MANUAL DE REFERENCIA DE C++

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA
DE SISTEMAS E INDUSTRIAL

BOGOTÁ D.C.
2009

Objetivo: Brindar a los estudiantes del curso de Programación de Computadores una guía rápida de la sintaxis de C++.

1. ESTRUCTURAS DE CONTROL

En C++ existen tres tipos de estructuras de control:

Estructuras de control

- **Estructura de secuencia**
  - Incluida en C++. A menos que se indique lo contrario, la computadora ejecutará las instrucciones una tras orea, en el orden en que se escribieron.
- **Estructura de selección**
  - **if** : Realiza una acción si la **condición** es **verdadera**, o evita una acción si la condición es falsa.
  - **if/else** : Realiza una **acción** si la **condición** es **verdadera**, o realiza una **acción** diferente si la **condición** es **falsa**.
  - **switch** : Estructura de **selección múltiple**. Se selecciona la acción a realizar, a partir de muchos estados diferentes, dependiendo del valor de una expresión **entera**.
- **Estructura de repetición**
  - **for** : Realiza la **acción** un **número finito de veces**. Es una **repetición controlada por** un **contador**.
  - **while** : Realiza una **acción** un **número finito de veces mientras** alguna **condición** es **verdadera**.
  - **do/while** : Similar al while, sólo que se **garantiza** que el **cuerpo** del **ciclo** se **ejecuta al menos una vez**. Útil para la validación de datos dados por el usuario.

C++ cuenta con siete estructuras de control: una de **secuencia**, tres de selección (**if, if/else, swhitch**) y tres de repetición (**for, while, do/while**).

Recomendación 1. *Utilizar la estructura de selección* ***if*** *no implica utilizar siempre un* ***else****. Hay ocasiones en que un* ***else*** *no es necesario, por tanto, no lo coloque.*

Recomendación 2. *Utilice* ***for*** *cuando sepa de antemano cuántas veces se ejecutará el cuerpo del ciclo, de lo contrario utilice* ***while*** *o* ***do/while****.*

Recomendación 3. *Utilice* ***while*** *cuando el cuerpo del ciclo tenga que ejecutarse sólo si una condición es cierta.*

Recomendación 4. *Utilice* ***do/while*** *cuando desee que el cuerpo del ciclo se ejecute al menos una vez.*

2. EJEMPLOS DE USO DE LAS ESTRUCTURAS DE CONTROL

2.1 ESTRUCTURA DE SECUENCIA IF

La sintaxis de un **if** es la siguiente:

```
if (condición_es_verdadera)
{
    /*Bloque de instrucciones*/
}
```

```
if (n % 2 == 0)
{
   cout << "El número digitado es par…";
}
```

***Recomendación* 5**. Nunca coloque un **; (punto y coma)** después de la condición del **if** porque esto crearía una estructura vacía (sin cuerpo) ya que las instrucciones que hay dentro de él se ejecutan aunque la condición sea falsa.

***Recomendación* 6**. C++ utiliza las siguientes convenciones:

=: **Asignar un valor a una variable.**

== **:** **Comparar dos expresiones.**

Cuando se evalúan condiciones que implican comparaciones de sus valores entonces se utiliza el "*doble igual*".

Ifs anidados:

```
#include <iostream.h>
#include <stdlib.h>

void main() /*main no retorna ningún valor (no es necesario "return 0;") */
{
    int numero; /* Variable de entrada */

    cout << "Digite un número entre 1 y 5: ";
    cin >> numero;

    if (numero == 1)
    {
        cout << "El número digitado es: 1..." << endl;
    }
    if (numero == 2)
    {
        cout << "El número digitado es: 2..." << endl;
    }
    if (numero == 3)
    {
        cout << "El número digitado es: 3..." << endl;
    }
    if (numero == 4)
    {
        cout << "El número digitado es: 4..." << endl;
    }
    if (numero == 5)
    {
        cout << "El número digitado es: 5..." << endl;
    }
    system("PAUSE");  /* Ver resultado en consola antes de pulsar una tecla */

}
```

Recomendación 7. Utilice sangrías de 3 o 4 espacios para destacar el cuerpo de una estructura. Esto facilita la búsqueda de errores de sintaxis y le da una presentación más elegante y llamativa a los programas.

2.2 ESTRUCTURA DE SECUENCIA IF

La sintaxis de **if/else** es la siguiente:

```
if (condición_es_verdadera)
{
    /*Bloque de instrucciones
      que se ejecutarán si la
      condición es verdadera. */
}
else
{
    /*Bloque de instrucciones
      que se ejecutarán si la
      condición es falsa.     */
}
```

```
if (n % 2 == 0)
{
   cout << "El número digitado es par…";
}
else
{
   cout << "El número digitado es par…";
}
```

Escala if-else-if

Una empresa comercializadora contrata vendedores a los cuales les paga un salario que va de acuerdo al total de las ventas realizadas en el mes. La siguiente tabla es tomada como referencia para calcular el sueldo de cada vendedor

Venta ($)	Sueldo ($)
0 – 500000	80.000
500001-1000000	160.000
1000001-1500000	320.000
1500001-2500000	450.000
2500001-4000000	550.000
Más de 4000000	20 % de las ventas

Realizar un programa en C++ que sistematice este procedimiento y al ingresar las ventas de un empleado inmediatamente muestre el sueldo que le corresponde.

*Para este caso se puede utilizar una estructura **if/else**.*

```
#include <iostream.h>
#include <stdlib.h>

void main() /*main no retorna ningún valor (no es necesario "return 0;") */
{
    int ventas;   /* Variable de entrada */
    int salario;   /* Variable de salida */

    cout << "*****************************" << endl
         << "*** PROGRAMA NÓMINA ***" << endl
         << "*****************************" << endl;
    cout << "\n\nDigite el total de ventas del empleado en pesos($): ";
    cin >>ventas;

    if (ventas <= 500000)
       salario = 80000;
    else if (ventas <= 1000000)
       salario = 160000;
    else if (ventas <= 1500000)
       salario = 320000;
    else if (ventas <= 2500000)
       salario = 450000;
    else if (ventas <= 4000000)
       salario = 550000;
    else
       salario = int(ventas * 0.20);

    cout << "El salario del empleado es: $"  << salario << endl;

    system("PAUSE");  /* Ver resultado en consola antes de pulsar una tecla */
}  /* fin main() */
```

2.3 ESTRUCTURA DE SECUENCIA SWITCH

En algunas ocasiones hay algoritmos que contendrán una serie de decisiones para los cuales utilizar **if/else if** en escala es tedioso. Para estos casos se puede hacer uso de la estructura de selección múltiple **switch** en la cual una variable o expresión se probará por separado contra cada uno de los **valores constantes enteros** que puede asumir y de acuerdo con ellos determinar que acción se debe seguir.

Un **switch** consta de una variable entera que lo controla, un conjunto de etiquetas case que se usan para cada valor implicado, una etiqueta **default** que corresponde a

los casos que las etiquetas **case** no cubren. Aunque no es obligatorio, cada etiqueta **case** utiliza un enunciado **break** que rompe la estructura (se sale de ella) cuando el valor que controla el **switch** es equivalente.

Nota: en el caso **default** (por defecto) no tiene sentido usar un **break** ya que es el último caso evaluado.

La sintaxis de un **switch** es la siguiente:

```
switch (expresion)  {
    case constante1:
       /*Conjunto de instrucciones*/
       break;

    case constante2:
       /*Conjunto de instrucciones*/
       break;

    case constante3:
       /*Conjunto de instrucciones*/
       break;
                    .
                    .
                    .
    Default :
       /*Conjunto de instrucciones*/
}
```

```
switch (numero)
{
    case 1:
       cout << "Digitó el valor 1";
       break;
    case 2:
       cout << "Digitó el valor 2";
       break;
    case 3:
       cout << "Digitó el valor 3";
       break;
    case 4:
       cout << "Digitó el valor 4";
       break;
    default :
       cout << "Digitó un valor mayor a 4";
}
```

Un almacén ofrece descuentos según la hora de compra:

	7:30 a 8:30 a.m	8:30 a 9:30 a.m	Otra hora
Frutas y verduras	10%	5%	0%
Productos de aseo	5%	2%	0%
Carnes	5%	2%	0%
Ropa	10%	8%	0%
Licores	8%	6%	0%

Realizar un programa en C++ que calcule el descuento de acuerdo a la hora de compra.

```
#include <iostream.h>
#include <stdlib.h>

void main()
{
  system ("CLS");   //limpiar pantalla
  //Variables
  int hora;  //Guarda la hora de llegada del cliente
  float legumbres, aseo, carnes, ropa, abarrotes;
  float valorReal, valorNeto;
  float ahorro;
  //Presentación
  cout    << "*****************************************************************************\n"
          << "******************      PROGRAMA DESCUENTOS EN MERCADO     ******************\n"
          << "*****************************************************************************\n\n";
  //FASE DE INICIACIÓN: Iniciación de variables
  cout << "Digite el valor de compra de las frutas y verduras:   ";
  cin  >> legumbres;
  cout << "Digite el valor de compra de los productos de aseo: ";
  cin  >> aseo;
  cout << "Digite el valor de compra de carnes:                ";
  cin  >> carnes;
  cout << "Digite el valor de compra de ropa y calzado:        ";
  cin  >> ropa;
  cout << "Digite el valor de compra de licores:               ";
  cin  >> abarrotes;
  cout << "\nDigite la opción según la hora de compra:\n"
          << "\n 1             : La hora está entre las 7:30 a.m y 8:30 a.m"
          << "\n 2             : La hora está entre las 8:30 a.m y 9:30 a.m"
          << "\n Otro valor : Ninguna de las anteriores"
          << "\n\nOpción: ";
  cin >> hora;
  //FASE DE PROCESAMIENTO: Procesamiento de los datos
  valorReal = legumbres + aseo + carnes + ropa + abarrotes;
  switch(hora)
  {
     case 1:
          ahorro = legumbres*0.1 + aseo*0.05 + carnes*0.05 + ropa*0.1 + abarrotes*0.08;
          valorNeto = valorReal - ahorro;
          break;    //Salir del switch
     case 2 :
          ahorro = legumbres*0.05 + aseo*0.02 + carnes*0.02 + ropa*0.08 + abarrotes*0.06;
          valorNeto = valorReal - ahorro;
          break;    //Salir del switch
     default :
          valorNeto = valorReal;
          ahorro = 0;
  }
  //FASE DE TERMINACIÓN : Muestra de resultados
  cout    << "\n************************************"
          << "\n  Valor total:           " << valorReal
          << "\n  Valor neto a pagar: " << valorNeto
          << "\n  Usted ahorró:         " << ahorro
          << "\n************************************";
  system("PAUSE");  //Ver resultados antes de pulsar una tecla
}
```

2.4 ESTRUCTURA DE ITERACIÓN FOR

La sintaxis de la estructura **for** en C++ es la siguiente:

```
for (contador = lim_inf; contador < lim_sup; contador++)
{
    /*Conjunto de instrucciones*/
}
```

contador. Variable que controla el ciclo **for**. En este caso aumenta de 1 en 1 desde el límite inferior hasta lim_sup (no incluye el valor lim_sup), es decir:

Dominio (contador) = [lim_inf, lim_sup - 1]

lim_inf. Valor inicial que toma la variable contador.

lim_sup. Valor final que toma la variable contador. Para este caso se debe tener en cuenta que:

*Si el encabezado del **for** es* **for**(contador = lim_inf; contador < lim_sup; contador++) entonces el límite superior es **lim_sup - 1** porque hay un operador **<**.

*Si el encabezado del **for** es* **for**(contador = lim_inf; contador <= lim_sup; contador++) entonces el límite superior es **lim_sup** porque hay un operador **<=**.

Se utiliza un **for** cuando se sabe de antemano cuántas veces se ejecutará el bloque de instrucciones del ciclo.

Ejemplos de encabezados for

for (j = 0; j <= 24; j = j +4**)**	Dom(j) = [0, 24] donde j varía de 4 en 4
for (j = 50; j > 5; j -= 5**)**	Dom(j) = (5, 50] donde j varía de 5 en 5 Incrementos de -5 o decrementos de 5
for (j = 2; j <= 20; j += 3**)**	j $\in$ {2, 5, 8, 11, 14, 17, 20}, j varía de 3 en 3

Recomendación 7. No inicialice fuera del ciclo **for** la variable de control de mismo, ya que esto se hace automáticamente. En caso de hacerlo, no se generará ningún error.

Juan quiere saber qué valores divisibles en 11 y en 15 existen en el rango de 0 a 1000 con el fin de ayudarle a hacer la tarea a su hermanita que está en primaria. Juan desea adquirir un programa que calcule dichos números y los imprima en pantalla.

```
#include <iostream.h>
#include <stdlib.h>

int main() /* La función main() retorna un valor entero, usa return*/
{
    int valor; /* variable de control del for */

    cout << "Valores divisibles en 11 y 15 en un rango de 0 a 1000\n\n";

    for (valor = 0; valor <= 1000; valor++)
    {
       if (valor % 11 == 0 && valor % 15 == 0) /* &&: operador booleano Y */
       {
          cout << valor << "\t";     /* imprime valor */
       }
    }
    cout << endl;  /* Coloca el cursor en la siguiente línea */

    system("PAUSE");    /* Ver resultados antes de presionar una tecla */
    return 0;
}
```

2.5 ESTRUCTURA DE ITERACIÓN WHILE

La sintaxis de la estructura **while** en C++ es la siguiente:

```
while (condición)
{
    /*Conjunto de instrucciones*/
    /*Una instrucción que en algún momento haga falsa la condición
      que controla el while */
}
```

Por lo regular alguna acción del cuerpo de un **while** eventualmente debe hacer que la condición que lo controla se haga falsa, de lo contrario, el ciclo no terminará nunca (**ciclo infinito**).

***Recomendación* 8.** Antes de ejecutar un programa que utilice estructuras de control **while** verifique en cada una que la condición que la controla en algún momento se hará falsa. Si no hace lo anterior, se corre el riesgo que el programa se quede ejecutando continuamente y en muchos casos el computador no responda.

*Todo **for** puede especificarse con un **while**:*

for (j = 0; j <= 24; j = j +4)
{
/*Conjunto de instrucciones */
}
	j = 0;
	while (j <= 24)
	{
	/*Conjunto de instrucciones */
	j = j + 4;
	}

for (j = 50; j > 5; j -= 5)
{
/*Conjunto de instrucciones */
}
	j = 50;
	while (j > 5)
	{
	/*Conjunto de instrucciones */
	j -= 5; /* j = j -5 */
	}

for (j = 2; j <= 20; j += 3)
{
/*Conjunto de instrucciones */
}
	j = 2;
	while (j <= 20)
	{
	/*Conjunto de instrucciones */
	j += 3;
	}

La empresa comercializadora a la cual usted le desarrolló el programa, desea que usted lo modifique para que pueda calcularse el salario de un número n de empleados. Además desea que los valores digitados sean válidos ya que se han presentado casos en los que se ingresan valores negativos y aún así se calcula el salario del empleado.

Las condiciones del problema son las mismas.

```
#include <iostream.h>
#include <stdlib.h>

void main() /*main no retorna ningún valor (no es necesario "return 0;") */
{
    int ventas;          /* Variable de entrada */
    int salario;         /* Variable de salida */
    int empleados;       /* Número de empleados */
    int empleado;        /* Variable de control del while */

    cout << "*****************************" << endl
         << "*** PROGRAMA NÓMINA ***" << endl
         << "*****************************" << endl;

    cout << "Digite el número de empleados de la nómina para este mes: ";
    cin >> empleados;

    //Validación del número de empleados
    while (empleados <= 0)
    {
       cout << "Digite valores mayores a 0...\n\n";
       cout << "Digite el número de empleados de la nómina para este mes: ";
       cin >> empleados;
    }

    empleado = 1;   /* Inicializar la variable de control */
    while (empleado <= empleados)
    {
      cout << "\n\nDigite el total de ventas del empleado " << empleado  <<" en pesos($): ";
      cin >>ventas;

      //Validación total de ventas
      while (ventas <= 0)
      {
         cout << "Digite valores mayores a 0...\n\n";
         cout << "Digite el total de ventas del empleado " << empleado <<" en pesos($): ";
         cin >> ventas;
      }

      if (ventas <= 500000)
        salario = 80000;
      else if (ventas <= 1000000)
        salario = 160000;
      else if (ventas <= 1500000)
        salario = 320000;
      else if (ventas <= 2500000)
        salario = 450000;
```

```
      else if (ventas <= 4000000)
        salario = 550000;
      else
        salario = int(ventas * 0.20);

      cout << "El salario del empleado " << empleado << " es: $"  << salario << endl;

      empleado++;    /* aumentar la variable de control para que en algún
                        momento la condición se haga falsa */
    } /*fin while*/

    cout  << "\n\nFin programa nómina..."  << endl;
    system("PAUSE");  /* Ver resultado en consola antes de pulsar una tecla */

}  /* fin main() */
```

2.6 ESTRUCTURA DE ITERACIÓN DO/WHILE

La sintaxis de la estructura **do/while** en C++ es la siguiente:

```
do {
    /*Conjunto de instrucciones*/
    /*Una instrucción que en algún momento haga falsa la condición
      que controla el do/while */
} while (condición);
```

***Recomendación* 9.** Cuando utilice la estructura de iteración **do/while** acuérdese de colocar el ;(**punto y coma**) después de la condición. Esta es la única estructura donde se coloca el **punto y coma** después de la condición.

***Recomendación* 10.** Utilice **do/while** para el proceso de validación de datos dados por el usuario, ya que esta estructura garantiza que su cuerpo se ejecute al menos una vez.

3. ARREGLOS Y MATRICES

A continuación se hará una introducción básica del manejo de arreglos y matrices en C++.

3.1 ARREGLOS

Un arreglo es una colección de variables del mismo tipo que se referencian por un nombre común. Para acceder a un elemento específico se utiliza un índice que indica la posición del elemento.

Arreglo

E1	E2	E3	E4	E5	E6	E7	E8
Pos 0	Pos 1	Pos 2	Pos 3	Pos 4	Pos 5	Pos 6	Pos 7

Arreglo[0] = E1 Arreglo[1] = E2 Arreglo[2] = E3 Arreglo[3] = E4
Arreglo[4] = E5 Arreglo[5] = E6 Arreglo[6] = E7 Arreglo[7] = E8

*El **índice** de un arreglo siempre debe ser un **valor entero**.*
El **sexto elemento** de un arreglo **equivale** al elemento ubicado en la posición **n-1**
El **elemento 6** del arreglo **tiene indice 6** y y es el **séptimo** elemento del arreglo.

Recomendación 11. El hecho de que los índices de arreglos y matrices en C++ empiecen en 0 da pie a muchos errores lógicos (*errores no detectados por el compilador*), por tanto, se debe tener especial cuidado al momento de realizar operaciones con los índices de un arreglo o matriz.

3.1.1 Declaración de un arreglo. Se debe especificar el tipo, el nombre y el tamaño del mismo entre paréntesis cuadrados ([ ]).

tipo nombre_del_arreglo[*tamaño*];

El tamaño del arreglo debe especificarse de forma explicita, es decir, su valor debe definirse en tiempo de codificación y no en tiempo de ejecución.

Recomendación 12. Si no se tiene certeza de cuántos elementos puede llegar a tener un arreglo, se debe establecer un rango que cubra el máximo número de

elementos. Si no se define un tamaño para el arreglo o este es una variable no constante entonces se producirá un error.

A continuación se muestra dos maneras de definir arreglos:

```
#include <iostream.h>
#include <stdlib.h>
#define TAM 10

int main()
{
    int tamanio;
    int arreglo[TAM];

    do {
      cout << "Digite tamaño del arreglo (entre 1 y 10): ";
      cin >> tamanio;
    } while (tamanio < 1 || tamanio > 10);

    system("PAUSE");
    return 0;
}
```

```
#include <iostream.h>
#include <stdlib.h>

int main()
{
    const int TAM = 10;
    int tamanio;
    int arreglo[TAM];

    do {
      cout << "Digite tamaño del arreglo (entre 1 y 10): ";
      cin >> tamanio;
    } while (tamanio < 1 || tamanio > 10);

    system("PAUSE");
    return 0;
}
```

3.1.2 Acceder a un elemento de un arreglo. Como se dijo anteriormente, se hace mediante el uso de un índice.

Acceder al séptimo elemento del arreglo: arreglo**[6];**
Acceder al elemento siete del arreglo: arreglo**[7];**

3.1.3 Inicializar un arreglo. Si se desea inicializar un arreglo en tiempo de codificación se debe hacer al inicializar el arreglo. Se puede hacer de las siguientes maneras:

```
int arreglo[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
int arreglo[5] = { 0 };       /* todos los elementos del arreglo se inicializan con 0 */
int arreglo[10] = { 5 };      /* arreglo = { 5, 0, 0, 0, 0, 0, 0, 0, 0, 0 } */
int arreglo[6] = { 5, 4, 3 }; /* arreglo = { 5, 4, 3, 0, 0, 0 } */
```

Para el caso 2, 3, 4, la inicialización a 0 de los elementos ocurre en tiempo de compilación.

3.1.4 Inicializar e imprimir un arreglo a un valor dado por el usuario usando for y while

```
/* SOLUCIÓN DEL PROBLEMA USANDO FOR */

#include <iostream.h>
#include <stdlib.h>
#define TAM 10

int main()
{
    int arreglo[TAM];          /*Arreglo de enteros                  */
    int valor;                 /*Valor de inicialización del arreglo */
    int indice;                /*Variable de control ciclo for       */

    cout << "Digite el valor con el que desea inicializar los elementos del arreglo: ";
    cin >> valor;
    //INICIALIZAR EL ARREGLO
    for (indice = 0; indice < TAM; indice++)
    {
        arreglo[indice] = valor;
    }
    //IMPRIMIR EL ARREGLO
    for (indice = 0; indice < TAM; indice++)
    {
        cout << arreglo[indice] << " ";
    }
    system("PAUSE");
    return 0;
}
```

Se define una constante TAM que es igual a 10, se declara un vector de nombre arreglo y es de 10 elementos (TAM). Se declaran dos valores enteros, *valor* que guarda el valor dado por el usuario para inicializar el arreglo, *indice* cuyo fin es controlar el ciclo **for**. Se pide al usuario que digite un valor para inicializar el arreglo, a continuación se inicializa el arreglo usando un ciclo **for**. Por último se imprime el arreglo usando también la estructura **for** y la misma variable de control.

A continuación se muestra el mismo arreglo pero usando **while**. Se debe tener en cuenta que todo ciclo **for** se puede especificar con una estructura **while**.

```
/* SOLUCIÓN DEL PROBLEMA USANDO WHILE */

#include <iostream.h>
#include <stdlib.h>
#define TAM 10

int main()
{
    int arreglo[TAM];          /*Arreglo de enteros                  */
    int valor;                 /*Valor de inicialización del arreglo */
    int indice;                /*Variable de control ciclo for       */

    cout << "Digite el valor con el que desea inicializar los elementos del arreglo: ";
    cin >> valor;
    //INICIALIZAR EL ARREGLO
    indice = 0;
    while (indice < TAM)
    {
        arreglo[indice] = valor;
        indice++;
    }
    //IMPRIMIR EL ARREGLO
    indice = 0;
    while (indice < TAM)
    {
        cout << arreglo[indice] << " ";
        indice++;
    }
    system("PAUSE");
    return 0;
}
```

3.1.5 Ejercicios resueltos usando arreglos

a. Una empresa desea procesar las ventas realizadas mes a mes anualmente con el fin de obtener la siguiente información:
   - Mes en el que se obtuvo la mejor venta
   - El monto de la venta máxima obtenida
   - Total de las ventas
   - Promedio de ventas
   - Mostrar las ventas de menor a mayor

```
#include <stdlib.h>
#include <iostream.h>

void main(void)
{
   float ventas[12];          //Arreglo que guarda las ventas del mes
   int mes, i;                //Guarda mes(indice) de una venta en el arreglo
   int mesVMax;               //Guarda el mes de la venta máxima
   float ventaMax;            //Guarda el monto de la venta máxima
   float totalVentas;         //Guarda el monto total de ventas en  el año
   float temporal;            //Se utiliza en el ordenamiento de las ventas
   system("CLS");             //Limpiar pantalla
   cout << "PROGRAMA DE PROCESAMIENTO DE VENTAS ANUALES: \n\n";
   /*CAPTURA DE DATOS*/
   for(mes = 0; mes < 12 ; mes++) {
          cout << "Digite las ventas del mes número " << mes +1 <<": ";
          cin  >> ventas[mes];
   }
   system("CLS");    //Limpiar pantalla
   cout << "PROGRAMA DE PROCESAMIENTO DE VENTAS ANUALES: \nVentas\n";
   //Escribir ventas
   for(mes = 0; mes < 12 ; mes++)
          cout << "Mes " << mes +1 <<": " << ventas[mes] << "\n" ;
   cout << "\n";
   /*DETERMINAR VENTA MAXIMA, MES VENTA MAXIMA, TOTAL VENTAS*/
   totalVentas = ventas[0];
   ventaMax = ventas[0];
   mesVMax = 0;
   for(mes = 1; mes < 12 ; mes++) {
          if(ventas[mes] > ventaMax) {
            ventaMax = ventas[mes];
            mesVMax  = mes;
          }                                   //fin if
          totalVentas += ventas[mes];
   }                                          //fin for
   //Ventas en orden ascendente
   for (i = 0; i < 12; i++)
     for(mes = 0; mes < 11 - i ; mes++) {
            if(ventas[mes] > ventas[mes+1]) {
              temporal = ventas[mes];
              ventas[mes] = ventas[mes+1];
              ventas[mes+1] = temporal;
            }                                        //fin if
     }                                               //fin for
   /*MUESTRA DE RESULTADOS*/
   cout   << "La venta máxima se dio en el mes:      "  << mesVMax + 1   << endl
          << "La venta máxima en el año fue:        "  << ventaMax      << endl
          << "Total de ventas en el año:            "  << totalVentas   << endl
          << "Promedio de ventas en el año:         "  << totalVentas/12  << endl
          << "\nVentas en orden ascendente:         "  << endl;
   //Ventas en orden ascendente
   for(mes = 0; mes < 12 ; mes++)
          cout << ventas[mes] << "  ";
   cout << "\n\n";
   system("PAUSE"); //Ver resultados en pantalla antes de pulsar una tecla

}
```

Búsqueda lineal en un arreglo. A continuación se implementa un algoritmo de búsqueda lineal que permite determinar si un elemento está o no dentro de un arreglo. Este tipo de búsqueda consiste en comparar el elemento buscado con cada uno de los elementos del arreglo. Si el arreglo no está ordenado, existe la misma probabilidad de que el elemento se encuentre, ya sea en el primer elemento como en el último[1]. Este método funciona bien en arreglos pequeños o para arreglos no ordenados ¿Por qué?

```
#include <iostream.h>
#include <stdlib.h>

 int main()
{
    const int tamanoArreglo = 100;       // tamaño del arreglo a
    int a[ tamanoArreglo ];              // crea el arreglo a
    int claveBusqueda;                   // valor a localizar dentro de a
    int pos;                             // Posición de la clave buscada

    for ( int i = 0; i < tamanoArreglo; i++ )  // crea algunos datos
        a[ i ] = 2 * i;

    cout  <<  "Introduce la clave de búsqueda entera: ";
    cin >> claveBusqueda;

    pos = -1;                 // Si no se encuentra entonces en pos se guarda -1
    // intenta localizar claveBusqueda dentro del arreglo a
    for ( int j = 0; j < tamanoArreglo; j++ )
        if ( a[ j ] == claveBusqueda )       // si se encuentra ,
              pos = j;                       // guarda la ubicación de la clave

     // despliega los resultados
     if ( pos != -1 )
        cout << "Encontré valor en el elemento " << pos << endl;
     else
        cout << "Valor no encontrado" << endl;

     system ("PAUSE");
        return 0;   // indica terminación exitosa

} // fin de main
```

[1] Ejercicio 4.19 de C++ Cómo Programar 4 ed. Deitel y Deitel. Se hicieron algunas modificaciones al ejercicio.

3.2 MATRICES

Una matriz es un arreglo bidimensional, es decir, que se maneja con dos índices. En C++ se representan sus posiciones así:

Matriz **M** de 3x3

M[0][0]	M[0][1]	M[0][2]
M[1][0]	M[1][2]	M[1][2]
M[2][0]	M[2][1]	M[2][2]

El primer índice indica las filas y el segundo las columnas.

3.2.1 Inicializar matrices

```
int arreglo1[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
int arreglo2[3][3] = { 0 }; /* todos los elementos del arreglo se inicializan con 0 */
int arreglo3[2][3] = { {2,4},{6,8} };    /* { {2, 4, 0}, {6, 8, 0} } */
```

3.3.2 Inicializar e imprimir una matriz

```
#include <iostream.h>
#include <stdlib.h>
#define FILAS 2
#define COL 3

int main()
{
   int matriz[FILAS][COL];
   /*capturar datos de la matriz */
   for (int fila = 0; fila < FILAS; fila++)
   {
       for (int col = 0; col < COL; col++)
       {
            cout << "Digite el elemento de la posición [" << fila << "][" << col << "]: ";
            cin >> matriz[fila][col];
       }
   }
   /*imprimir datos de la matriz */
   cout << "\n\n*****  MATRIZ GENERADA  *****\n\n" ;
   for (int fila = 0; fila < FILAS; fila++)
   {
       for (int col = 0; col < COL; col++)
       {
          cout << matriz[fila][col] << "\t";
       }
       cout << "\n";
   }
   system("PAUSE");
   return 0;
}
```

Los resultados de las elecciones de gobernadores en Colombia se esquematizaron en una matriz como la que se muestra a continuación:

Partido → Departamento ↓	P1	P2	P3	P4	P5	P6	P0(Votos en blanco)
Valle							
Cauca							
…							
Caldas							

La matriz almacena el número de votos que obtuvo cada partido en el departamento correspondiente, así como los votos en blanco en cada departamento.

Se desea conocer:

- Total de votos

- Total de votos por departamento, para comparar con una tabla que indica el número de personas que deben votar por departamento con el fin de detectar en que departamentos votó menos del 60% de la población electora y tomar así medidas de concientización cívica.

Número de personas que deben votar por estado:

Valle	Cauca	. . .	Caldas

- Que partido obtuvo el mayor número de votos

- ¿Cuál es el departamento con mayor número de abstenciones y cuál es el departamento con mayor porcentaje de abstenciones?

En la siguiente página se encuentra el algoritmo implementado en C++.

```
#include <iostream.h>
#include <stdlib.h>
#define DPTOS 3

void main()
{
    int votos[DPTOS][7],votantes[DPTOS],abstencion[DPTOS],total,i,j;
    int maximo,max_indice;
    system("CLS");
    for(i=0;i<DPTOS;i++) {
        for(j=0;j<=6;j++)  {
            if(j == 0) {
                cout<<"Votos en blanco en el Dpto " <<i<<": ";
                cin>>votos[i][j]; // Almacena los datos leidos en la matriz
            }
            else {
                cout<<"Votos del partido P"<<j<<" en el Dpto "<<i<<": ";
                cin>>votos[i][j]; // Almacena los datos leidos en la matriz
            }
        }
    }
    cout<<endl;
    for(i=0;i<DPTOS;i++){
            cout<<"Personas que deben votar en el Dpto "<<i<<": ";
            cin>>votantes[i];
    }
    cout <<endl;
    // Suma la matriz por columnas para dar el total de votos de cada partido
    for(j=1;j<=6;j++){
            total=0; // Variable acumuladora, por lo cual hay que inicializarla en 0
            for(i=0;i<DPTOS;i++){
                    total = total+votos[i][j];
            }
            cout<<"Votos obtenidos por el partido P"<<j<<": "<<total<<endl;
    }
    cout<<endl;
    // Suma la matriz por filas para dar el total de votos por departamento
    maximo=0;
    for(i=0;i<DPTOS;i++){
            total=0; // Variable acumuladora, por lo cual hay que inicializarla en 0
            for(j=0;j<=6;j++){
                    total=total+votos[i][j];
                    if(i==1 && maximo<votos[i][j]){ // Busca el partido de más votos en el segundo Dpto.
                            maximo=votos[i][j];
                            max_indice=j; // Almacena el índice del partido con más votos
                    }
            }
            abstencion[i]=votantes[i]-total; // Calcula la abstención de una vez
            cout<<"Votos totales en el Dpto "<<i<<": "<<total;
            cout<<"; Votó el "<<total*100/votantes[i]<<"% de la población."<<endl;
    }
    cout<<endl<<"El partido P"<<max_indice<<" obtuvo el mayor número de votos en el segundo Dpto."<<endl<<endl;
    maximo=0;
    for(i=0;i<DPTOS;i++){
            if(maximo<abstencion[i]){ // Busca el Dpto de más abstención
                    maximo=abstencion[i];
                    max_indice=i;
            }
            cout <<"Abstención de "<<abstencion[i]<<" en el Dpto "<<i<<", para un porcentaje de     "
                    <<abstencion[i]*100/votantes[i]<<"% "<<endl;
    }
    cout<<endl<<"El partido con mayor abstención fue el Dpto "<<max_indice<<endl;
    system("PAUSE");
}
```