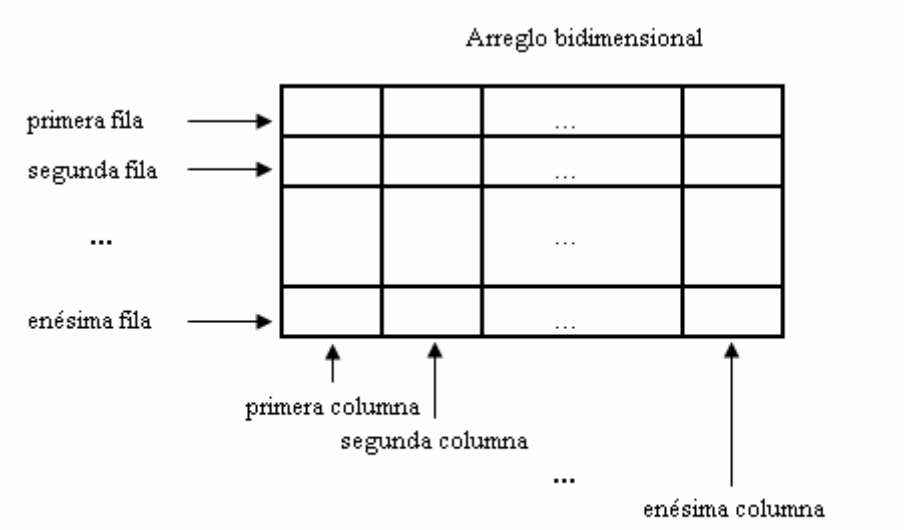
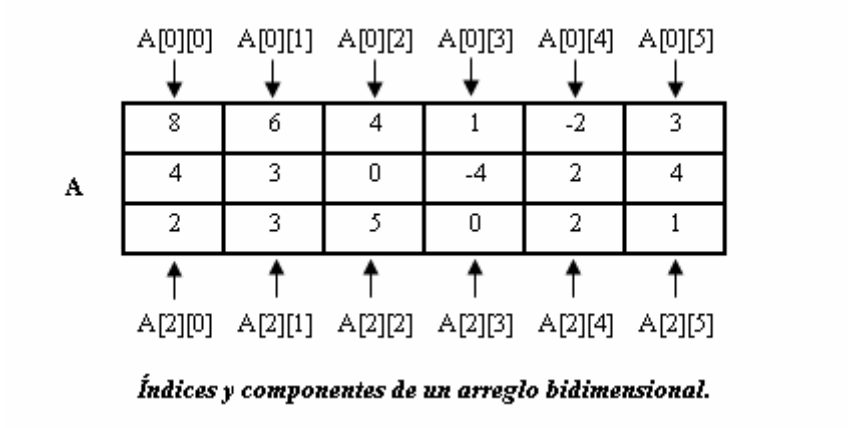
Matrices

Son arreglos bidimensionales, es una colección finita, homogénea y ordenada de datos. Una matriz esta compuesta por filas y columnas, en la que se hace referencia a cada elemento por medio de dos índices. El primero de los índices se utiliza para indicar la fila y el segundo de los índices para indicar la columna.



Representación gráfica de un arreglo bidimensional.



Índices y componentes de un arreglo bidimensional.

Declaración de una matriz

tipo_de_dato identificador[numFilas][numColumnas];;

Dónde :

tipo_de_dato: Es el tipo de datos que contendrá la matriz.

identificador: Es el nombre que le damos a la variable matriz y por el cual la referenciaremos en nuestro programa.

[numFilas][numColumnas] : Especifica el numero de Filas y de columnas que tendrá la matriz

El espacio que las matrices ocupan en memoria se reserva en el momento de realizar la declaración de los mismos.
Ejemplo:

```
int  A[3][4];    // Declaración de una matriz de enteros de 3 filas y 4 columnas
float B[6][2];  // Declaración de una matriz de reales de 6 filas y 2 columnas
char C[4][10]; // Declamación de una matriz de caracteres de 4 filas y 10 columnas
```

Inicializacion de una Matriz

Una matriz se pude inicializar para esto hay que agrupar entre {} cada fila. El formato a utilizar sería el siguiente:

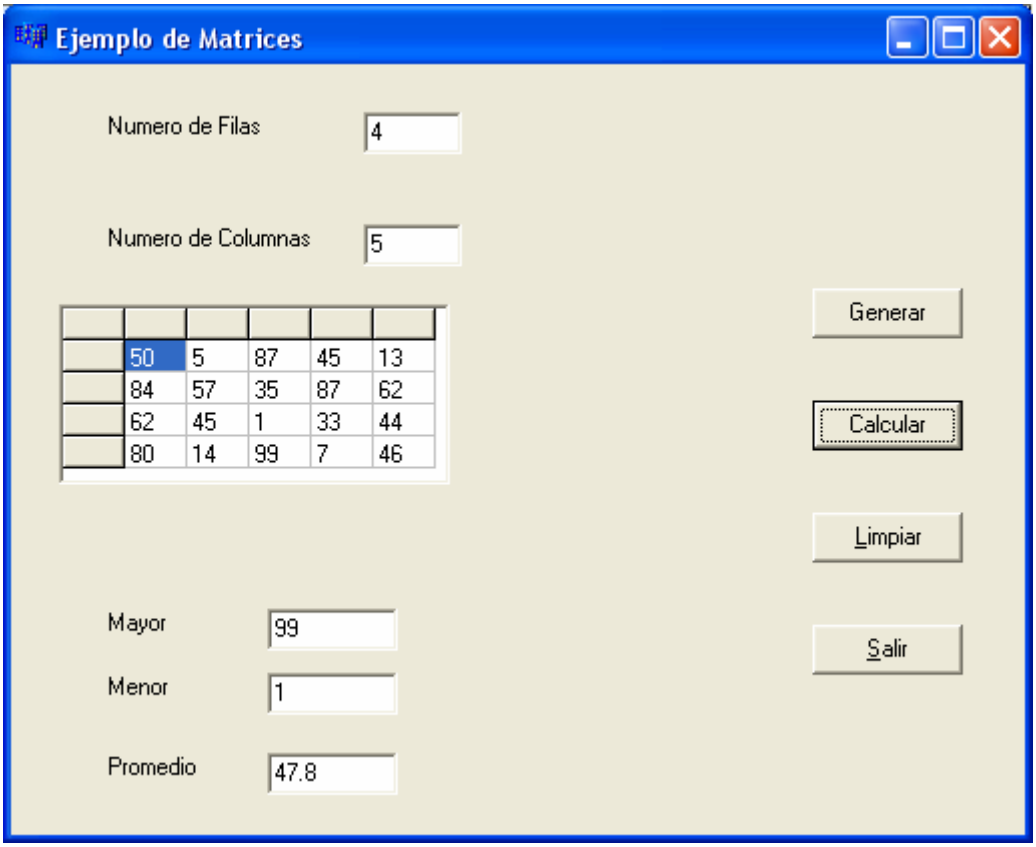
```
tipo_de_dato identificador[ filas ][ columnas ] = {
        { columnas de la fila 1 },
        { columnas de la fila 2 },
        ... ,
        { columnas de la última fila }
        };
```

No debemos olvidar el ';' al final.
Ejemplo:

```
int temperaturas[3][5] = { { 15, 17, 20, 25, 10 }, { 18, 20, 21, 23, 18 }, { 12, 17, 23, 29, 16 } };
```

Ejercicios Resueltos de Matrices

1) Hacer un programa para generar una matriz de f filas y c columnas y calcular el mayor, el menor y el promedio.



Archivo matrices.h

```
//---------------------------------------------------------------------------
#ifndef matricesH
#define matricesH
//---------------------------------------------------------------------------
#endif
#include<stdlib.h>
#include<grids.hpp>

const int MAX=20;
void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd);
void mostrar(float A[MAX][MAX], int f, int c, TStringGrid *grd);
float mayor(float M[MAX][MAX], int f, int c);
float menor(float M[MAX][MAX], int f, int c);
float promedio(float M[MAX][MAX], int f, int c);
void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac);
```

Archivo matrices.cpp

```
//---------------------------------------------------------------------------

#pragma hdrstop
```

```
#include "matrices.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)


void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac)
{
  grd->RowCount=f+1;
  grd->ColCount=c+1;
  grd->DefaultColWidth=ac;
  grd->DefaultRowHeight=af;
  grd->Width=(c+1)*grd->DefaultColWidth+15;
  grd->Height=(f+1)*grd->DefaultRowHeight+15;

}

void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
    {
       M[i][j]=random(100)+1;
       grd->Cells[j+1][i+1]=M[i][j];
    }
}

void mostrar(float A[MAX][MAX], int f, int c, TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
      grd->Cells[j+1][i+1]=A[i][j];
}

float mayor(float M[MAX][MAX], int f, int c)
{
  int i,j;
  float may=M[0][0];
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
    {
      if(M[i][j]>may)
         may=M[i][j];
    }
  return may;
}

float menor(float M[MAX][MAX], int f, int c)
{
  int i,j;
  float men=M[0][0];
```

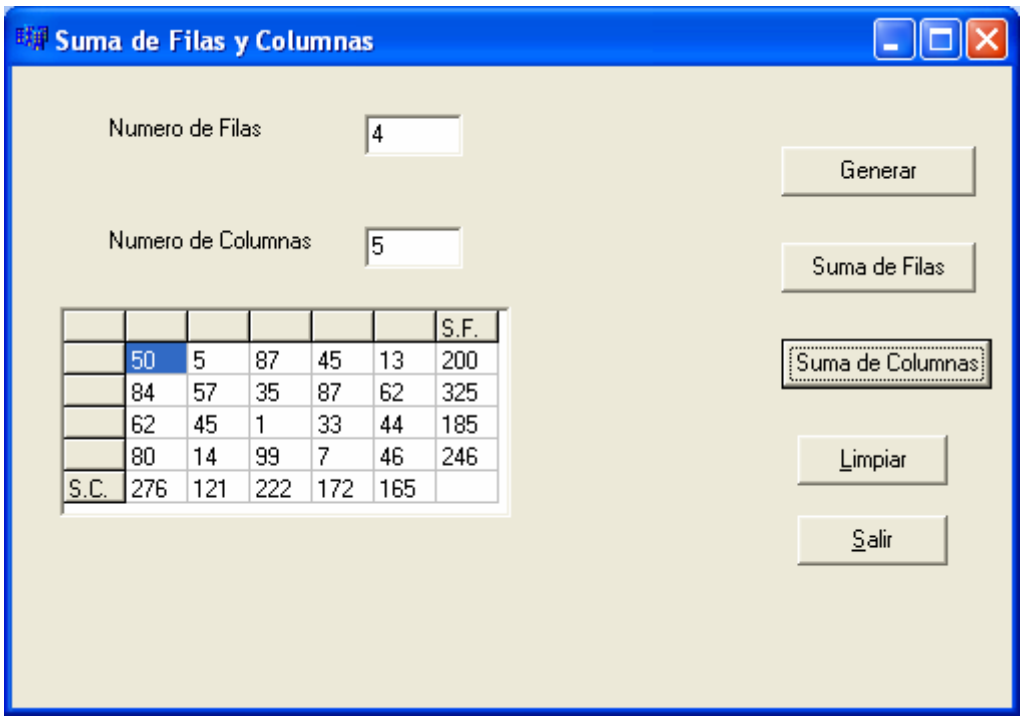
```
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
    {
      if(M[i][j]<men)
        men=M[i][j];
    }
  return men;

}

float promedio(float M[MAX][MAX], int f, int c)
{
  int i,j;
  float s=0;
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
      s=s+M[i][j];
  return s/(f*c);
}
```

Archivo Unit1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "matrices.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float A[MAX][MAX];
int f, c;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnGenerarClick(TObject *Sender)
{
  f=EdF->Text.ToInt();
  c=EdC->Text.ToInt();
  generaMatriz(A,f,c,grdNum);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  EdMayor->Text=mayor(A,f,c);
  EdMenor->Text=menor(A,f,c);
  EdPromedio->Text=promedio(A,f,c);
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  EdF->Clear();
  EdC->Clear();
  preparaGrid(grdNum,1,1,15,30);
  EdMayor->Clear();
  EdMenor->Clear();
  EdPromedio->Clear();
  grdNum->Cells[1][1]="";
  EdF->SetFocus();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
```

2) Ingresar una matiz de f filas y c columnas y calcular la suma de filas y la suma de columnas



Unit2.h

```
#include<stdlib.h>
#include<grids.hpp>

const int MAX=20;
void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac);
void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd);
void sumaDeFilas(float A[MAX][MAX], int f, int c,float sf[MAX]);
```

```
void sumaDeColumnas(float A[MAX][MAX], int f, int c,float sc[MAX]);
void mostrarSumaDeFilas(float sf[], int f,int c,TStringGrid *grd);
void mostrarSumaDeColumnas(float sc[], int f,int c,TStringGrid *grd);
```

Unit2.cpp

```
//---------------------------------------------------------------------------


#pragma hdrstop

#include "Unit2.h"

//---------------------------------------------------------------------------

#pragma package(smart_init)

void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac)
{
  grd->RowCount=f+1;
  grd->ColCount=c+1;
  grd->DefaultColWidth=ac;
  grd->DefaultRowHeight=af;
  grd->Width=(c+1)*grd->DefaultColWidth+15;
  grd->Height=(f+1)*grd->DefaultRowHeight+15;

}

void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
    {
      M[i][j]=random(100)+1;
      grd->Cells[j+1][i+1]=M[i][j];
    }
}


void sumaDeFilas(float A[MAX][MAX], int f, int c,float sf[MAX])
{
  int i,j;
  for(i=0;i<f;i++)
  {
    sf[i]=0;
    for(j=0;j<c;j++)
      sf[i]=sf[i]+A[i][j];
  }
}

void sumaDeColumnas(float A[MAX][MAX], int f, int c,float sc[MAX])
{
  int i,j;
  for(j=0;j<c;j++)
```

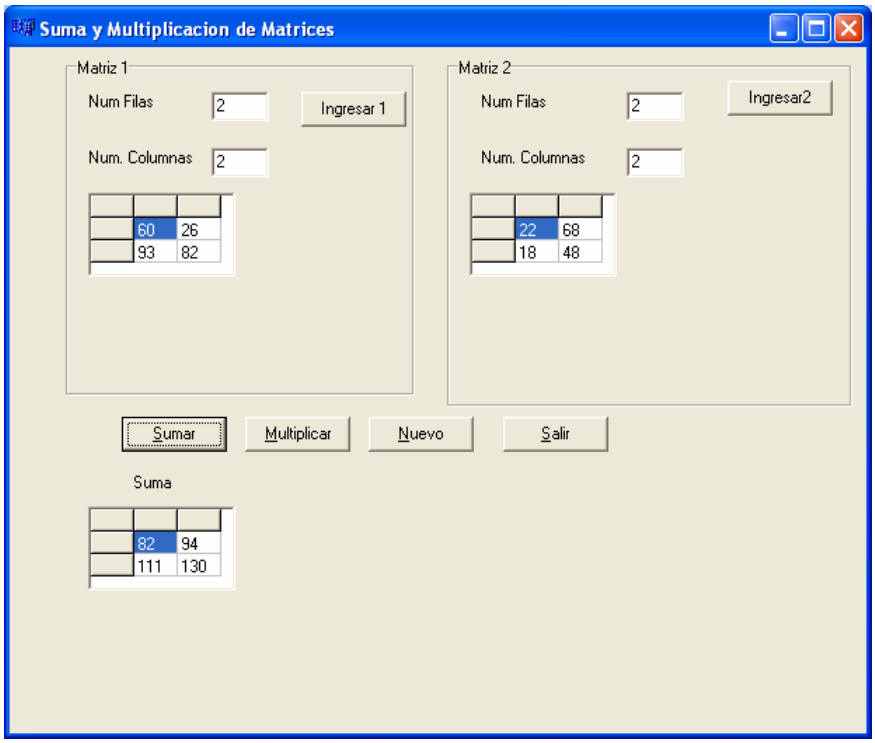
```
  {
    sc[j]=0;
    for(i=0;i<f;i++)
     sc[j]=sc[j]+A[i][j];
  }
}

void mostrarSumaDeFilas(float sf[], int f,int c,TStringGrid *grd)
{
   int i;
   grd->ColCount=grd->ColCount+1;
   grd->Cells[c+1][0]="S.F.";
   grd->Width=grd->Width+grd->DefaultColWidth;
   for(i=0;i<f;i++)
    grd->Cells[c+1][i+1]=sf[i];
}

void mostrarSumaDeColumnas(float sc[], int f,int c,TStringGrid *grd)
{
   int j;
   grd->RowCount=grd->RowCount+1;
   grd->Cells[0][f+1]="S.C.";
   grd->Height=grd->Height+grd->DefaultRowHeight;
   for(j=0;j<c;j++)
    grd->Cells[j+1][f+1]=sc[j];
}
```

Unit1.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float A[MAX][MAX],sf[MAX],sc[MAX];
int f, c;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnGenerarClick(TObject *Sender)
{
  f=EdF->Text.ToInt();
  c=EdC->Text.ToInt();
  generaMatriz(A,f,c,grdNum);

}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  sumaDeFilas(A,f,c,sf);
  mostrarSumaDeFilas(sf,f,c,grdNum);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  sumaDeColumnas(A,f,c,sc);
  mostrarSumaDeColumnas(sc,f,c,grdNum);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
   EdF->Clear();
   EdC->Clear();
   preparaGrid(grdNum,1,1,15,30);
   grdNum->Cells[1][1]="";
   EdF->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

3) Programa para ingresar dos matrices, una de f1 filas y c1 columnas y otra de f2 filas y c2 columnas y reportar su suma y su producto si es que se pueden realizar.



Archivo Unit2.h

```
//---------------------------------------------------------------------------

#ifndef Unit2H
#define Unit2H
//---------------------------------------------------------------------------
#endif
#include<stdlib.h>
#include<grids.hpp>

const int MAX=20;
void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac);
void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd);
void mostrar(float A[MAX][MAX], int f, int c, TStringGrid *grd);
void sumaMatrices(float A[MAX][MAX],float B[MAX][MAX],float C[MAX][MAX], int f, int c);
void productoMatrices(float A[MAX][MAX],float B[MAX][MAX],float P[MAX][MAX], int f1, int
c1,int c2);
```

Archivo Unit2.cpp

```
//---------------------------------------------------------------------------
#pragma hdrstop
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac)
{
  grd->RowCount=f+1;
  grd->ColCount=c+1;
  grd->DefaultColWidth=ac;
  grd->DefaultRowHeight=af;
  grd->Width=(c+1)*grd->DefaultColWidth+15;
  grd->Height=(f+1)*grd->DefaultRowHeight+15;
}

void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
    {
       M[i][j]=random(100)+1;
       grd->Cells[j+1][i+1]=M[i][j];
    }
}

void mostrar(float A[MAX][MAX], int f, int c, TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
      grd->Cells[j+1][i+1]=A[i][j];
}

void sumaMatrices(float A[MAX][MAX],float B[MAX][MAX],float S[MAX][MAX], int f, int c)
```

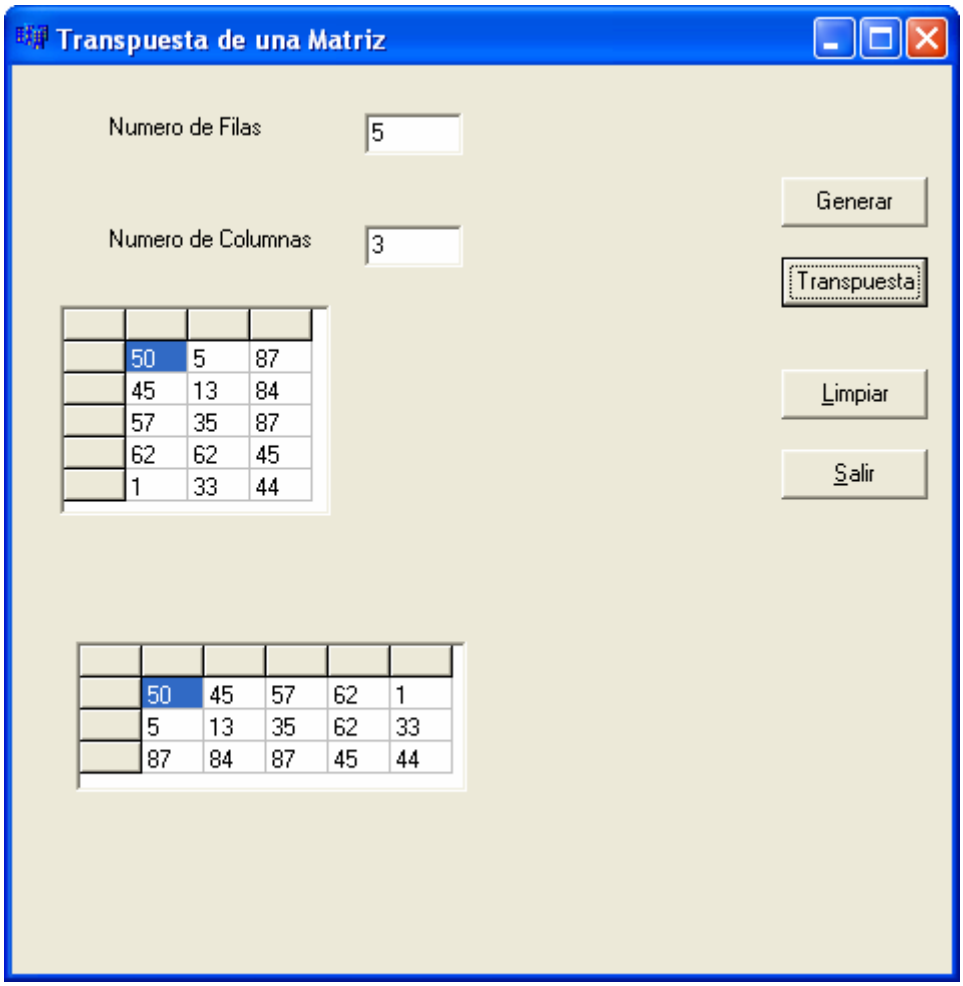
```
{
  int i,j;
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
      S[i][j]=A[i][j]+B[i][j];

}
void productoMatrices(float A[MAX][MAX],float B[MAX][MAX],float P[MAX][MAX], int f1, int
c1,int c2)
{
  int i,j,k;
  for(i=0;i<f1;i++)
    for(j=0;j<c2;j++)
    {
      P[i][j]=0;
      for(k=0;k<c1;k++)
        P[i][j]=P[i][j]+A[i][k]*B[k][j];
    }
}
```

Archivo Unit1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float A[MAX][MAX],B[MAX][MAX],S[MAX][MAX],P[MAX][MAX];
int f1,c1,f2,c2;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnIngreso1Click(TObject *Sender)
{
  f1=edF1->Text.ToInt();
  c1=edC1->Text.ToInt();
  generaMatriz(A,f1,c1,grdA);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnIngresar2Click(TObject *Sender)
{
  f2=edF2->Text.ToInt();
  c2=edC2->Text.ToInt();
  generaMatriz(B,f2,c2,grdB);
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSumarClick(TObject *Sender)
{
  if(f1==f2 && c1==c2)
  {
     lblR->Caption="Suma";
     sumaMatrices(A,B,S,f1,c1);
     mostrar(S,f1,c1,grdR);
  }
  else
      ShowMessage("No se puede Sumarse las matrices");

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnMultiplicarClick(TObject *Sender)
{
  if(c1==f2)
  {
    lblR->Caption="Producto";
    productoMatrices(A,B,P,f1,c1,c2);
    mostrar(P,f1,c2,grdR);
  }
  else
     ShowMessage("No se pueden Multiplicar las matrices");

}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnNuevoClick(TObject *Sender)
{
  edF1->Clear();
  edC1->Clear();
  edF2->Clear();
  edC2->Clear();
  preparaGrid(grdA,1,1,15,30);
  grdA->Cells[1][1]="";
  preparaGrid(grdB,1,1,15,30);
  grdB->Cells[1][1]="";
  preparaGrid(grdR,1,1,15,30);
  grdR->Cells[1][1]="";
  edF1->SetFocus();
}
//---------------------------------------------------------------------------
```

4) Ingresar una matriz de numeros de f filas y c columnas y calcular su matriz transpuesta.



Archivo unit2.h

```
#include<stdlib.h>
#include<grids.hpp>

const int MAX=20;
void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac);
void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd);
void mostrar(float A[MAX][MAX], int f, int c, TStringGrid *grd);
void transpuesta(float A[MAX][MAX], int f, int c,float T[MAX][MAX]);
```

Archivo unit2.cpp

```
//---------------------------------------------------------------------------


#pragma hdrstop

#include "Unit2.h"

//---------------------------------------------------------------------------

#pragma package(smart_init)

void preparaGrid(TStringGrid *grd,int f,int c, int af, int ac)
{
```

```
  grd->RowCount=f+1;
  grd->ColCount=c+1;
  grd->DefaultColWidth=ac;
  grd->DefaultRowHeight=af;
  grd->Width=(c+1)*grd->DefaultColWidth+15;
  grd->Height=(f+1)*grd->DefaultRowHeight+15;

}

void generaMatriz(float M[MAX][MAX], int f, int c,TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
    {
       M[i][j]=random(100)+1;
       grd->Cells[j+1][i+1]=M[i][j];
    }
}

void mostrar(float A[MAX][MAX], int f, int c, TStringGrid *grd)
{
  int i,j;
  preparaGrid(grd,f,c,15,30);
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
      grd->Cells[j+1][i+1]=A[i][j];
}


void transpuesta(float A[MAX][MAX], int f, int c,float T[MAX][MAX])
{
  int i,j;
  for(i=0;i<f;i++)
    for(j=0;j<c;j++)
       T[j][i]=A[i][j];
}
```

Archivo Unit1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float A[MAX][MAX],T[MAX][MAX];
int f, c;
```

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnGenerarClick(TObject *Sender)
{
  f=EdF->Text.ToInt();
  c=EdC->Text.ToInt();
  generaMatriz(A,f,c,grdNum);

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnTranspuestaClick(TObject *Sender)
{
    transpuesta(A,f,c,T);
    mostrar(T,c,f,grdT);


}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
   EdF->Clear();
   EdC->Clear();
   preparaGrid(grdNum,1,1,15,30);
   grdNum->Cells[1][1]="";
   preparaGrid(grdT,1,1,15,30);
   grdT->Cells[1][1]="";
   EdF->SetFocus();


}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnSalirClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

Ejercicios Propuestos

1) Ingresar una matriz cuadrada de orden n y reportar si es simétrica. Recordar que una matriz es simétrica si se cumple la condición: a[i][j]=a[j][i]

2) Programa para ingresar una matriz de f filas y c columnas, y que se haga lo siguiente:
a) Ingresar un número de fila y eliminarla de la matriz.
b) Ingresar un número de columna y eliminarla de la matriz.
c) Ingresar un número de fila e insertar una fila en la matriz.
d) Ingresar un número de columna e insertar una columna en la matriz.
e) Intercambiar 2 filas de la Matriz. El número de las filas a intercambiar debe ingresarse.
f) Intercambiar 2 columnas de la Matriz. El número de las columnas a intercambiar debe ingresarse.
g) Ordenar las filas de una matriz
h) Ordenar las columnas de una matriz

3) El curso de Computación tiene n alumnos y se rinden 4 exámenes. Escribir un programa que reporte lo siguiente:
a) El promedio de calificaciones de cada alumno.
b) El promedio de cada Examen
c) El alumno que obtuvo el mejor Promedio
d) El examen que tuvo el mayor promedio de calificación.

4) Hacer un programa para invertir las columnas de una matriz (Los elementos de la primera columna se intercambian con los de la ultima, los de la segunda con los de la penúltima y así sucesivamente).

5) Escribir un programa que genere un cuadrado mágico. Un cuadrado mágico se representa por una matriz cuadrada de orden n, impar y contiene los números comprendidos entre 1 y n*n. En un cuadrado mágico la suma de cualquiera de las filas, columnas y diagonales principales siempre es la misma. El cuadrado mágico se genera aplicando el siguiente algoritmo:
a) El primer numero 1 se coloca en la celda central de la primera fila.
b) El siguiente número se coloca en la celda de la fila anterior y columna posterior.
c) La fila anterior al primero es el último. La columna posterior a la última es la primera.
d) Si el número es un sucesor de un múltiplo de n, no aplique la regla 2. Coloque el número en la celda de la misma columna de la fila posterior.

6) Hacer un programa para que coloque un 1 en las diagonales principales de una matriz cuadrada. El resto se debe completar con ceros.

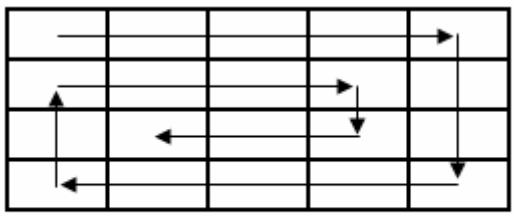
7) Hacer un programa que, al recibir los montos de ventas mensuales de cinco departamentos de una fábrica proporcione la siguiente información

a) Las ventas mensuales de la fábrica incluido el monto anual.
b) El departamento que tuvo la mayor venta en el mes de Julio, incluyendo el monto de la venta.
c) El mes en el que se obtuvieron las mayores y menores ventas del departamento I, donde I se debe ingresar.

8) Hacer un programa para invertir las filas de una matriz (Los elementos de la primera fila se intercambian con los de la ultima, los de la segunda con los de la penúltima y así sucesivamente).

9) Hacer un programa que al recibir una matriz cuadrada de orden impar. Determine si la misma se pude considerar un cuadrado Mágico.(En un cuadrado mágico la suma de cualquiera de las filas, columnas y diagonales principales siempre es la misma).

10) Hacer un programa que al recibir como dato una matriz , recorra esta matriz en forma de espiral. Tal como se muestra en la figura:



Recorrido en forma espiral en una matriz.

11) Hacer un programa que al recibir como dato una matriz recorra esta matriz columna a columna tal como se muestra en la figura.

12) Programa que ingresa el orden de una Matriz cuadrada y generarla y luego hacer lo siguiente:
a) Calcula la suma de los elementos de la diagonal principal.
b) Calcula el promedio de los elementos de la diagonal secundaria.
c) Calcula el mayor de los elementos de la matriz triangular inferior.
d) Calcula el promedio de los elementos de la matriz triangular superior.
e) Reporta solo las diagonales.
f) Intercambia las diagonales.
g) Invierte las diagonales.
h) Reporta los elementos que están arriba y abajo de la diagonal principal.
i) Reporta los elementos que están arriba y abajo de la diagonal secundaria.

13) Programa para ingresar una matriz de números enteros diferentes de cero de filas y c columnas y la acomode para que queden primero los números positivos y luego los números negativos

14) Ingresar una matriz de f filas y c columnas y calcular el Mayor, y reportar todas las posiciones donde se encuentra el mayor. Y calcular el menor y reportar todas las posiciones donde se encuentra el menor

15) Ingresar una matriz cuadrada y verificar si es una matriz Triangular inferior.