Arreglos

Un arreglo (array) es una colección de datos del mismo tipo, que se almacenan en posiciones consecutivas de memoria y reciben un nombre común. Para referirse a un determinado elemento de un array se deberá utilizar un índice, que especifique su posición relativa en el array. El índice de un arreglo comienza en 0.

Un arreglo es una colección finita, homogénea y ordenada de elementos. Finita: Todo arreglo tiene un límite; es decir, debe determinarse cuál será el número máximo de elementos que podrán formar parte del arreglo. Homogénea: Todos los elementos del arreglo deben ser del mismo tipo. Ordenada: Se puede determinar cuál es el primer elemento, el segundo, el tercero,.... y el n-ésimo elmento.

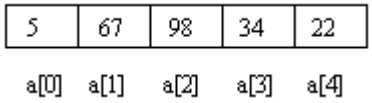
Los arreglos se clasifican de acuerdo con el número de dimensiones que tienen. Así se tienen los:

- Unidimensionales (vectores)
- Bidimensionales (tablas o matrices)
- Multidimensionales (tres o más dimensiones)

Vectores

La Declaración de un Arreglo es igual a como se haría con una variable, a excepción de que también se especifica la cantidad de elementos en el arreglo encerrado entre corchetes de la siguiente manera:

Tipo_de_dato nombre_del_arreglo[tamaño];



Ejemplo:

```
int a[20]; // declara un vector de 20 elementos enteros

flota x[50]; // declara un vector de 50 elementos reales

char w[30]; // declara un vector de 30 elementos tipo carácter

char nombres[50][80]; // declara un vector de 50 elementos donde cada elemento
                      // es una cadena de 80 caracteres
```

Inicialización de un Vector

Cuando se declara un arreglo, sus valores se pueden inicializar de la siguiente manera:

int lista[9]= {0, 4, 78, 5, 32, 9, 77, 1, 23}

Una Característica importante de los arreglos en C es que no se pueden modificar los limites superior e inferior (y por tanto el rango) durante el programa. El limite inferior se fija siempre en 0 y el superior lo fija el programador, es decir:

```
Int lista[9]= {0, 4, 78,  5,  32,  9, 77, 1, 23}
Posición →  0  1  2  3    4  5   6  7  8 = (9 posiciones)
```

Acceso a los elementos de un arreglo

Cada valor dentro de un arreglo se conoce como *elemento del arreglo*. Para acceder a un elemento de un arreglo especifique el nombre del arreglo con el índice del elemento entre corchetes [].

Ejemplo:

```
int números = { 12, 20, 60, 80, 100 };

for(int  índice = 0; índice < 5 ; índice++)
        cout<<números[índice]<<endl;
```

El índice comienza en 0 y termina en 4 porque hay cinco elementos, y cada elemento del arreglo se llama números[índice].

Ejercicios de Vectores

1) Hacer un programa para generar n números aleatoriamente en un Vector y calcular el mayor, menor y el promedio.

En la unidad llamada vector.h se declaran las funciones para generar el vector, calcular el promedio, el mayor y el menor

Unidad arreglos.h

```
#ifndef arreglosH
#define arreglosH
//---------------------------------
#endif
void generaVector(float x[], int n,TListBox *lst);
float mayor(float x[], int n);
float menor(float x[], int n);
float promedio(float x[], int n);
//---------------------------------------------------------------------------
```

Archivo arreglos.cpp

```
#include <vcl.h>
#pragma hdrstop
#include "arreglos.h"
#pragma package(smart_init)

void generaVector(float x[], int n,TListBox *lst)
{
  int i;
  for(i=0;i<n;i++)
  {
      x[i]=random(100)+1;
      lst->Items->Add(x[i]);
  }
}
```

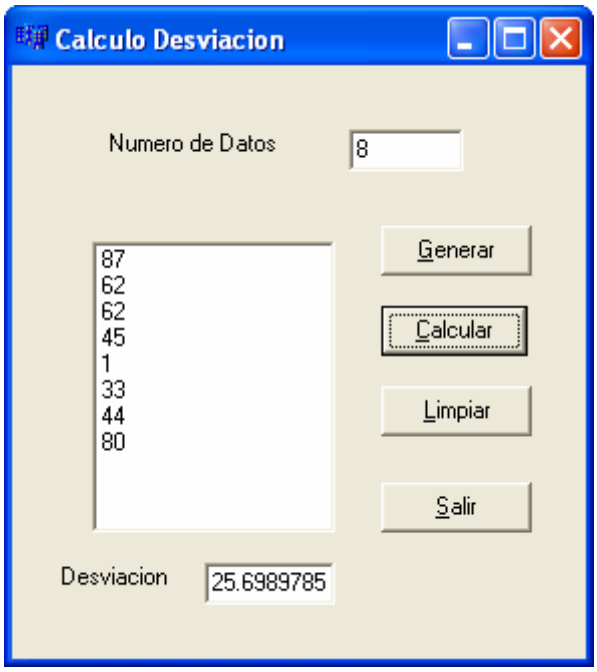
```
float mayor(float x[], int n)
{
  int i;
  float may;
  may=x[0];
  for(i=0;i<n;i++)
  {
    if(x[i]>may)
      may=x[i];
  }
  return may;
}

float menor(float x[], int n)
{
  int i;
  float men;
  men=x[0];
  for(i=0;i<n;i++)
  {
    if(x[i]<men)
      men=x[i];
  }
  return men;
}
float promedio(float x[], int n)
{
  int i;
  float s=0;
  for(i=0;i<n;i++)
    s=s+x[i];
  return s/n;
}
```

En la unidad unit1.cpp donde se encuentra el formulario, llamaremos a las funciones declaradas en la unidad vector.h

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

#include "Unit1.h"
#include "arreglos.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float x[100];
int n;

__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
```

```
void __fastcall TForm1::btnGenerarClick(TObject *Sender)
{
   n=edN->Text.ToInt();
   generaVector(x,n,lstN);
}


void __fastcall TForm1::btnCalcularClick(TObject *Sender)
{
  edMay->Text=mayor(x,n);
  edMen->Text=menor(x,n);
  edProm->Text=promedio(x,n);

}

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
 edN->Clear();
 lstN->Items->Clear();
 edMay->Clear();
 edMen->Clear();
 edProm->Clear();
 edN->SetFocus();
}

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

2) Programa para generar n números enteros aleatoriamente y calcular la desviación standard.



En este ejercicio usamos la unidad vector.h que se creo en el ejercicio anterior.

Para añadirla en el Proyecto se hace lo siguiente:
a) Clic en Project
b) Clic en add To Project (o shift+F11)
c) Se escoge la unidad que se quiere agregar al Proyecto, y luego clic en Abrir.

Después para usarla en el Proyecto

1) Hacer clic en File
2) Hacer clic en Include Unit Hdr

Y escogemos la unidad en este caso vector.h

Tenemos los archivos unit2.h que contiene la cabecera de la funcion desviacionStandard.

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

#ifndef Unit2H
#define Unit2H
//---------------------------------------------------------------------------
#endif
#include<math.h>

float desviacionStandard(float x[], int n);
```

El archive unit2.cpp

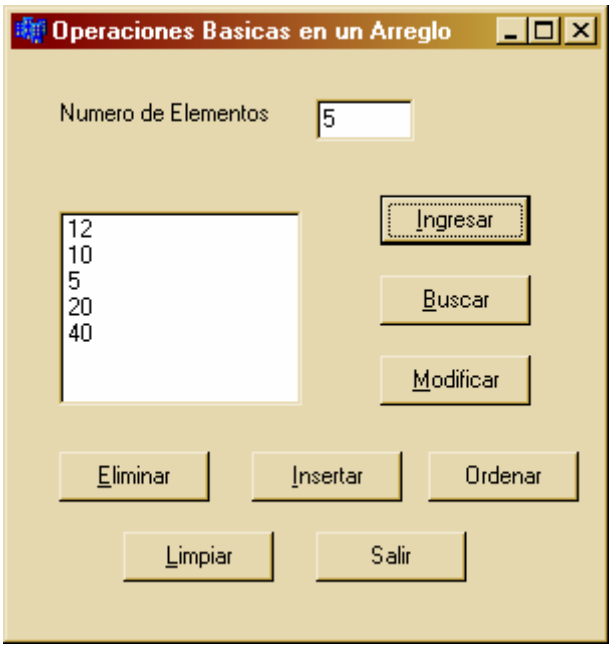
```
//---------------------------------------------------------------------------

#pragma hdrstop

#include "Unit2.h"
#include "vector.h"

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

#pragma package(smart_init)

float desviacionStandard(float x[], int n)
{
  int i;
  float s=0,p;
  p=promedio(x,n);
  for(i=0;i<n;i++)
    s=s+pow(x[i]-p,2);
  return sqrt(s/n);
}
```

Ademas el archive unit1.cpp donde esta el formulario, En este archivo se tiene que incluir a las unidasdes vector.h y unit2.h

```
#include <vcl.h>
#include<math.h>
#pragma hdrstop

#include "Unit1.h"
#include "arreglos.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TForm1 *Form1;
float x[100];
int n;

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

void __fastcall TForm1::btnGenerarClick(TObject *Sender)
{
  n=edN->Text.ToInt();
  generaVector(x,n,lstN);
}

void __fastcall TForm1::Button3Click(TObject *Sender)
{
   Close();
}

void __fastcall TForm1::Button1Click(TObject *Sender)
{
    edDesv->Text=desviacionStandard(x,n);
}

void __fastcall TForm1::Button2Click(TObject *Sender)
{
   edN->Clear();
   lstN->Clear();
   edDesv->Clear();
   edN->SetFocus();
}
```

3) Programa para ingresar n elementos en un vector y permita mostrar los elementos y permita:
    a) Dado un elemento buscarlo en el vector, si este se encuentra mostrar su posición, si el elemento no se encuentra reportar dato no se encuentra.
    b) Dado un elemento eliminarlo del vector si es que se encuentra en caso contrario reportar dato no se encuentra,
    c) Dato un elemento modificarlo, es decir si este se encuentra pedir un nuevo elemento y colocar este nuevo en la posición donde estaba el anterior, sino se encuentra mostrar dato no se encuentra
    d) Insertar un elemento y la posición donde se desea insertar y luego insertarlo en el vector, Si la posición no es correcta mostrar Mensaje indicando posición incorrecta.
    e) Ordenar los elementos de menor a mayor.



Archivo unit2.h

```
//---------------------------------------------------------------------------
#ifndef Unit2H
#define Unit2H
//---------------------------------------------------------------------------
#endif
#include<vcl.h>
void ingresoVector(float x[], int n,TListBox *lst);
int buscar(float x[], int n, float dato);
void elimina(float x[], int &n, int p);
void inserta(float x[], int &n,float dato, int p);
void mostrarVector(float x[], int n, TListBox *lst);
void burbuja(float x[], int n);
```

Archivo unit2.cpp

```
//---------------------------------------------------------------------------
#pragma hdrstop
#include "Unit2.h"

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

#pragma package(smart_init)

void ingresoVector(float x[], int n,TListBox *lst)
{
  int i;
  lst->Items->Clear();
  for(i=0;i<n;i++)
  {
    x[i]=InputBox("Ingreso","elemento["+IntToStr(i)+"]:","").ToDouble();
    lst->Items->Add(x[i]);
  }
}
```

```
int buscar(float x[], int n, float dato)
{
  int i;
  for(i=0;i<n;i++)
  {
    if(x[i]==dato)
      return i;
  }
  return -1;
}

void elimina(float x[], int &n, int p)
{
  int i;
  for(i=p;i<n-1;i++)
    x[i]=x[i+1];
  n=n-1;
}

void mostrarVector(float x[], int n, TListBox *lst)
{
  int i;
  lst->Items->Clear();
  for(i=0;i<n;i++)
    lst->Items->Add(x[i]);
}

void inserta(float x[], int &n,float dato, int p)
{
  int i;
  for(i=n-1;i>=p;i--)
     x[i+1]=x[i];
  x[p]=dato;
  n=n+1;
}

void burbuja(float x[], int n)
{
  int i,j;
  float temp;
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      if(x[i]>x[j])
      {
        temp=x[i];
        x[i]=x[j];
        x[j]=temp;
      }
}
```

Archivo unit1.cpp

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

#include <vcl.h>
```

```
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
float x[100];
int n;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnIngresarClick(TObject *Sender)
{
  n=edN->Text.ToInt();
  ingresoVector(x,n,LstNum);

}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnBuscarClick(TObject *Sender)
{
  float dato;
  int p;
  dato=InputBox("Buscar","Dato a buscar : ","").ToDouble();
  p=buscar(x,n,dato);
  if(p!=-1)
    ShowMessage("Dato se encuentra en posicon "+IntToStr(p));
  else
    ShowMessage("Dato no se encuentra");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::btnEliminarClick(TObject *Sender)
{
  float dato;
  int p;
  dato=InputBox("Buscar","Dato a eliminar : ","").ToDouble();
  p=buscar(x,n,dato);
  if(p!=-1)
  {
    elimina(x,n,p);
    mostrarVector(x,n,LstNum);
    ShowMessage("Dato eliminado");
  }
  else
    ShowMessage("Dato no se encuentra");

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

```
void __fastcall TForm1::btnModificarClick(TObject *Sender)
{
  float dato;
  int p;
  dato=InputBox("Buscar","Dato a modificar : ","").ToDouble();
  p=buscar(x,n,dato);
  if(p!=-1)
  {
    x[p]=InputBox("Modificar","Nuevo dato : ","").ToDouble();
    mostrarVector(x,n,LstNum);
    ShowMessage("Dato modificado");
  }
  else
    ShowMessage("Dato no se encuentra");

}

//---------------------------------------------------------------------------
void __fastcall TForm1::btnInsertarClick(TObject *Sender)
{
 float dato;
 int p;
 dato=InputBox("Insercion","Dato a insertar","").ToDouble();
 p=InputBox("Insercion","posicion a Insertar","").ToInt();
 if(p>=0 && p<=n)
 {
    inserta(x,n,dato,p);
    mostrarVector(x,n,LstNum);
    ShowMessage("Dato insertado");
 }
 else
     ShowMessage("Posicion no valida");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::brnOrdenarClick(TObject *Sender)
{
  burbuja(x,n);
  mostrarVector(x,n,LstNum);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::btnLimpiarClick(TObject *Sender)
{
  edN->Clear();
  LstNum->Items->Clear();
  edN->SetFocus();
  n=0;
}
//---------------------------------------------------------------------------

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