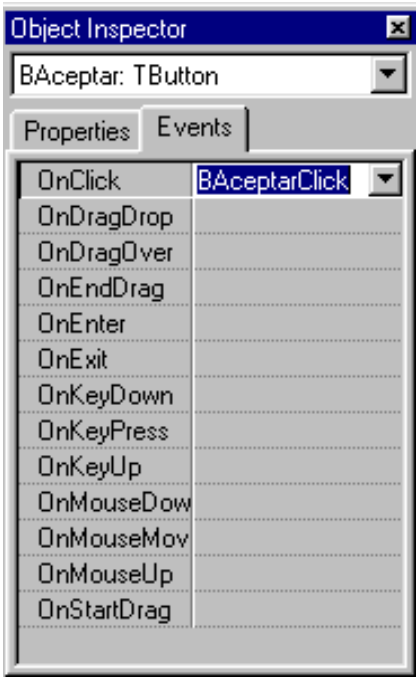
Com es veu, hi ha diversos tipus de declaracions d'aquestes variables, aquests tipus de variables com ara són: TLabel, TEdit, TButton, TMainMenu i TMenuItem. Aquestes variables no són més que noves classes amb una sèrie de propietats com les ja comentades.

Aquestes propietats, encara que són generalment diferents per a cada tipus d'objectes, (a partir d'ara anomenarem objecte a qualsevol classe ja definida, així li donarem sentit a la programació orientada a objectes abans comentada), com posteriorment es comentarà, entre alguns d'aquests objectes si que existeixen una sèrie de propietats que són comuns, com ara les propietats de *Visible, AutoSelect*, *Enabled* ... Aquestes propietats les podem observar a la figura anterior de l'**object inspector**.

A més d'aquestes propietats ja esmentades, com que el Borland Builder és un entorn de programació basat en finestres *Windows*, cada objecte dels que hi ha en aquest entorn tenen un subapartat anomenat *Events*. Aquests events no són més que funcions que es realitzen al executar una de les operacions de les que apareixen en aquest subapartat, com per exemple són: fer *click* damunt un objecte, doble *click*, polsar una tecla, etc.

Per ficar un exemple, podem assignar a un objecte molt comú, com ho és un botó, una funció definida, com potser tancar una finestra, validar les dades d'una finestra, etc.

Aquests events podem observar-los en la següent figura, que correspon amb la finestra de l'**object inspector** abans comentada, però ara oberta per a la pestanya *Events*.



Aquesta figura, mostra tots els events possibles que existeixen per al objecte de la classe TButton i que al programa com es veu s'anomena BAceptar. Com es veu, l'event al que es fa menció és a l'event OnClick, que com es pot pensar, es l'event associat a la polsada mitjançant el ratolí d'aquest objecte. Com es veu duu associat el concepte *BAceptarClick*, el qual es correspon amb el nom de la funció que s'executarà al realitzar aquest event.

Com es veu també es poden modificar més paràmetres dins d'aquesta finestra i que al realitzar-se, la funció associada al event s'executarà, igual que ocorre amb l'exemple de la figura d'abans.

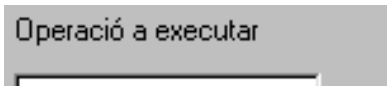
Per comentar alguns del objectes que formen part de l'entorn **Borland Builder**, es comentarà els que s'han utilitzat per a realitzar el present projecte:

- **MainMenu**, és el típic menú emergent de tots els programes que funcionen baix l'abast del sistema operatiu ***Windows***, el menú es desplega al fer *click* damunt d'algun dels components que el composen. L'aspecte general, és el següent.

L'aspecte quan es fa un click damunt d'un component dels que el formen, és el següent:

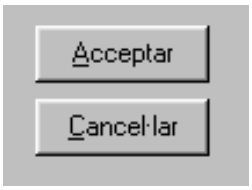
MainMenu amb una opció desplegada.

-Labels, són les etiquetes de text que podem visualitzar a l'execució del programa. L'ús més comú es el de nombrar a elements dins d'una fitxa. Gràcies a les propietats i als events podem realitzar canvis en temps d'execució de les etiquetes, a més dels canvis que es poden realitzar en temps de programació, que és l'ús més normal. La forma que tenen aquestes etiquetes és el següent:

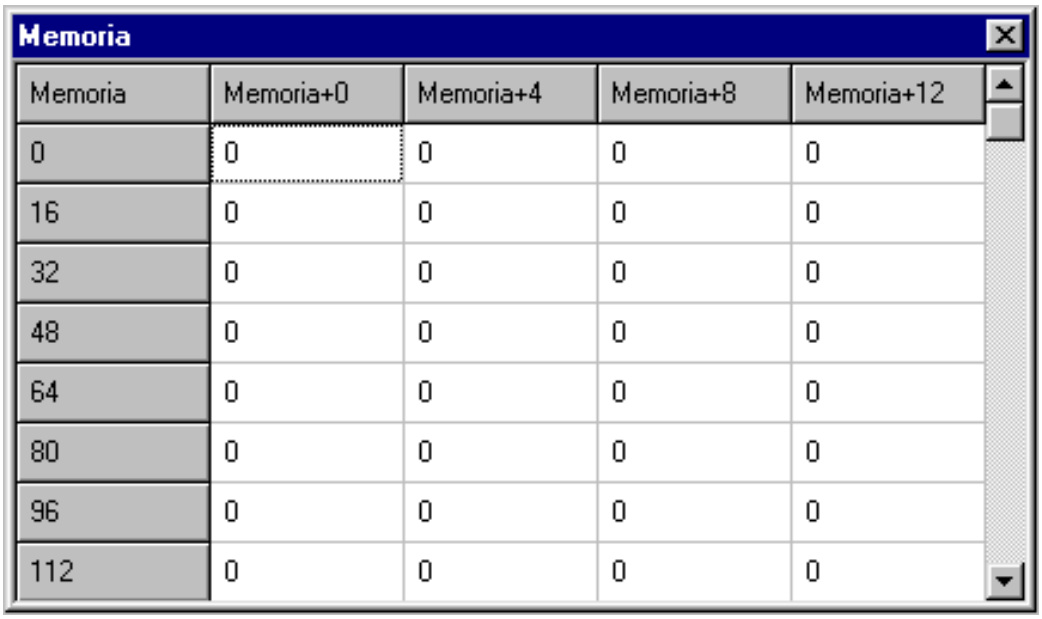


-Edits, són espais on en temps d'execució podem insertar un cert codi o una certa cadena de caràcters, que podem emmagatzemar i treballar amb ells, l'aspecte és el que segueix.

-Buttons, com el seu nom indica són botons, és clar que l'ús més corrent dels quals és l'execució d'un cert procediment en cas que es polse damunt ell, o millor dit, al fer un click damunt d'ell, s'activarà l'event corresponent a eixe botó. Podem veure'n un parell d'aquests components a la següent figura:



-DrawGrid, aquest component és una reixeta, que per al programa es tracta com una matriu, de tantes files i columnes com hi ha a la reixeta creada. Al projecte s'ha utilitzat com a taula que simula la memòria, com podem veure a la figura que segueix, la qual es troba dins d'una nova fitxa:



Feta aquesta valoració del llenguatge C++ i de l'entorn Borland Builder, passarem al quart capítol, en el qual es realitza una avaluació més exhaustiva de les parts del codi del programa més interessants.