Arreglos unidimensionales y bidimensionales

Periodo de verano 2022-2023

## Idea general

Los vectores y las matrices son estructuras de datos muy útiles en una amplia variedad de contextos. Algunos ejemplos de su uso en el día a día podrían incluir:

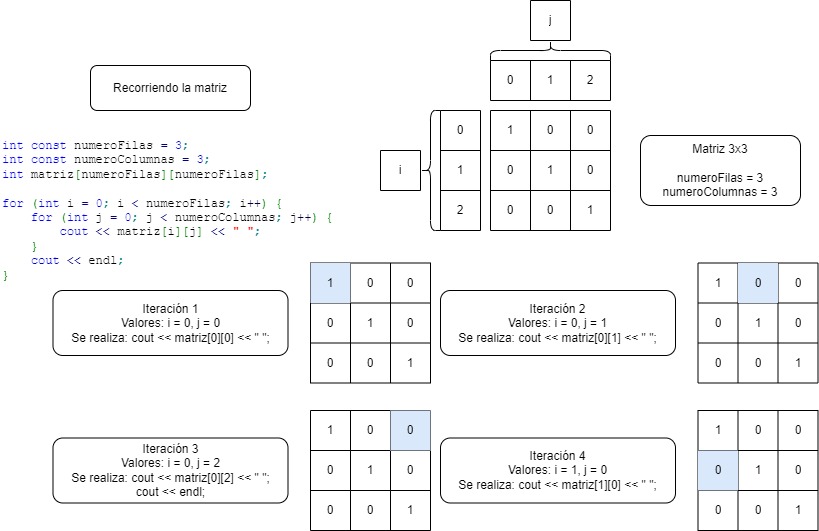
1. Procesamiento de imágenes y vídeos.
2. Procesamiento de datos.
3. Cálculo matemático.
4. Videojuegos.

En general, los vectores y matrices son estructuras de datos muy versátiles y útiles en una amplia variedad de aplicaciones.

Se tratan de un tipo de estructuras de datos con capacidad de acceso aleatorio, que permiten al usuario acceder y modificar elementos individuales de manera eficiente utilizando índices.

## Bidimensionales (matrices)

Tienen dos dimensiones y se pueden pensar como matrices con filas y columnas. Los elementos se pueden acceder a través de dos índices: el índice de fila (denotado por i) y el índice de columna (denotado por j).



### Memoria automática

#### MatrizA.h

// Preámbulo...  
const int numeroFilas = 10;  
const int numeroColumnas = 10;  
  
class MatrizA  
{  
private:  
 int matriz[numeroFilas][numeroColumnas];  
public:  
 MatrizA();  
 int getNumeroFilas() const;  
 int getNumeroColumnas() const;  
 void insertarPosicion(int, int, int);  
 void insertarFinal(int);  
 void eliminarPosicion(int, int);  
 void intercambiar(int, int, int, int);  
 string toString();  
 virtual ~MatrizA() = default;  
};

#### MatrizA.cpp

#include "MatrizA.h"  
  
MatrizA::MatrizA()  
{  
 for (int i = 0; i < numeroFilas; i++) {  
 for (int j = 0; j < numeroColumnas; j++) {  
 matriz[i][j] = INT\_MIN;  
 }  
 }  
}  
  
int MatrizA::getNumeroFilas() const  
{  
 return numeroFilas;  
}  
  
int MatrizA::getNumeroColumnas() const  
{  
 return numeroColumnas;  
}  
  
void MatrizA::insertarPosicion(int fila, int columna, int valor)  
{  
 if (fila < numeroFilas && columna < numeroColumnas) {  
 matriz[fila][columna] = valor;  
 }  
}  
  
void MatrizA::insertarFinal(int valor)  
{  
 for (int i = 0; i < numeroFilas; i++) {  
 for (int j = 0; j < numeroColumnas; j++) {  
 if (matriz[i][j] == INT\_MIN) {  
 matriz[i][j] = valor;  
 return;  
 }  
 }  
 }  
}  
  
void MatrizA::eliminarPosicion(int fila, int columna)  
{  
 if (fila < numeroFilas && columna < numeroColumnas) {  
 matriz[fila][columna] = INT\_MIN;  
 }  
}  
  
void MatrizA::intercambiar(int filaA, int columnaA, int filaB, int columnaB)  
{  
 if (filaA < numeroFilas &&  
 columnaA < numeroColumnas &&  
 filaB < numeroFilas &&  
 columnaB < numeroColumnas) {  
 int aux = matriz[filaA][columnaA];  
 matriz[filaA][columnaA] = matriz[filaB][columnaB];  
 matriz[filaB][columnaB] = aux;  
 }  
}  
  
string MatrizA::toString()  
{  
 stringstream s;  
 s << "\n--> Matriz automatica <--" << endl;  
 for (int i = 0; i < numeroFilas; i++) {  
 for (int j = 0; j < numeroColumnas; j++) {  
 if (matriz[i][j] == INT\_MIN) s << "0 ";  
 else s << matriz[i][j] << " ";  
 }  
 s << endl;  
 }  
 return s.str();  
}

### Memoria dinámica

#### MatrizD.h

// Preámbulo...  
class MatrizD  
{  
private:  
 int numeroFilas;  
 int numeroColumnas;  
 int\*\*\* matriz;  
public:  
 MatrizD(int = 0, int = 0);  
 int getNumeroFilas() const;  
 int getNumeroColumnas() const;  
 void insertarPosicion(int, int, int);  
 void insertarFinal(int);  
 void eliminarPosicion(int, int);  
 void intercambiar(int, int, int, int);  
 string toString();  
 virtual ~MatrizD();  
};

#### MatrizD.cpp

#include "MatrizD.h"  
  
MatrizD::MatrizD(int numeroFilas, int numeroColumnas)  
{  
 this->numeroFilas = numeroFilas;  
 this->numeroColumnas = numeroColumnas;  
 this->matriz = new int\*\* [this->numeroFilas];  
  
 for (int i = 0; i < this->numeroFilas; i++) {  
 this->matriz[i] = new int\* [this->numeroColumnas];  
 }  
  
 for (int i = 0; i < this->numeroFilas; i++) {  
 for (int j = 0; j < this->numeroColumnas; j++) {  
 this->matriz[i][j] = nullptr;  
 }  
 }  
}  
  
int MatrizD::getNumeroFilas() const  
{  
 return this->numeroFilas;  
}  
  
int MatrizD::getNumeroColumnas() const  
{  
 return this->numeroColumnas;  
}  
  
void MatrizD::insertarPosicion(int fila, int columna, int valor)  
{  
 if (fila < numeroFilas && columna < numeroColumnas) {  
 if (!matriz[fila][columna]) matriz[fila][columna] = new int(valor);  
 }  
}  
  
void MatrizD::insertarFinal(int valor)  
{  
 for (int i = 0; i < numeroFilas; i++) {  
 for (int j = 0; j < numeroColumnas; j++) {  
 if (!matriz[i][j]) {  
 matriz[i][j] = new int(valor);  
 return;  
 }  
 }  
 }  
}  
  
void MatrizD::eliminarPosicion(int fila, int columna)  
{  
 if (fila < numeroFilas && columna < numeroColumnas) {  
 if (matriz[fila][columna]) {  
 delete matriz[fila][columna];  
 matriz[fila][columna] = nullptr;  
 }  
 }  
}  
  
void MatrizD::intercambiar(int filaA, int columnaA, int filaB, int columnaB)  
{  
 if (filaA < numeroFilas &&  
 columnaA < numeroColumnas &&  
 filaB < numeroFilas &&  
 columnaB < numeroColumnas) {  
 int\* temp = matriz[filaA][columnaA];  
 matriz[filaA][columnaA] = matriz[filaB][columnaB];  
 matriz[filaB][columnaB] = temp;  
 }  
}  
  
string MatrizD::toString()  
{  
 stringstream s;  
 s << "\n--> Matriz dinamica <--" << endl;  
 for (int i = 0; i < numeroFilas; i++) {  
 for (int j = 0; j < numeroColumnas; j++) {  
 if (!matriz[i][j]) s << "0 ";  
 else s << \*matriz[i][j] << " ";  
 }  
 s << endl;  
 }  
 return s.str();  
}  
  
MatrizD::~MatrizD()  
{  
 for (int i = 0; i < numeroFilas; i++) {  
 for (int j = 0; j < numeroColumnas; j++) {  
 if (matriz[i][j]) delete matriz[i][j];  
 }  
 delete[] matriz[i];  
 }  
 delete[] matriz;  
}