UNIVERSIDAD DE GUADALAJARA

Red Universitaria e Institución Benemérita de Jalisco

Centro Universitario de Ciencias Exactas e Ingenierias

Departamento de Ciencias Computacionales

Programación

“Arreglos bidimensionales”

Profesora: Patricia Sanchez Rosario

Alumno: Jonathan Silva Morales

Codigo: 216852287

Carrera: Ingenieria en Computación (INCO)

Materia: I5882

NRC: 942555

Sección: D07

Ciclo: 2020B

Arreglos bidimensionales en C++

Un arreglo de una dimensión es usado para representar elementos en una lista o secuencia de valores. En algunos casos la relación entre elementos del arreglo no pueden representarse en forma de lista. Un arreglo de 2 dimensiones es usado para representar elementos en una tabla con columnas y filas, se puede acceder a cualquier elemento dentro del arreglo indicando el indice de la columna y la fila del arreglo.

```
Conts int NUMFIL=4;                                    //número de filas

Const int NUMCOL=4;                                    //número de columnas

Float caja[NUMFIL][NUMCOL]     //arreglo creado con las constantes anteriores
```

Este ejemplo declara caja como un arreglo de 2 dimensiones cuyos elementos son variables del tipo float. Para acceder a un elemento individual del arreglo debe de especificar 2 expresiones (una por cada dimensión) ejemplo: caja[0][2] donde [0] es fila y [2] columna.

Ejemplos de declaración de arreglos bidimensionales en C++

Necesitamos un arreglo que muestre el clima de cada día de año indicando la semana y el día (un arreglo de 52*7=364)

```
Int clima[52][7];
```

Nuestro arreglo clima consta de 52 filas y 7 columnas partiendo del 0 al 51 y el 0 al 6, cada espacio de este contiene un valor int. Nuestra intención es que el arreglo guarde los datos de cada día del año. Si usamos la expresión clima[2][6], nos dará el clima del día 7 de la tercer semana..

Procesando arreglos bidimensionales en C++ para dos dimensiones

Procesar los datos de un arreglo de 2 dimensiones generalmente significa acceder al arreglo en uno o 4 patrones: aleatorio, a través de columna, a través de filas o a través del arreglo entero. Por ejemplo un usuario ingresa las coordenadas de un mapa a través de un indice en el arreglo nombre_calle para llegar a la dirección que desea, este proceso es conocido como aleatorio.

Hay veces que necesitamos utilizar una operación en cada elemento de una fila o columna en el arreglo. Suponiendo en el caso anterior del clima que queremos sacar un promedio de la temperatura de una semana debemos sumar lo valores en la fila semana y dividirlo entre 7, este caso accede al arreglo a través de las filas, si por el contrario lo que queremos es el promedio de un día de la semana en especifico (ejemplo promedio de todos lunes del año) debemos sumar los elementos de la columna días y dividirlo entre 52, este caso accede al arreglo a través de las columnas, un ejemplo mas, si queremos sacar el promedio del clima del año entero debemos acceder a cada elemento del arreglo (en este caso el orden para acceder a las filas o columnas no tiene relevancia) y dividirlo entre 364, esto accede a todo el arreglo.

Considerando los 4 ejemplos comunes para procesar los arreglos echemos un vistazo a los siguientes patrones para acceder a ellos.

1. sumar las filas.
2. sumar las columnas.
3. inicializar el arreglo en 0 (o algún valor especifico)
4. mostrar el arreglo.

Primero definimos algunas constantes y variables usando identificadores generales como fila y columna.

```
const int NUMCOL=10;

const int NUMFIL=10;

int arreglo[NUMFIL][NUMCOL];      //arreglo bidimensional
int columna                       //indice de columna
int fila                          //indice de fila
int total                         //la variable para la suma
```

Sumando filas de arreglos bidimensionales en C++

Suponiendo que queremos sumar la fila numero 3 (cuarta fila) en el arreglo y queremos mostrarlo. Lo haremos fácilmente con la función for

```
   total=0;
3  for(columna=0;columna<NUMCOL;col++){
4     total=total+arreglo[3][columna];
5  }
6
7  cout<<"la suma es "<<total<<endl;
```

El for suma cada columna mientras mantienes fija la fila con el indice 3, sumando cada valor que hay en ella.

Ahora supongamos que queremos sumar la fila 2 y 3, anidamos un for al que ya teníamos ejemplo:

```
total=0;

for(fila=2;fila<4;fila++){
    for(columna=0;columna<NUMCOL;columna++){
        total=total+arreglo[fila][columna];
    }
}
cout<<"el total es "<<total<<endl;
```

el for de afuera controla las filas y el de adentro las columnas.

En la primera iteración se accede a la posición del arreglo: arreglo[2][0], arreglo[2][1], arreglo[2[2] … arreglo[2][NUMCOL-1];

En la segunda accede a: arreglo[3][0], arreglo[3][1], arreglo[3][2] …. arreglo[3][NUMCOL-1];

Si queremos acceder a una porción del arreglo declararemos las siguientes variables (subarreglo)

```
1 int filaFilled         //dato esta en 0.....filaFilled
2 int colFilled          //datos esta en 0 ....colFilled
```

Entonces escribimos el fragmento de código siguiente

```
  total=0;
2
3 for(columna=0;columna<columnaFilled;columna++){
4     for(fila=2;fila<filaFilled;fila++){
5         total=total+arreglo[fila][columna];
6     }
7 }
8 cout<<"el total es "<<total<<endl;
```

Inicializando arreglos bidimensionales en C++

Como con los arreglos unidimensionales podemos inicializar los arreglos bidimensionales en su declaración o usando declaración de asignaciones.

Si el arreglo es corto es simple iniciarlo en su declaración, para iniciar un arreglo de 2 flas y 3 columnas como este

14,4,-5
0, 46 ,7

podemos usar la siguiente declaración

```
int arreglo[2][3]=
{
   {14, 3, -5},
   {0, 46, 7}
};
```

En esta declaración la lista consiste en dos elementos, siendo cada uno una lista de inicialización: la primera guarda los valores 14, 3, -5 en la fila 0 del arreglo; la segunda guarda los valores 0, 46 y 7 en la fila 1.

Inicializar un arreglo en su declaración es impractico si el arreglo es largo por ejemplo uno de 100*100, no necesariamente tienes que listar 10,000 elementos. Si los valores son todos distintos deberías guardarlos en un archivo y entrarlos en el arreglo al mismo tiempo que corre el programa. Si los valores son iguales lo usual es anidar iteraciones en su declaración de asignaciones, el siguiente ejemplo inicializa todos los arreglos en 0.

```
for(fila=0;fila<NUMFIL;fila++)
   for(columna=0;columna<NUMCOL;columna++)
      arreglo[fila][columna]=0;
```

en este caso no importa si accedemos por las filas primero o las columnas, mientras accedamos a todos los elementos del arreglo.

Mostrar arreglos bidimensionales en C++

Si queremos mostrar el arreglo con una fila por fila entonces tendremos otro caso de procesamiento por fila:

```
#include <iomanip>          //para el setw();
.
.
.
for(fila=0;fila<NUMFIL;fila++)
{
   for(columna=0;columna<NUMCOL;columna++)
      cout<<setw(15)<<arreglo[fila][columna];
   cout<<endl;}
```

Este fragmento de código imprime valores del arreglo en columnas de 15 caracteres.

No hay regla que diga que tenemos que imprimir cada fila en una linea, podemos voltear el arreglo e imprimir cada columna en una linea, simplemente cambiando los dos for.

Cuando imprimes un arreglo bidimensional, debes considerar cual orden de presentación tiene mas sentido y como el arreglo encaja en la pagina. Por ejemplo un arreglo de 6 columnas y 100 filas, seria mejor presentado como 6 columnas y 100 lineas de largo.

Referencias

H. (2017, 18 noviembre). *Arreglos bidimensionales en C++*. yosoy.dev. https://yosoy.dev/arreglos-bidimensionales-en-c/#:%7E:text=Un%20arreglo%20de%202%20dimensiones,y%20la%20fila%20del%20arreglo.

F, E. (2018, 5 agosto). *Arreglos bidimensionales*. Solucion ingenieril. http://solucioningenieril.com/programacion_en_c/arreglos_bidimensionales_matrices_o_tablas