Proyecto Integrador PEL + Algebra

Sergio Navarro Manzano
Juan Garde Jimenez
El código está subido en github en el repositorio. https://github.com/Navarro21111/ProyectoIntegradorPel

El trabajo en realizar consiste es realizar un programa en C++ que sea capaz de recibir cualquier ecuación de una recta o un plano y es capaz de ver las posiciones en que están dos rectas, dos planos, tres rectas, una plano y una recta o un punto y un plano. Lo que se ha realizado son una serie de métodos que se explicarán posteriormente.

Para poder realizar el programa hemos necesitado incluir la librería iostream para pedir la ecuación de la recta o del plano y la función stoi para poder pasar el string a integer.

Para el cálculo de las posiciones relativas nos hemos basado en los apuntes del profesor Alberto de Algebra. Y mediante ellos nos hemos basado para la manipulación de de los distintos arrays bidimensionales que utilizamos dependiendo del tipo de comparación que el usuario decida usar. Para ellos dispone de un pequeño menú que aparece en el terminal

```
int main() {

    int n;
    cout << "1 para 2 rectas,"
            "\n2 para 2 planos,"
            "\n3 para plano y recta, "
            "\n4 para 3 planos ",
            cin >> n;

    if (n == 1) {
        dosRectas();

    } else if (n == 2) {
        DosPlanos();

    } else if (n == 3) {
        RectaPlano();
    } else if (n == 4) {
        tresPlanos();
    }
}
```

Como se puede ver en el código hay cuatro opciones y dependiendo cuál se elija sale un segundo para que el programa sepa en qué formato se va a introducir las diferentes ecuaciones.

Para ver las posiciones de dos rectas se utilizó una matriz bidimensional de 3x3 que fue manipulada para poder calcular el determinante de la matriz que generan y el de la ampliada.

```
}
//para el rango de la matriz ampliada
if(detA1!=0||detA2!=0||detA3!=0||detA4!=0){
    rangoAmp=2;
}else{
    rangoAmp=1;
}

if(rangoM==2&&rangoAmp==2){
    cout<<"las dos rectas son secantes";
}else if(rangoM==1&&rangoAmp==2){
    cout<<"las dos rectas son paralelas";
}else{
    cout<<"las dos rectas son coincidentes";

}

  }
```

```
[1][0]);//para calcular el primer determinate de 2x2
[2][0]);//para calcular el segundo determinante de 2x2

A[1][0]);//
A[2][0]);//determinantes
A[1][2]);//de la ampliada
A[2][2]);//
```

Como se puede ver en la primera imagen se hace todas las posibles combinaciones de matrices de rango 2x2 para el determinante. No se observa que pueda ser todos 0 porque eso no sería ni un plano ni una recta por lo tanto sólo tendría sentido usar o ese plano o esa recta.Después de calcular todos los determinantes de la matriz se ve si es de rango uno o dos. En la segunda imagen lo que se realiza es ver si la matriz ampliada es de rango 2 o1. En el último if encadenado lo que se realiza es ver si las dos rectas son secantes, paralelas o coincidentes.

```
int rangoM=0;
int rangoAmp=0;

//los diferentes determinantes necesarios
int det1=matriz[0][0]*matriz[1][1]-(matriz[0][1]*matriz[1][0]);
int det2=matriz[0][1]*matriz[1][2]-(matriz[0][2]*matriz[1][1]);
int det3=matriz[0][0]*matriz[1][2]-(matriz[0][2]*matriz[1][0]);

int detA1=matriz[0][0]*matriz[1][3]-(matriz[0][3]*matriz[1][0]);
int detA2=matriz[0][1]*matriz[1][3]-(matriz[0][3]*matriz[1][1]);
int detA3=matriz[0][2]*matriz[1][3]-(matriz[0][3]*matriz[1][2]);
```

```
//para ver el rango que tienen
if(det1!=0||det2!=0||det3!=0){
    rangoM=2;
}else{
    rangoM=1;
}
if(detA1!=0||detA2!=0||detA3!=0){
    rangoAmp=2;
}else{
    rangoAmp=1;
}

if(rangoM==2&&rangoAmp==2){
```

En estas dos imágenes se puede ver como el método empleado para comparar dos planos. Para ello lo que se recibe un array bidimensional de 2x4 en donde se almacena dos tanto la matriz como la ampliada. Como sucede en el caso anterior lo

que se hace es ver el determinante de mayor grado posible de ambos casos y se ve en el último if encadenado la posición de ambos planos. Pudiendo cortarse, ser paralelos o coincidentes.

```
int det1=matriz[0][0]*matriz[1][1]*matriz[2][2]+
         matriz[1][2]*matriz[0][1]*matriz[2][0]+
         matriz[2][1]*matriz[1][0]*matriz[0][2]-
         (matriz[0][2]*matriz[1][1]*matriz[2][0]+
          matriz[2][1]*matriz[1][2]*matriz[0][0]+
          matriz[1][0]*matriz[0][1]*matriz[2][2]);
int det2=matriz[0][0]*matriz[1][1]-(matriz[0][1]*matriz[1][0]);
int det3=matriz[0][1]*matriz[1][2]-(matriz[0][2]*matriz[1][1]);
int det4=matriz[1][0]*matriz[2][1]-(matriz[1][1]*matriz[2][0]);
int det5=matriz[0][1]*matriz[1][2]-(matriz[0][2]*matriz[1][1]);

//para ver el rango de 3 de la ampliada
int detA1=matriz[0][0]*matriz[1][1]*matriz[2][3]+
          matriz[1][3]*matriz[0][1]*matriz[2][0]+
          matriz[2][1]*matriz[1][0]*matriz[0][3]-
          (matriz[0][3]*matriz[1][1]*matriz[2][0]+
           matriz[2][1]*matriz[1][3]*matriz[0][0]+
           matriz[1][0]*matriz[0][1]*matriz[2][3]);
```

```
//para ver el rango de 3 de la ampliada
int detA1=matriz[0][0]*matriz[1][1]*matriz[2][3]+
          matriz[1][3]*matriz[0][1]*matriz[2][0]+
          matriz[2][1]*matriz[1][0]*matriz[0][3]-
          (matriz[0][3]*matriz[1][1]*matriz[2][0]+
           matriz[2][1]*matriz[1][3]*matriz[0][0]+
           matriz[1][0]*matriz[0][1]*matriz[2][3]);

int detA2=matriz[0][0]*matriz[1][3]*matriz[2][2]+
          matriz[1][2]*matriz[0][3]*matriz[2][0]+
          matriz[2][3]*matriz[1][0]*matriz[0][2]-
          (matriz[0][2]*matriz[1][3]*matriz[2][0]+
           matriz[2][3]*matriz[1][2]*matriz[0][0]+
           matriz[1][0]*matriz[0][3]*matriz[2][2]);

int detA3=matriz[0][3]*matriz[1][1]*matriz[2][2]+
          matriz[1][2]*matriz[0][1]*matriz[2][3]+
          matriz[2][1]*matriz[1][3]*matriz[0][2]-
          (matriz[0][2]*matriz[1][1]*matriz[2][3]+
           matriz[2][1]*matriz[1][2]*matriz[0][3]+
           matriz[1][3]*matriz[0][1]*matriz[2][2]);
//para ver el rango 2
```

En esta serie de imágenes se puede ver el método encargado de ver las posiciones relativas de 3 planos, por lo tanto se ha necesitado crear un array bidimensional de 3x4. Para ello primero hemos calculado el determinante de 3x3 y en el caso de que sea 0 se han calculado todos los posibles determinantes de 2x2. Y lo mismo hemos hecho con la matriz ampliada pero con la diferencia de que hay que ir cambiando la columna por la que se cambian los resultado.

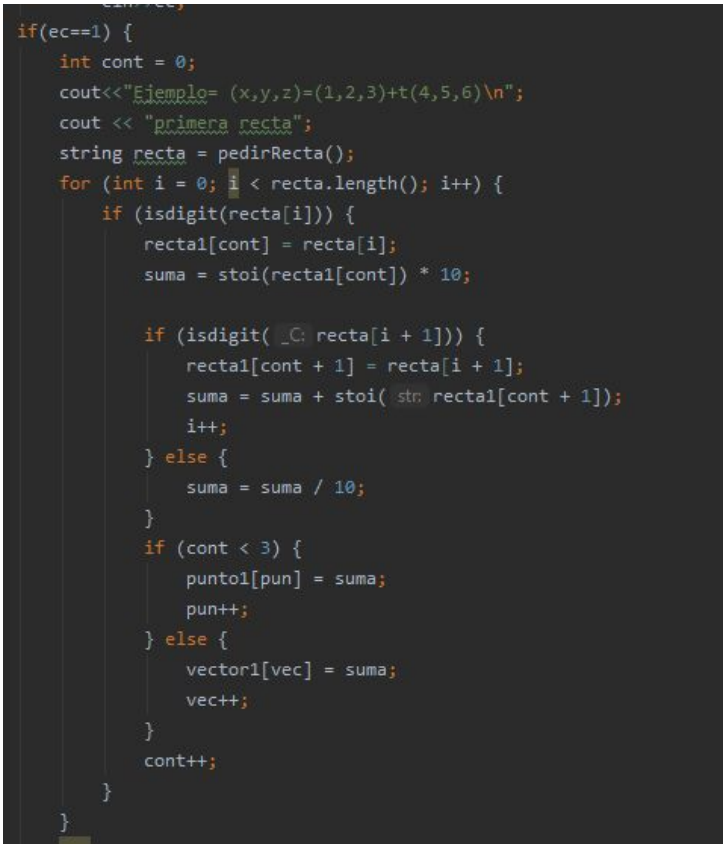
```
if(det1!=0){
    rangoM=3;
}else if(det2!=0|| det3!=0|| det4!=0||det5!=0){
    rangoM=2;
}else{
    rangoM=1;
}

if(detA1!=0||detA2!=0||detA3!=0){
    rangoAmp=3;
}else if(detA4!=0||detA5!=0||detA6!=0||detA7!=0||detA8!=0||detA9!=0){
    rangoAmp=2;
}else{
    rangoAmp=1;
}
```

```
if(rangoM==3&rangoAmp==3){
    cout<<"Los planos se cortan en un punto";
}if(rangoM==2&rangoAmp==3){
    cout<<"Los planos se cortan 2 a 2";
}if(rangoM==2&rangoAmp==2){
    cout<<"Los planos se cortan 2 a 2";
}if(rangoM==1&&rangoAmp==2){
    cout<<"Los planos son paralelos";
}if(rangoM==1&&rangoAmp==1){
    cout<<"Los planos son coincidentes";
}
```

Para ver las posiciones lo que hacemos es comparar los rango de la matriz como de la ampliada y según los resultado se muestran las posiciones.

Para pedir la ecuación lo que hacemos es que el usuario introduzca la ecuación entera por consola y de ahí sacamos los dígitos de dicha ecuación en caso de que el siguiente sea un número también coge ambos dígitos y de ahí se crea la matriz para después manipularla

En el caso de la recta este es el código de la vectorial



```
if(ec==1) {
    int cont = 0;
    cout<<"Ejemplo= (x,y,z)=(1,2,3)+t(4,5,6)\n";
    cout << "primera recta";
    string recta = pedirRecta();
    for (int i = 0; i < recta.length(); i++) {
        if (isdigit(recta[i])) {
            recta1[cont] = recta[i];
            suma = stoi(recta1[cont]) * 10;

            if (isdigit( _C: recta[i + 1])) {
                recta1[cont + 1] = recta[i + 1];
                suma = suma + stoi( str: recta1[cont + 1]);
                i++;
            } else {
                suma = suma / 10;
            }
            if (cont < 3) {
                punto1[pun] = suma;
                pun++;
            } else {
                vector1[vec] = suma;
                vec++;
            }
            cont++;
        }
    }
```

Y este el de las paramétricas.

```
cout<<"Primera Recta\n";
cout<<"x=",cin>>ec1;
cout<<"y=",cin>>ec2;
cout<<"z=",cin>>ec3;
string total=ec1+"b"+ec2+"c"+ec3;
int cont=0;
int
        pun=0;
for (int i = 0; i < total.length(); i++) {
    if (isdigit(total[i])) {
        recta1[cont] = total[i];
        suma = stoi(recta1[cont]);

        if (isdigit( _C: total[i + 1])) {
            recta2[cont + 1] = total[i + 1];
            suma = suma*10 + stoi( str: recta2[cont + 1]);
            i++;
        }
        if(cont%2==0){
            punto1[pun]=suma;
            pun++;
        }
        else{
            vector1[vec]=suma;
            vec++;
        }
        cont++;
    }
}

vec=0;
pun=0;
```

Y en el caso de los planos lo pedimos con tres ecuaciones diferentes vectorial, paramétricas y continua,

El código de la vectorial primero con la función pedir recta pide la ecuación y comprueba si es dígito o no y si hay otro seguido multiplica el primero por 10 y se lo suma al segundo y se crea el vector y el punto.

```
cout<<"primer plano";

string plano=pedirRecta();
for (int i = 0; i < plano.length(); i++) {
    if (isdigit(plano[i])) {
        plano1[cont] = plano[i];
        suma = stoi(plano1[cont]);


        if (isdigit( _C: plano[i + 1])) {
            plano1[cont + 1] = plano[i + 1];
            suma = suma*10 + stoi( str: plano1[cont + 1]);
            i++;
        }

        if(cont<3){
            punto1[cont]=suma;

        }
        else {
            vectorp1[vec]=suma;
            vec++;



        }
        cont++;
    }
}

vec=0;
cont=0;
```

En el caso de las paramétricas el código es este y se cogen los números igual que antes.

```
cout<<"Primer plano\n";
cout<<"x=",cin>>ec1;
cout<<"y=",cin>>ec2;
cout<<"z=",cin>>ec3;
string total=ec1+ec2+ec3;
int cont=0;
int pun=0;
for (int i = 0; i < total.length(); i++) {
    if (isdigit(total[i])) {
        plano1[cont] = total[i];
        suma = stoi(plano1[cont]);

        if (isdigit( _C: total[i + 1])) {
            plano1[cont + 1] = total[i + 1];
            suma = suma*10 + stoi( str: plano1[cont + 1]);
            i++;
        }
        if(cont%3==0){
            punto1[pun]=suma;
            pun++;
        }
        else{
            vectorp1[vec]=suma;
            vec++;
        }
        cont++;
    }
}
```

Y la continua se pide así

```
string plano =pedirRecta();
for (int i = 0; i < plano.length(); i++) {
    if (isdigit(plano[i])) {
        recta1[cont] = plano[i];
        suma = stoi(recta1[cont]);

        if (isdigit( _C: plano[i + 1])) {
            recta1[cont + 1] = plano[i + 1];
            suma = suma*10 + stoi( str: recta1[cont + 1]);
            i++;
        }


        cont++;
    }
}
```

Luego para crear las matrices en el caso de las rectas se coge el vector y el punto y se forma la matriz y en el caso de los planos con la ecuación paramétrica o la vectorial se forma la continua y se crea la matriz y en el caso de la continua se coge directa.

Como conclusión de este trabajo hemos aprendido a trabajar cuando apenas hay comunicación entre los compañeros, debido a asuntos personales por parte de Sergio, debido a que no estaba en

españa en parte del desarrollo. También ha ayudado a aprender a usar tanto matrices bidimensional como ha calcular las posiciones relativas de los planos y rectas.