UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA AGRÍCOLA
DEPTO. DE AGROINDUSTRIAS
Juan Carlos Sandoval Avendaño

Ejercicios Resueltos Lenguaje de Programacion
Vectores y Matrices
Enero de 2020

1) Escriba un programa en C que calcule y muestre por pantalla la traspuesta de una matriz cuadrada de orden 2 con elementos enteros ingresados por el usuario.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  const n=2;
  int a[n][n], at[n][n];
  int i, j;

  for(i=0;i<=n-1;i++)
   for(j=0;j<=n-1;j++)
    {
      printf("\nA(%i,%i) = ",i+1,j+1);
      scanf("%i",&a[i][j]);
    }

  for(i=0;i<=n-1;i++)
   for(j=0;j<=n-1;j++)
    {
      at[i][j]=a[j][i];
    }
  printf("\n\t\Matriz Traspuesta\n");
  for(i=0;i<=n-1;i++)
```

```
 {
  printf("\n");
  for(j=0;j<=n-1;j++)
   {
    printf("\t%i\t  ",at[i][j]);
   }
 }
 printf("\n\n");
 system("PAUSE");
 return 0;
}
```

❑

2) Escriba un programa en C que muestre por pantalla la inversa de una matriz $A = (a_{ij}) \in \mathcal{M}_2(\mathbb{R})$ cuyos elementos son ingresados por el usuario. El programa debe considerar todos los casos posibles.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int n=2;
 float a[n][n], ainv[n][n], Det;
 int i, j;

 for(i=0;i<=n-1;i++)
  for(j=0;j<=n-1;j++)
   {
    printf("\nA(%i,%i) = ",i+1,j+1);
    scanf("%f",&a[i][j]);
   }

 Det=a[0][0]*a[1][1]-a[0][1]*a[1][0];
 if(Det!=0.0)
  {
   printf("\n\t\tInversa\n");
   ainv[0][0]=a[1][1]/Det;
```

```
    ainv[0][1]=-a[0][1]/Det;
    ainv[1][0]=-a[1][0]/Det;
    ainv[1][1]=a[0][0]/Det;
    for(i=0;i<=n-1;i++)
    {
     printf("\n");
     for(j=0;j<=n-1;j++)
      {
        printf("\t%0.4f\t",ainv[i][j]);
      }
    }
  }
  else printf("\n\nLa inversa no existe\n\n");

  printf("\n\n");

  system("PAUSE");
  return 0;
}
```

3) Escriba un programa en C que muestre por pantalla la solución del sistema

$$a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 = b_1$$
$$a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 = b_2$$
$$a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 = b_3$$

usando el método de Cramer.
El usuario es quien ingresa los coeficientes $a_{i\,j}$ y b_i con $i = 1, 2, 3$ y $j = 1, 2, 3$.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int n=3;
  int i,j;
  float a[n][n], b[n];
  float DetA, DetAp, DetAn;
```

```
float x1p, x1n, x2p, x2n, x3p, x3n, x1, x2, x3;

for(i=0;i<=n-1;i++)
 for(j=0;j<=n-1;j++)
  {
   printf("\na(%i,%i) = ",i+1,j+1);
   scanf("%f",&a[i][j]);
  }
for(i=0;i<=n-1;i++)
 {
   printf("\nb(%i) = ",i+1);
   scanf("%f",&b[i]);
  }
DetAp=a[0][0]*a[1][1]*a[2][2]+a[0][1]*a[1][2]*a[2][0]+a[1][0]*a[2][1]*a[0][2];
DetAn=a[2][0]*a[1][1]*a[0][2]+a[2][1]*a[1][2]*a[0][0]+a[1][0]*a[0][1]*a[2][2];
DetA=DetAp-DetAn;
if(DetA!=0.0)
 {
   x1p=b[0]*a[1][1]*a[2][2]+b[2]*a[0][1]*a[1][2]+b[1]*a[2][1]*a[0][2];
   x1n=b[2]*a[1][1]*a[0][2]+b[0]*a[2][1]*a[1][2]+b[1]*a[0][1]*a[2][2];

   x2p=b[1]*a[0][0]*a[2][2]+b[0]*a[2][0]*a[1][2]+b[2]*a[1][0]*a[0][2];
   x2n=b[1]*a[2][0]*a[0][2]+b[2]*a[0][0]*a[1][2]+b[0]*a[1][0]*a[2][2];

   x3p=b[2]*a[1][1]*a[0][0]+b[1]*a[0][1]*a[2][0]+b[0]*a[2][1]*a[1][0];
   x3n=b[0]*a[1][1]*a[2][0]+b[1]*a[2][1]*a[0][0]+b[2]*a[0][1]*a[1][0];

   x1=(x1p-x1n)/DetA;
   x2=(x2p-x2n)/DetA;
   x3=(x3p-x3n)/DetA;

   printf("\nx1 = %0.2f / %0.2f ~= %0.2f\n",(x1p-x1n),DetA,x1);
   printf("\nx2 = %0.2f / %0.2f ~= %0.2f\n",(x2p-x2n),DetA,x2);
   printf("\nx3 = %0.2f / %0.2f ~= %0.2f\n",(x3p-x3n),DetA,x3);
 }
else printf("\n\nEl sistema no posee solucion\n\n");
printf("\n\n");

system("PAUSE");
return 0;
```

} ❑

4) Escriba un programa en C que calcule y muestre por pantalla la norma de una matriz $A \in \mathcal{M}_{m\times n}(\mathbb{R})$, donde el orden de la matriz y los elementos de ésta son ingresados por el usuario.

(Indicación: $\| A \| = \sqrt{Traza(A^T \cdot A)}$)

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 unsigned int m, n;
 float Traza=0.0;
 printf("\nIngrese el orden de la matriz A\n");
 printf("Ingrese el numero de filas m : ");
 scanf("%u",&m);
 printf("Ingrese el numero de columnas n : ");
 scanf("%u",&n);

 float A[m][n], AT[m][n], C[n][n];
 int i, j, k;

 for (i=0;i<=m-1;i++)
  for (j=0;j<=n-1;j++)
   {
    printf("A(%d,%d) = ",i+1,j+1);
    scanf("%f",&A[i][j]);
   }

 for (i=0;i<=m-1;i++)
  for (j=0;j<=n-1;j++)
   {
    AT[i][j]=A[j][i];
```

```
   }

  for (i=0;i<=m-1;i++)
   for (j=0;j<=n-1;j++)
    {
      C[i][j]=0.0;
      for (k=0;k<=n;k++)
       C[i][j]=C[i][j]+AT[i][k]*A[k][j];
    }

   for (i=0;i<=n-1;i++)
    Traza = Traza + C[i][i];

  printf("\n\nLa norma de la matriz A es %0.1f\n\n",sqrt(Traza));

  system("PAUSE");
  return 0;
}
```

5) Escriba un programa en C que muestre por pantalla la solución del sistema

$$a_{11}\,x + a_{12}\,y = b_1$$
$$a_{21}\,x + a_{22}\,y = b_2$$

usando el método de Cramer.
El usuario es quien ingresa los coeficientes $a_{i\,j}\,;\; i, j = 1, 2$, y $b_i;\; i = 1, 2.$

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  const n=2;
  float A[n][n], b[n], x[n];
  int i, j;
  float DetA;

  printf("\n\nIngrese los elementos de A\n\n");
```

```
 for (i=0;i<=n-1;i++)
  for (j=0;j<=n-1;j++)
  {
   printf("A(%d,%d)=",i+1,j+1);
   scanf("%f",&A[i][j]);
  }

  for (i=0;i<=n-1;i++)
  {
   printf("b(%d)=",i+1);
   scanf("%f",&b[i]);
  }

  DetA=A[0][0]*A[1][1]-A[0][1]*A[1][0];

  if (DetA!=0.0)
  {
   x[0]=(b[0]*A[1][1]-b[1]*A[0][1])/DetA;
   x[1]=(b[1]*A[0][0]-b[0]*A[1][0])/DetA;
   for (i=0;i<=n-1;i++)
    printf("\nx(%d)= %0.1f\n\n",i+1,x[i]);
  }
  else
  printf("\nEl sistema no puede resolverse usando Cramer\n\n");

  system("PAUSE");
  return 0;
}
```

6) Use C para determinar si un vector de $\mathbb{R}^n$ es unitario o no. Las componentes del vector son ingresadas por el usuario, con n un número natural aleatorio no mayor a 20.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
```

```
int main()
{
 srand(time(NULL));
 int n, i;
 n=rand()%20+1;
 float v[n], suma=0.0, norma;

 for(i=1;i<=n;i++)
  {
    printf("\nv[%i] = ",i);
    scanf("%f",&v[i]);
  }
 for(i=1;i<=n;i++)
  suma=suma+v[i]*v[i];
 norma=sqrt(suma);

 if(norma==1.0) printf("\nEl vector es unitario\n");
 else printf("\nEl vector no es unitario\n");

 system("PAUSE");
 return 0;
}
```

7) Escriba un programa en C que muestre por pantalla el producto escalar triple $< \boldsymbol{a}, \boldsymbol{b} \times \boldsymbol{c} >$, donde las componentes de los vectores son ingresadas por el usuario.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 float a[3], b[3], c[3];
 int i;
 float Prod1, Prod2, Prod3, ProdEscalarTriple;
 printf("\nIngrese las componentes del vector a\n");
```

```
 for(i=1;i<=3;i++)
  {
    printf("\n a(%i) = ",i);
    scanf("%f",&a[i]);
  }
 printf("\nIngrese las componentes del vector b\n");
 for(i=1;i<=3;i++)
  {
    printf("\n b(%i) = ",i);
    scanf("%f",&b[i]);
  }
 printf("\nIngrese las componentes del vector c\n");
 for(i=1;i<=3;i++)
  {
    printf("\n c(%i) = ",i);
    scanf("%f",&c[i]);
  }
 Prod1=b[2]*c[3]-b[3]*c[2];
 Prod2=b[3]*c[1]-b[1]*c[3];
 Prod3=b[1]*c[2]-b[2]*c[1];
 ProdEscalarTriple=a[1]*Prod1+a[2]*Prod2+a[3]*Prod3;
 printf("\nEl producto escalar triple <a , b x c> es %0.2f\n",ProdEscalarTriple);

 system("PAUSE");
 return 0;
}
```

Observaciones:

1) Sean $\boldsymbol{a} = [a_1, a_2, a_3]$, $\boldsymbol{b} = [b_1, b_2, b_3]$ y $\boldsymbol{c} = [c_1, c_2, c_3]$

$$\boldsymbol{b} \times \boldsymbol{c} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix} = (b_2c_3 - b_3c_2)\,\boldsymbol{i} + (b_3c_1 - b_1c_3)\,\boldsymbol{j} + (b_1c_2 - b_2c_1)\,\boldsymbol{k} =$$

$$[b_2c_3 - b_3c_2, b_3c_1 - b_1c_3, b_1c_2 - b_2c_1]$$

Luego el producto vectorial triple, en términos de las componentes de los vectores participantes, es:

$$< \boldsymbol{a}, \boldsymbol{b} \times \boldsymbol{c} > \; = \; < [a_1, a_2, a_3],\; [b_2c_3 - b_3c_2, b_3c_1 - b_1c_3, b_1c_2 - b_2c_1] > \; =$$

$a_1(b_2c_3 - b_3c_2) + a_2(b_3c_1 - b_1c_3) + a_3(b_1c_2 - b_2c_1)$

2) Las variables $\mathrm{Prod1}$, $\mathrm{Prod2}$, y $\mathrm{Prod3}$ almacenan las componentes del vector $\boldsymbol{b} \times \boldsymbol{c}$ ❑

8) Sean $\boldsymbol{a}$ y $\boldsymbol{b}$ dos vectores no nulos de $\mathbb{R}^n$. Se llama proyección de $\boldsymbol{b}$ sobre $\boldsymbol{a}$ al vector:

$$proy_{\boldsymbol{a}}\,\boldsymbol{b} = \frac{<\boldsymbol{a},\boldsymbol{b}>}{\|\boldsymbol{a}\|^2}\,\boldsymbol{a}$$

Escriba un programa en C que muestre por pantalla la proyección de $\boldsymbol{b}$ sobre $\boldsymbol{a}$, donde $\boldsymbol{a}$ y $\boldsymbol{b}$ son ingresados por el usuario al igual que el valor de n.

Código Fuente

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
 int n;
 printf("\nIngrese n : ");
 scanf("%i",&n);
 float a[n], b[n];
 int i;
 int Nuloa=0, Nulob=0;
 float ProdInt=0.0, suma=0.0, normaa;

 printf("\nIngrese las componentes del vector a\n");
 do
 {
 for(i=1;i<=n;i++)
  {
    printf("\n a(%i) = ",i);
    scanf("%f",&a[i]);
    if(a[i]!=0.0) Nuloa=1;
```

```
 }
 }
 while(Nuloa==0);

 printf("\nIngrese las componentes del vector b\n");
 do
 {
 for(i=1;i<=n;i++)
  {
    printf("\n b(%i) = ",i);
    scanf("%f",&b[i]);
    if(b[i]!=0.0) Nulob=1;
  }
 }
 while(Nulob==0);

 for(i=1;i<=n;i++)
  ProdInt=ProdInt+a[i]*b[i];

 for(i=1;i<=n;i++)
  suma=suma+a[i]*a[i];
 normaa=suma;

 for(i=1;i<=n;i++)
  printf("\nproy(%i) = %0.2f\n",i,ProdInt/normaa*a[i]);

 system("PAUSE");
 return 0;
}
```

Observaciones:

1) En este ejercicio se usan dos variables banderín (flag) que nos permiten detectar si el usuario ha ingresado un vector nulo; en tal caso se le solicita al usuario el reingreso del vector.

2) La variable ProdInt almacena el producto interior $< \boldsymbol{a}, \boldsymbol{b} >$ y la variable normaa almacena $\| \boldsymbol{a} \|^2$ ❑