**Universidad Politécnica de Durango**

Emmanuel Vázquez Pando

**Resolución de ecuaciones por el método de Gauss-Jordán**

**Herramientas Algorítmicas**

**M.T.I Roobed Trejo Mier**

Introducción

En este reporte se expondrá la problemática planteada y el uso de herramientas algorítmicas para la resolución del mismo.

Se verá paso a paso como se fue planteando la solución del problema, la sintaxis utilizada y los ciclos involucrados.

Desarrollo

El problema indicaba la resolución de problemas de ecuaciones lineales por el método de Gauss-Jordán el cual consiste en la formulación de operaciones para llegar a ceros y unos dentro de las columnas con los valores de las variables puede ser X, Y, Z… y en la última columna el valor de cada una de las variables.

Para comenzar a desarrollar el programa fue necesario entender la resolución de este método.

Ejemplo

2x+ 5y= 10

x + 3y= 15

Se trata de elegir pivotes para hacerlos uno y lo demás hacerlo ceros excepto la columna donde están los valores del resultado de las ecuaciones.

Pivotes coloreados de azul

**2** 5 = 10

1 **3** = 15

Se convierten los pivotes a uno por medio de reglas para convertir a todo el renglón en resultados diferentes por ejemplo: Renglon1 (R1)= R1/R1, Renglon2 (R2) =R2-R1.

**2**  5 =10

1 **3** = 15

R1=

R2= R2-R1

**1**  = 5

0 = 10

R2= R2\*2

**1**  0 = -4 5

0 **1** = 20

R1= R1- ( \*R2)

Entonces:

**X**= -45

**Y**= 20

Ahora habiendo resuelto la lógica podemos traducirla a Visual C++ con algunas operaciones simples pero que ayudaran a resolver el problema.

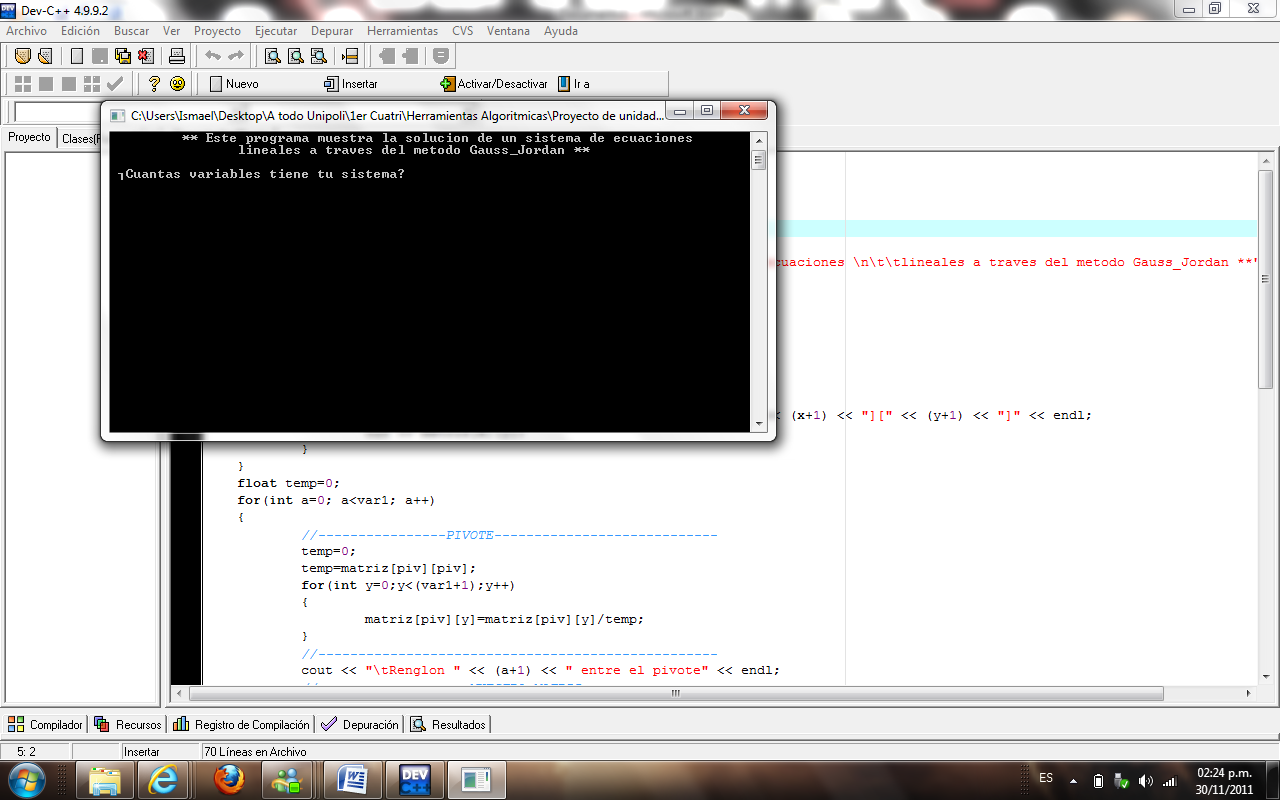
Empecemos pues pidiendo cuantas variables tendrá el sistema de ecuaciones en nuestro programa.

cout << "\n ¿Cuantas variables tiene tu sistema?" << endl;

cin >> leer;

var1 = leer;

float matriz [var1] [var1+1];



En este paso se determina la extensión de la matriz incrementando la columna en 1 para tener la extensión de cuantas columnas será la matriz.

Hay que recordar que el método Gaussiano solo se resuelven ecuaciones de 2\*3, 3\*4, 4\*5… no por ecuaciones de 2\*2, 3\*3, 4\*4.

Después pediremos al usuario los valores de las variables involucradas, obviamente empezando con la primera variable que casi siempre es X y así sucesivamente recorriendo la matriz con un ciclo for.

for (int x=0; x<var1; x++)

{

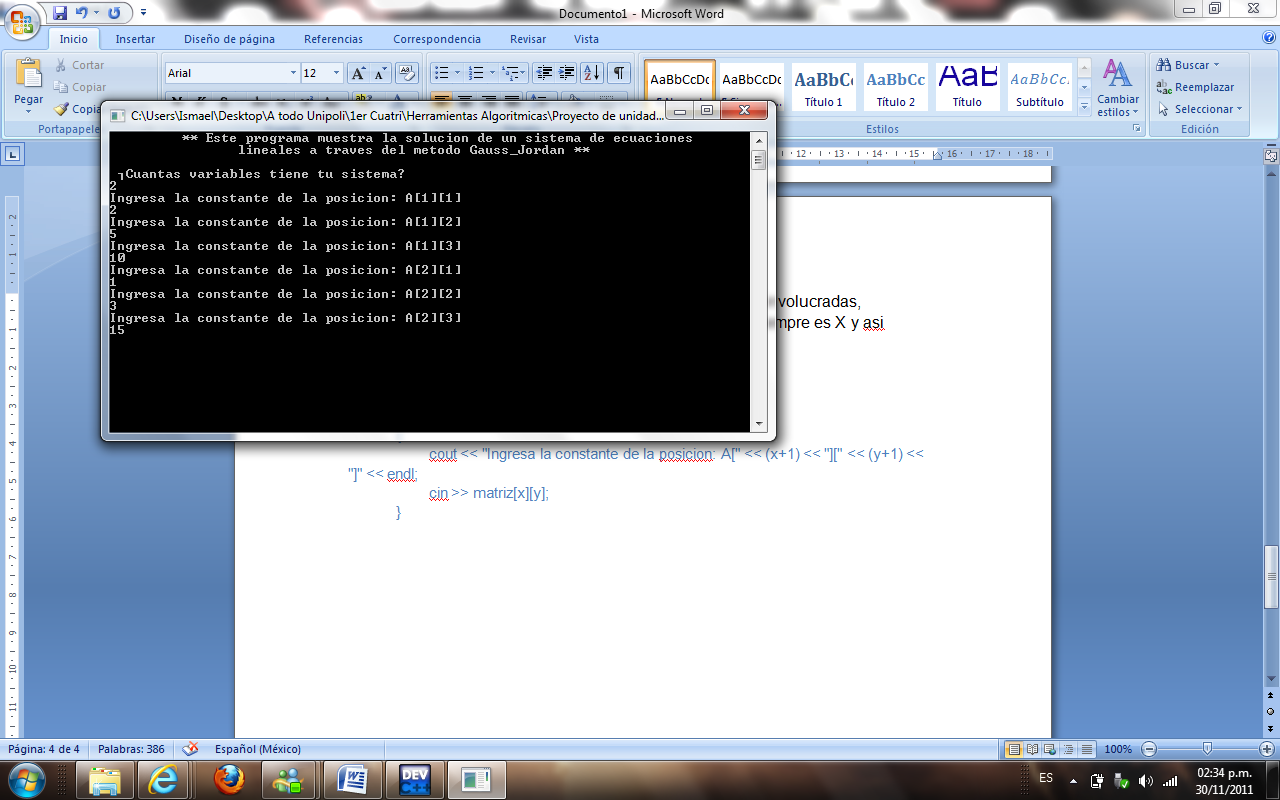
for(int y=0; y<(var1+1); y++)

{

cout << "Ingresa la constante de la posición: A[" << (x+1) << "][" << (y+1) << "]" << endl;

cin >> matriz[x][y];

}



Ahora empieza los métodos matemáticos del programa lo mero bueno.

Determinamos los pivotes por medio del siguiente código:

float temp=0;

for (int a=0; a<var1; a++)

{

//----------------PIVOTE----------------------------

temp=0;

temp=matriz[piv][piv];

for(int y=0;y<(var1+1);y++)

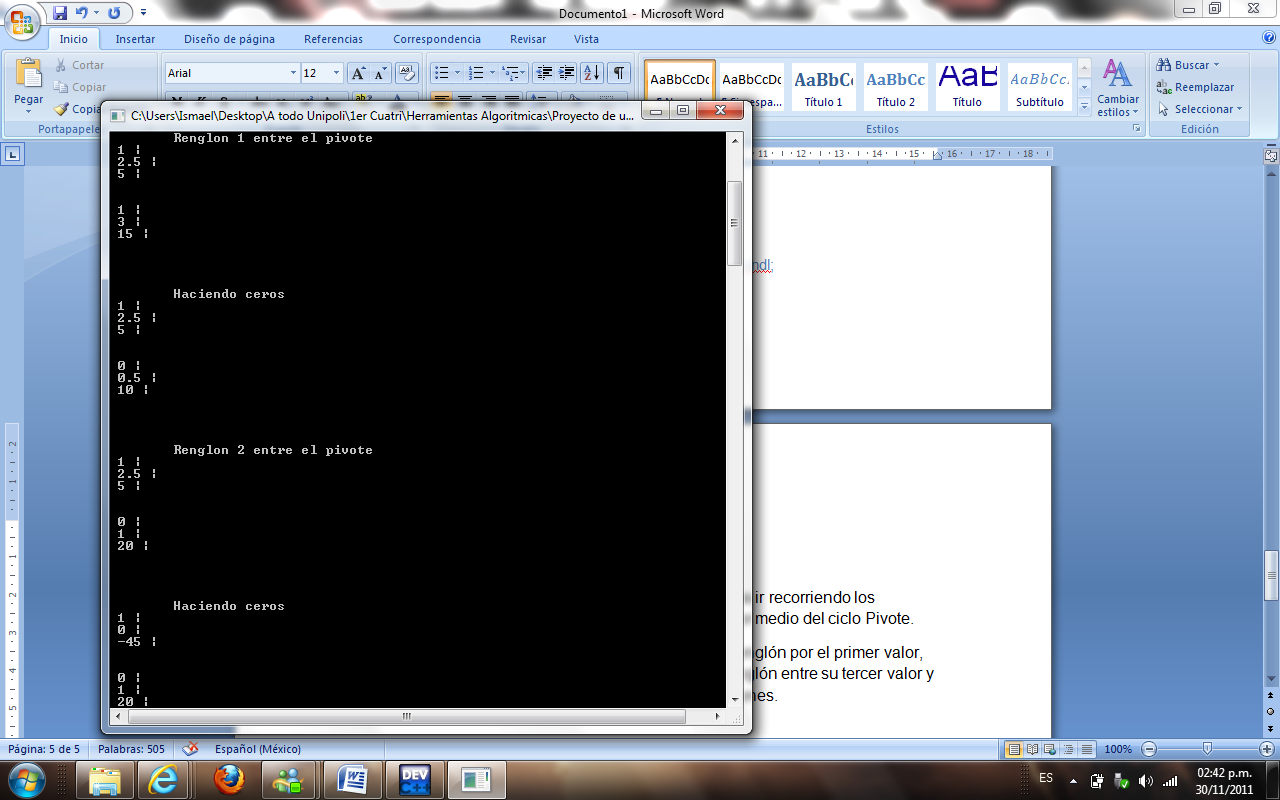
{

matriz[piv][y]=matriz[piv][y]/temp;

}

//--------------------------------------------------

cout << "\tRenglon " << (a+1) << " entre el pivote" << endl;



Todos estos códigos van dentro de un gran ciclo for para ir recorriendo los renglones y encontrar los pivotes, quienes encuentra por medio del ciclo Pivote.

Por ejemplo para el primer renglón va a dividir todo el renglón por el primer valor, el segundo renglón entre su segundo valor, el tercer renglón entre su tercer valor y así según sea el tamaño del nuestro sistema de ecuaciones.

Después de haber hecho la primer división el programa vuelve a ceros los demás términos excepto la última columna donde van los resultados.

Además de mostrar lo que quedo de la matriz después de esas operaciones.

//-------------------MUESTRA MATRIZ-----------------

for(int x=0;x<var1;x++)

{

for (int y=0;y<(var1+1);y++)

cout << " " << matriz[x][y] << " |" << endl;

cout << "\n" << endl;

}

//--------------------------------------------------

cout << "\n" << endl;

cout << "\tHaciendo ceros" << endl;

//----------------HACER CEROS-----------------------

for(int x=0;x<var1;x++)

{

if(x!=piv)

{

float c=matriz[x][piv];

for(int z=0;z<(var1+1);z++)

matriz[x][z]=((-1\*c)\*matriz[piv][z])+matriz[x][z];

}

}

//--------------------------------------------------

//-------------------MUESTRA MATRIZ-----------------

for(int x=0;x<var1;x++)

{

for(int y=0;y<(var1+1);y++)

cout << " " << matriz[x][y] << " |" << endl;

cout << "\n" << endl;

}

//--------------------------------------------------

cout << "\n" << endl;

piv++;

El condicional que detecta los pivotes y evita que se vuelvan ceros:

if(x!=piv)

{

float c=matriz[x][piv];

for(int z=0;z<(var1+1);z++)

matriz[x][z]=((-1\*c)\*matriz[piv][z])+matriz[x][z];

}

y dice que si la variable en x es diferente al pivote entonces se hará:

matriz[x][z]=((-1\*c)\*matriz[piv][z])+matriz[x][z];

Para recorrer los pivotes al último del ciclo se va aumentando la variable piv, para que continúe con el siguiente valor del renglon sucesivo.

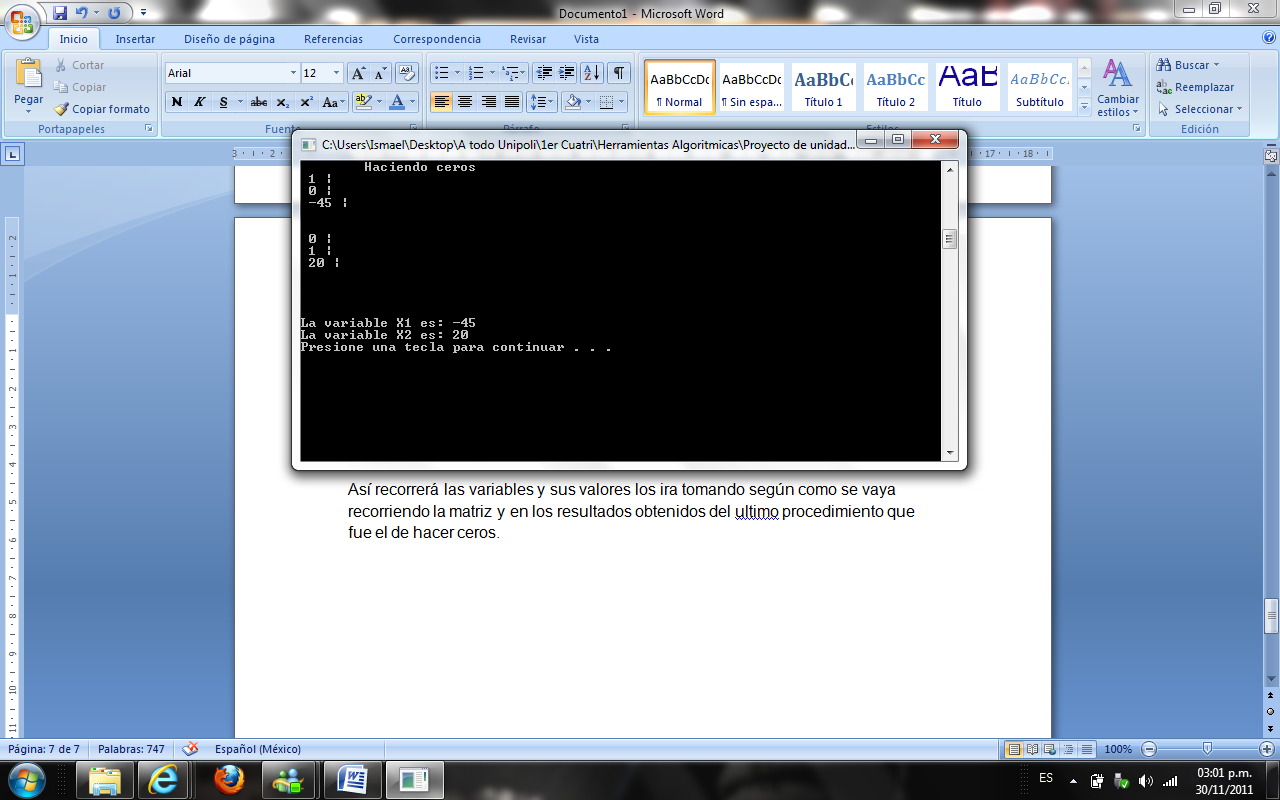
Para saber los resultados de nuestras variables haremos un ciclo for

for(int x=0;x<var1;x++)

cout << "La variable X" << (x+1) << " es: " << matriz[x][var1] << endl;

system("PAUSE");

Así recorrerá las variables y sus valores los ira tomando según como se vaya recorriendo la matriz y en los resultados obtenidos del último procedimiento que fue el de hacer ceros.



Indicara solo x1, x2, xn, claro que serán interpretados como las variables que se fueron introduciendo en orden, en este caso primero fue el valor de X después el de Y, al final el resultado pues era una ecuación de 2\*3.

Entonces se sabrá que X1= x y X2= y.

x= -45.

y= 20.

Que es el resultado obtenido anteriormente en el ejemplo.

Conclusión

Me agrado mucho este curso y mas el proyecto final, realmente no tenía idea de cómo organizar un algoritmo para que me resolviera un tipo de problema así pero con orientación y asesoría de gente que sabe y vive de esto, pude comprender y utilizar los conocimientos aprendidos por ejemplo, haber aplicado el ciclo for y saber cómo extender una matriz.

Buen proyecto y lo mejor de todo es que pude comprender como desarrollar un algoritmo, tal vez no tan complejo pero que si necesite mucha lógica.