ESCUELA DE INGENIERÍA DE SISTEMAS
DEPARTAMENTO DE COMPUTACIÓN
PROGRAMACIÓN 2

PRÁCTICA DE LABORATORIO 4

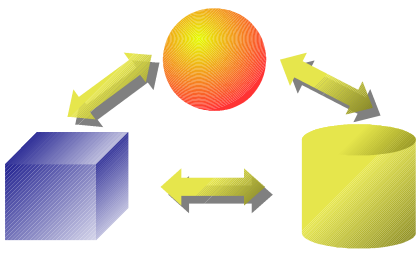
Programación Orientada a Objetos

Contenido

1. Introducción

El Paradigma de Programación Orientado a Objetos es una técnica de programación que usa objetos y sus interacciones para diseñar aplicaciones y buenos programas de computadora. Este paradigma proporciona una forma particular de programar, más cercana a la manera de como expresamos las cosas en la vida real.

En la programación orientada a objetos tenemos que diseñar nuestros progrmas en términos de objetos, propiedades y métodos. Estos conforman los elementos principales de este paradigma. La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.



En la programación convencional los programas se dividen en dos componentes: Procedimientos y Datos. Las estructuras de datos utilizadas en programación son globales o se pasan como parámetros. En esencia los datos se tratan separadamente de los procedimientos. En la programación orientada a objetos se combinan los datos y los procedimientos en una entidad única.

Podemos decir también, que este paradigma es una disciplina de diseño de sofware que facilita la construcción de sistemas a través de componentes individuales llamados clases de objetos.

Objeto:
En la programación orientada a objetos un programa se divide en componentes que contienen procedimientos y datos. Cada componente se considera un objeto. En otras palabras un objeto es una unidad que contiene datos y las funciones que operan sobre esos datos.

Los objetos son entidades que combinan estado, comportamiento e identidad:

> **El estado** está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
>
> **El comportamiento** está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.
>
> **La identidad** es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Atributo:
Los datos que estan contenidos dentro de un objeto se denominan atributos. Podemos definirlos como las características o propiedades que posee un objeto.

Métodos:
Los métodos en la programación orientada a objetos los constituyen las operaciones o procedimientos que operan sobre los atributos de los objetos.

De esta manera, podemos decir que los datos y las funciones se **encapsulan** en una única entidad. Los datos están ocultos y sólo mediante las funciones miembro es posible acceder a ellos.

Clase:
Una Clase es una colección de objetos similares y un objeto es una particulirarización de una definición de una clase. Una Clase es un tipo de dato definido por el usuario que determina las estructuras de datos y las operaciones asociadas con ese tipo. Las clases son las definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas. Cada vez que se construye un objeto de una clase, se crea una instancia de esa clase. En general, los términos objetos e instancias de una clase se pueden utilizar indistintamente.

2. Trabajo Práctico

En esta sección el estudiante será guiado en la confección de un TAD utilizando la programación orientada a objetos. Se tomará como ejemplo el TAD **Matriz**. Al mismo tiempo se le mostrará al estudiante como organizar el código fuente de tal manera de cumplir con un estilo de programación estándar para desarrollos de dimensión considerable.

Implantación del TAD Círculo en C++

Atributos

Los atributos de una matriz son: el número de filas, el número de columnas y por supuesto los valores de la matriz. Los primeros dos atributos los podemos gestionar como enteros y los valores como punto flotante.

Constructores

En el constructor por omisión podemos inicializar a cero las filas y las columnas y dejar la matriz de valores sin espacio de memoria. En el construtor paramétrico podemos solicitar el espacio de memoria necesario pasando como parámetros el número de filas y columnas.

Métodos de Acceso

Dentro de los métodos de acceso podemos tener métodos para obtener el radio el número de filas y el número de columnas. Podemos tener otro método para obtener un valor específico *(i,j)* de la matriz. También, podemos tener un método para mostrar todos los valores de la matriz.

Métodos de Modificación

En los métodos de modificación podemos tener un método para inicializar la matriz, para asignar un valor específico *(i,j)* de la matriz

Métodos Auxiliares
El estudiante debe encargarse de implantar métodos para sumar y multiplicar matrices. También, métodos para hallar el determinante, la matriz inversa y la matriz traspuesta.

2.1. Implantación

El código del TAD será organizado en tres archivos:

Matriz.h: Archivo cabecera contentivo de la clase y los prototipos de los métodos.
Matriz.cpp: Archivo contentivo de la implantación de todos los métodos.
Principal.cpp: Archivo con la función principal (main) donde se harán algunas pruebas del TAD.

Se utilizará un archivo Makefile para compilar todo el código.

1. Edite el archivo **Matriz.h** y agregue el siguiente código:

```
class Matriz {

private:
        int filas, columnas;
        float **elementos;

public:
      //Constructores
      Matriz();
      Matriz(const Matriz&);
      Matriz(const int, const int);
      //Destructor
      ~Matriz();

      //Métodos de acceso
      float Elemento(const int, const int);
      int Filas();
      int Columnas();
      void Mostrar();

      //Métodos de modificación
      void asignarElemento(const int, const int, const float);
      void Inicializar(const float);
      void Leer();

};
```

2. Edite el archivo **Matriz.cpp** y agregue las siguientes líneas:

```
#include "Matriz.h"
#include <iostream>
using namespace std;

//Constructor por omisión
Matriz::Matriz(){
        filas = 0;
        columnas = 0;
        elementos = NULL;
}
//Constructor por copia
Matriz::Matriz(const Matriz& mat){
        int i,j;
        filas = mat.filas;
        columnas = mat.columnas;
        elementos = new float * [filas*columnas];
        for(i=0; i<filas; i++){
                elementos[i] = new float[columnas];
                for(j=0; j<columnas; j++){
                        elementos[i][j] = mat.elementos[i][j];
                }
        }
}
Matriz::Matriz(const int f, const int c){
        int i, j;
        filas = f;
        columnas = c;
        elementos = new float * [filas*columnas];
        for(i=0; i<filas; i++){
                elementos[i] = new float[columnas];
        }
}
//Destructor
Matriz::~Matriz(){
        delete [] elementos;
}
//Métodos de acceso
float Matriz::Elemento(const int i, const int j){
        return this->elementos[i][j];
}
int Matriz::Filas(){
        return filas;
}
```

```
int Matriz::Columnas(){
        return columnas;
}
void Matriz::Mostrar(){
        int i, j;
        for(i=0; i<filas; i++){
                for(j=0; j<columnas; j++){
                        cout<<elementos[i][j]<<" ";
                }
                cout<<endl;
        }
}
//Métodos de modificación
void Matriz::asignarElemento(const int i, const int j,
                                    const float val){
        elementos[i][j] = val;
}
void Matriz::Inicializar(const float valor){
        int i, j;
        for(i=0; i<filas; i++){
                for(j=0; j<columnas; j++){
                        elementos[i][j] = valor;
                }
        }
}
```

3. Edite el archivo **Principal.cpp** y coloque las siguientes instrucciones:

```
/* Para correr este programa debe ejecutar el
 * siguiente comando:
 *
 *       ./matriz
 *
 */

#include "Matriz.h"

int main(){
        Matriz  m1(2, 2);

        m1.Inicializar(2.0);
        m1.Mostrar();

        return 0;
}
```

Las intrucciones para correr este programa están en el comentario ubicado al principio del archivo P**rincipal.cpp**

4. Por último edite un archivo de nombre **Makefile** y coloque las siguientes líneas dentro de él.

```
CC  =   g++
CFLAGS  = -g
OTHERFLAGS = -c

main: Principal Matriz
        $(CC) $(CFLAGS)  Principal.o Matriz.o -o matriz

Principal: Principal.cpp
        $(CC)  $(CFLAGS) $(OTHERFLAGS) Principal.cpp

Matriz: Matriz.cpp
        $(CC)  $(CFLAGS) $(OTHERFLAGS) Matriz.cpp
```

5. Para compilar ejecute el comando

   make

3. Ejercicio

1. Construya un método para sumar matrices de la forma

   $A(m \times n) + B(r \times s)$

2. Construya un método para hallar la traspuesta de la matriz
3. Implante la operación de matriz inversa
4. Construya un método para hallar el determinante de la matriz