Algoritmos y Estructuras de Datos I

Trabajo Práctico N°5	Unidad 5
Modalidad: Semi -Presencial	**Estratégica Didáctica:** Trabajo grupal
Metodología de Desarrollo: Det. docente	**Metodología de Corrección:** Via Classroom.
Carácter de Trabajo: Obligatorio – Con Nota	**Fecha Entrega:** A confirmar por el Docente.

MARCO TEÓRICO

Responder el siguiente cuestionario en función de la bibliografía Obligatoria.

Unidad 4

1. Escribir y explicar la secuencia de código de recorrido de Arrays con Centinela.

Centinelas: También llamados "banderas", los centinelas son variables, normalmente de tipo lógicas (boolean), conservan un estado hasta que un evento requiera cambiarlo y ejecutar otra funcionalidad.

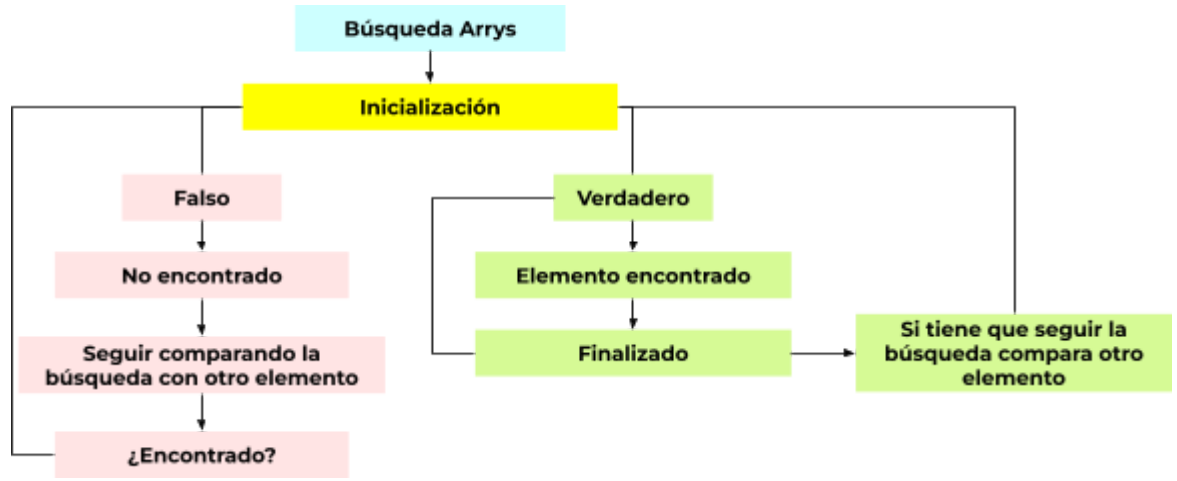
Los arrays (arreglos) en C++, estos son grupos de variables del mismo tipo, que se acceden utilizando un índice. No es posible conocer la longitud de un array a partir de un puntero a datos. Una vez se tiene el array, se puede utilizar una variable para marcar su fin, que generalmente se llama centinela o bandera.

para trabajar con arrays utilizando centinelas en C++, debe definir primero el array y luego establecer un valor específico para la centinela, normalmente un valor no válido o nulo. Utilizando este valor, puede iterar a través del array para procesar los valores y luego salir del ciclo al encontrar la centinela.

```
const int N = 10;
typedef double tArray[N];
tArray datos; // Datos positivos: centinela = -1
int i = 0; double elemento = datos[i];
while (elemento != -1) {
 // Procesar el elemento ...
 i++;
elemento = datos[i];
}
```

2. Escribir y explicar la secuencia de código de Búsqueda un Arrays.

La secuencia de búsqueda en un arreglo Arrays, comienza con la inicialización desde el primer elemento. Se comparan los datos buscados con cada elemento del arreglo. Si no se encuentra el dato buscado, se continúa recorriendo el arreglo elemento por elemento para verificar si coincide con el valor buscado. Cuando hay una coincidencia entre el valor buscado y el elemento actual del arreglo, se concluye que se ha encontrado el elemento buscado. Si se recorre todo el arreglo sin encontrar el valor deseado, se indica que la búsqueda no tuvo éxito.



3. ¿Qué entiende por Arrays Multidimensionales?

Los arreglos multidimensionales los entendemos como una forma de estructura similar a los vectores. Su matriz es una serie de vectores contenidos uno en el otro, u otros. está forma de estructura posee columnas y filas.

Estos son arrays que tienen más de un valor de índice. Un array multidimensional es una colección de componentes iguales ordenados en más de una dimensión. Se accede a cada componente por medio de un juego de índices, uno para cada dimensión, que representan la posición del componente en las diferentes dimensiones. Se podrá imaginar cada índice como la descripción de una característica de un componente de array dado.

4. Consigna: Dar un ejemplo de Arrays multidimensional en Código (definición e inicialización)

Colección de componentes, todos del mismo tipo, ordenados en N dimensiones (N 1). Cada componente es accedido por N índices, de los que cada uno representa la posición del componente dentro de esta dimensión.

Este ejemplo ilustra cómo se puede trabajar con arrays multidimensionales en C++ para almacenar y procesar datos estructurados:

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
//la definición de constantes es:
const int MaxDias = 31;                    // el Maxdias es el total que corresponde a 31
const int MED = 2; // Número de medidas por día        // MED es la medida min y max del día
// Definimos el tipo de array
typedef double tTemp[MaxDias][MED];          // Tipo de matriz para almacenar temperaturas
tTemp temp;              // tTemo = MaxDias y temp = MED / Array para almacenar las temperaturas
//Inicialización se almacena las medidas y valores de dia
double tMaxMedia = 0, tMinMedia = 0;
double tMaxAbs = -100, tMinAbs = 100;
int dia = 0;
double max, min;
//Lectura de archivo
ifstream archivo;
archivo.open("temp.txt");
//comprobación de lectura si abre correctamente
//luego procede a leer los mínimos y máximos indicado de -99, hasta llegar al máximo de días
if (!archivo.is_open()) {
   cout << "No se ha podido abrir el archivo!" << endl;
} else {
   archivo >> min >> max;
   while (!((min == -99) && (max == -99)) && (dia < MaxDias)) {
      temp[dia][0] = min;          // Temperatura mínima del día
      temp[dia][1] = max;          // Temperatura máxima del día
      dia++;
      archivo >> min >> max;
   }
   archivo.close();
}
//cálculo de suma entre  temperaturas minimas y maximas para determinar valor por dia
for (int i = 0; i < dia; i++) {
   tMinMedia += temp[i][0];
   if (temp[i][0] < tMinAbs) tMinAbs = temp[i][0];
```

```
    tMaxMedia += temp[i][1];
    if (temp[i][1] > tMaxAbs) tMaxAbs = temp[i][1];
  }
  tMinMedia /= dia;
  tMaxMedia /= dia;
  // Mostrar el resultado
  cout << "Temperaturas mínimas.-" << endl;
  cout << "    Media = " << fixed << setprecision(1) << tMinMedia << " C    Mínima absoluta = " <<
  setprecision(1) << tMinAbs << " C" << endl;
  cout << "Temperaturas máximas.-" << endl;
  cout << "    Media = " << fixed << setprecision(1) << tMaxMedia << " C    Máxima absoluta = " <<
  setprecision(1) << tMaxAbs << " C" << endl;
  // se organiza los datos en filas y columnas
  temp[dia][0] -> Temperatura mínima del día 'dia'
  temp[dia][1] -> Temperatura máxima del día 'dia'
```

5. Como se recorre un Arrays multidimensional, dar un ejemplo de Código.

Un array multidimensional puede ser de 1, 2, 3 o más dimensiones. A partir de 3 dimensiones, la complejidad de manejo del código aumenta significativamente. Para recorrer estos arrays, se utilizan bucles anidados. Cada bucle recorre una dimensión del array. Lo que hace el bucle anidado es fijar una variable, mientras recorre otra. Cada bucle recorre una dimensión del array.

Se recomienda:

Verificar los límites: Asegúrate de que los índices de los bucles estén dentro de los límites del array para evitar errores de acceso.

Declaración de arrays multidimensionales como parámetros: Cuando declares un array multidimensional como parámetro, debes especificar todos los tamaños de las dimensiones excepto la primera. Los tamaños deben coincidir con los del argumento.

Uso de `typedef` para definir tipos de arrays multidimensionales: Esto ayuda a evitar problemas de tamaño y proporciona una manera más clara de manejar arrays multidimensionales en tu código.

EJEMPLO DE CÓDIGO:

Se recorre un array bidimensional y se imprime en diferentes formatos utilizando bucles anidados.

Primera Parte del recorrido:
Imprime el array por columnas usando un bucle anidado.

```
//Inicio
#include <iostream>
#include <iomanip>
using namespace std;

const int ROWS = 4;          //número de filas
const int COLS = 10;         //número de columnas
//Array "votes" con 4 filas y 10 columnas, se inicializa con valores enteros.
int votes[ROWS][COLS] = {
   {0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
   {10, 11, 12, 13, 14, 15, 16, 17, 18, 19},
   {20, 21, 22, 23, 24, 25, 26, 27, 28, 29},
   {30, 31, 32, 33, 34, 35, 36, 37, 38, 39}
};
// Las columnas
```

```
//Los elementos se imprimen verticalmente en lugar de horizontalmente.
cout << "Array imprimido por columnas:" << endl;
for (int level = 0; level < COLS; level++) {      // BucleExterno(level): Recorre las columnas
for (int rockgroup = 0; rockgroup < ROWS; rockgroup++) {          //BucleInterno(rockgroup): Recorre las filas
      cout << setw(4) << votes[rockgroup][level];//              //imprime
   }
   cout << endl;
}
```

Segunda Parte del recorrido:

Imprime el array por filas usando un bucle anidado.

```
// Filas
// El array en formato fila por fila.
cout << "Array imprimido por filas:" << endl;
for (int level = 0; level < ROWS; level++) {     //Bucle Externo (level): Recorre las filas
for (int rockgroup = 0; rockgroup < COLS; rockgroup++) { //BucleInterno(rockgroup): Recorre las columnas
                                                          para cada fila.
 cout << setw(4) << votes[level][rockgroup];          //imprime
   }
   cout << endl;
}
```

Tercera Parte del recorrido:

Muestra una impresión adicional similar a la segunda parte, utilizando una constante para el número de filas.

```
// Recorrido adicional
const int NUM_LEVELS = ROWS; // NUM_LEVELS se define igual al número de filas (ROWS).

cout << "Recorrido adicional con NUM_LEVELS:" << endl;
for (int level = 0; level < NUM_LEVELS; level++) {
for (int rockgroup = 0; rockgroup < COLS; rockgroup++) {
cout << setw(4) << votes[level][rockgroup]; // Similar al segundo bucle, se recorre el array por filas usando
                                                NUM_LEVELS.
   }
   cout << endl;
}
```

MARCO PRÁCTICO

Desarrollar un Programa que:

a. Un archivo de texto contiene información acerca de los productos que se venden en un almacén. Lo único que se sabe acerca del número de productos es que no puede superar un cierto valor MaxProductos. De cada producto se guarda información sobre su código identificador (entero positivo), su precio (real) y el número de unidades existentes (entero positivo). El formato en el que se guarda la información dentro del archivo es el siguiente:
id1 precio1 unidades1
id2 precio2 unidades2
...
idN precioN unidadesN
-1

1. Declara un tipo tProducto que represente la información de un producto y un tipo tLista que mantenga la información de todos los productos.
2. Escribe un subprograma que lea los datos del archivo de texto que almacena la información, los guarde en la lista y luego los muestre en la pantalla.
3. Escribe un subprograma que encuentre el producto con máximo valor en el almacén, considerando que el valor del producto i es precioi*unidades i.
4. Escribe un subprograma que dado un identificador de producto a eliminar del almacén, lo busque en la lista y lo elimine actualizando la lista como corresponda.

b. Escribe un programa que lea del teclado una frase y a continuación visualice las palabras de la frase en columna, seguida cada una del número de letras que la componen.

c. Implementa un programa que permita realizar operaciones sobre matrices de NxN. El programa debe permitir al usuario la selección de alguna de las siguientes operaciones:
a) Sumar 2 matrices.
b) Restar 2 matrices.
c) Multiplicar 2 matrices.
d) Trasponer una matriz.
e) Mostrar una matriz señalando cuáles son los puntos de silla (los puntos de silla de una matriz son aquellos elementos de la misma que cumplen ser el mínimo de su fila y el máximo de su
columna).
Habrá también dos subprogramas para leer del teclado o mostrar en la pantalla una matriz.