Algoritmos y Estructura de Datos I

Trabajo Práctico N°1

Aclaración: la librería “vectores_y_matrices” se mantiene igual en todos los ejercicios (presente a partir del ejercicio 3).

1) Elaborar un algoritmo que lea un vector, busque el mayor elemento en valor absoluto y muestre el resultado.

main.py

```
1  from algo1 import *
2
3  def calcular_mayor(vector,longitud,menor):
4    for i in range(0,longitud):
5      if (abs(vector[i])>menor):
6        mayor=abs(vector[i])
7    return mayor
8
9  longitud=input_int("Ingrese la longitud del vector deseado: ")
10 vector=Array(longitud,0)
11 menor=0
12
13 print("A continuación, ingrese los elementos del vector")
14 for i in range(0,longitud):
15   vector[i]=input_int("")
16   menor=menor-vector[i]
17
18 mayor=calcular_mayor(vector,longitud,menor)
19 print("El mayor elemento del vector es: ",mayor)
```

Link repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej1#main.py

2) Elaborar un algoritmo que lea dos vectores, verifique si tienen la misma dimensión y los sume en un nuevo vector. Calcule la norma cuadrática de este último vector. Muestre el vector resultado y su norma cuadrática.

main.py

```
from algo1 import *
import math

def leer_vector(vector,n):
  for i in range(0,n):
    vector[i]=input_real(f"Ingrese el elemento {i} del vector: ")
  return vector

def sumar_vectores(vector1,vector2,n):
  vector3=Array(n,0.0)
  for i in range(0,n):
    vector3[i]=vector1[i]+vector2[i]
  return vector3

def mostrar_vector(vector,n):
  for i in range(0,n):
    if (i==0):
      print("[",vector[i],end="; ")
    elif(i==n-1):
      print(vector[i],"]",end=" ")
    else:
      print(vector[i],end="; ")
  print(" ")

def norma_cuadratica(vector,n):
  suma=0
  for i in range(0,n):
    suma+=math.pow(vector[i],2)
  suma=math.sqrt(suma)
  return suma

bandera=False

while (bandera==False):
  dimension1=input_int("Ingrese el tamaño del vector 1: ")
  dimension2=input_int("Ingrese el tamaño del vector 2: ")
  if (dimension1<=1) or (dimension2<=1):
    print("Las dimensiones de los vectores deben ser mayor a 1")
  else:
    if (dimension1==dimension2):
      bandera=True
    else:
      print("Las dimensiones de los vectores deben ser iguales")

vector1=Array(dimension1,0.0)
vector2=Array(dimension1,0.0)

print("A continuación, ingrese los elementos del vector 1: ")
vector1=leer_vector(vector1,dimension1)
print("A continuación, ingrese los elementos del vector 2: ")
vector2=leer_vector(vector2,dimension1)

vectorSuma= sumar_vectores(vector1,vector2,dimension1)
norma=norma_cuadratica(vectorSuma,dimension1)

print("El vector resultado de la suma de los 2 vectores es:")
mostrar_vector(vectorSuma,dimension1)
print("La norma cuadrática del vector suma es: ",round(norma))
```

Link repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej2#main.py

3) **Elaborar un algoritmo que lea una matriz y un vector, y que verifique si es posible la multiplicación. En el caso de ser posible realice la operación correspondiente, caso contrario, que muestre el mensaje "dimensiones incorrectas".**

main.py

```
from algo1 import *
from vectores_y_matrices import *

def multiplicacion(vector,matriz,n,m,resultado):
  for i in range(0,n):
    res=0
    for j in range(0,m):
      res+= vector[j] * matriz[i][j]
      resultado[i]=res
  return resultado

bandera=False

while(bandera==False):
  dimensionV=input_int("Ingrese la dimension del vector: ")
  filasM=input_int("Ingrese la cantidad de filas de la matriz: ")
  columnasM=input_int("Ingrese la cantidad de columnas de la matriz: ")
  if (dimensionV<=1) or (filasM<=1) or (columnasM<=1):
    print("Las dimensiones deben ser mayor a 1")
  else:
    if (dimensionV!=columnasM):
      print("Dimensiones incorrectas")
    else:
      bandera=True

vector=Array(dimensionV,0.0)
matriz=Array(filasM,Array(columnasM,0.0))
vectorR=Array(columnasM,0.0)

#Se llenan y muestran el vector y la matriz segun lo que ingrese el usuario
leer_vector(vector,dimensionV)
mostrar_vector(vector,dimensionV)
leer_matriz(matriz,filasM,columnasM)
mostrar_matriz(matriz,filasM,columnasM)

vectorR=multiplicacion(vector,matriz,filasM,columnasM,vectorR)
print("La multiplicación da como resultado: ")
mostrar_vector(vectorR,columnasM)

```

vectores_y_matrices.py

```python
from algo1 import *

def leer_vector(vector,n):
  for i in range(0,n):
    vector[i]=input_real(f"Ingrese el elemento {i} del vector: ")
  return vector

def mostrar_vector(vector,n):
  if n==1:
    print(f"[ {vector[0]} ]")
  else:
    for i in range(0,n):
      if (i==0):
        print("[",vector[i],end="; ")
      elif(i==n-1):
        print(vector[i],"]",end=" ")
      else:
        print(vector[i],end="; ")
    print(" ")

def leer_matriz(matriz,n,m):
  for i in range(0,n):
    for j in range(0,m):
      matriz[i][j]=input_real(f"Ingrese el elemento ({i}, {j}) de la matriz: ")
  return matriz

def mostrar_matriz(matriz,n,m):
  for i in range(0,n):
    print(end="|")
    for j in range(0,m):
      if (j!=m-1):
        print(matriz[i][j],end="; ")
      else:
        print(matriz[i][j],end="")
    print(end="|")
    print(" ")
  print(" ")

```

Link Repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej3#main.py

4) **Elaborar un algoritmo que lea dos matrices, calcule la diferencia de las mismas y almacene el resultado en una tercera matriz.**

main.py

```python
from algo1 import *
from vectores_y_matrices import *

def resta_matrices(matriz1,matriz2,resultado,n,m):
  for i in range(0,filas):
    for j in range(0,columnas):
      resultado[i][j]= matriz1[i][j] - matriz2[i][j]
  return resultado

```

```
bandera=False
while (bandera==False):
  filas=input_int("Ingrese la cantidad de filas de las 2 matrices: ")
  columnas=input_int("Ingrese la cantidad de columnas de las 2 matrices: ")
  if filas<=0 or columnas<=0:
    print("Las dimensiones deben ser mayor a 0")
  else:
    bandera=True

matriz1=Array(filas,Array(columnas,0.0))
matriz2=Array(filas,Array(columnas,0.0))
matrizR=Array(filas,Array(columnas,0.0))

print("A continuación, ingrese la primera matriz:")
leer_matriz(matriz1,filas,columnas)
mostrar_matriz(matriz1,filas,columnas)
print("Ahora llene la segunda matriz:")
leer_matriz(matriz2,filas,columnas)
mostrar_matriz(matriz2,filas,columnas)

matrizR=resta_matrices(matriz1,matriz2,matrizR,filas,columnas)
print("La resta de las 2 matrices ingresadas es igual a:")
mostrar_matriz(matrizR,filas,columnas)

```

Link repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej4#main.py

5) **Elaborar un algoritmo que lea una matriz y determine si es triangular superior. En caso afirmativo el algoritmo debe calcular el determinante de dicha matriz.**

main.py

```
from algo1 import *
from vectores_y_matrices import *

def triangular_superior(matriz,n,m):
  triangular=False
  for i in range(0,n):
    for j in range(0,m):
      if i>j:
        if matriz[i][j]==0:
          triangular=True
        else:
          triangular=False
          return triangular
  return triangular

def determinante(matriz,n,m):
  resultado=1
  for i in range(0,n):
    for j in range(0,m):
      #diagonal principal
      if(i==j):
        resultado=matriz[i][j]*resultado
  return resultado
```

```
24
25  bandera=False
26  while (bandera==False):
27    filas=input_int("Ingrese las filas de la matriz: ")
28    columnas=input_int("Ingrese las columnas de la matriz: ")
29    if filas<=0 or columnas<=0:
30      print("Las dimensiones deben ser mayor a 0")
31    else:
32      if filas!=columnas:
33        print("La matriz debe ser cuadrada (dimensiones iguales)")
34      else:
35        bandera=True
36
37  matriz=Array(filas,Array(columnas,0.0))
38  leer_matriz(matriz,filas,columnas)
39  mostrar_matriz(matriz,filas,columnas)
40  triangular=triangular_superior(matriz,filas,columnas)
41
42  if triangular==True:
43    print("La matriz es triangular superior")
44    det=determinante(matriz,filas,columnas)
45    print("El determinante de la matriz es: ",det)
46  else:
47    print("La matriz NO es triangular superior")
48
```

Link Repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej5#main.py

Ejercicios Propuestos:

6) **Elaborar un algoritmo que lea dos vectores, verifique si tienen la misma dimensión y obtenga el producto escalar de los mismos. Muestre el resultado.**

main.py ×

```
1  from algo1 import *
2  from vectores_y_matrices import *
3
4  def prod_escalar(vector1,vector2,n):
5    resultado=0
6    for i in range(0,n):
7      resultado+=vector1[i]*vector2[i]
8    return resultado
9
10 bandera=False
11 while(bandera==False):
12   dimension1=input_int("Ingrese la dimension del vector 1: ")
13   dimension2=input_int("Ingrese la dimension del vector 2: ")
14   if dimension1<=0 or dimension2<=0:
15     print("Las dimensiones deben ser mayor a 0")
16   else:
17     if dimension1!=dimension2:
18       print("Las dimensiones deben ser iguales")
19     else:
20       bandera=True
```

```
21
22 vector1=Array(dimension1,0.0)
23 vector2=Array(dimension1,0.0)
24 leer_vector(vector1,dimension1)
25 leer_vector(vector2,dimension1)
26 mostrar_vector(vector1,dimension1)
27 mostrar_vector(vector2,dimension1)
28
29 prodEscalar=prod_escalar(vector1,vector2,dimension1)
30 print("El producto escalar entre los 2 vectores es: ",prodEscalar)
31
```

Link Repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej6#main.py

8) Elaborar un algoritmo que lea una matriz y determine si es triangular inferior. En caso afirmativo el algoritmo debe calcular la matriz transpuesta de la misma.

main.py

```
1  from algo1 import *
2  from vectores_y_matrices import *
3
4  def triangular_inferior(matriz,n,m):
5      triangular=False
6      for i in range(0,n):
7          for j in range(0,m):
8              if i<j:
9                  if matriz[i][j]==0:
10                     triangular=True
11                 else:
12                     triangular=False
13                     return triangular
14     return triangular
15
16 def matriz_traspuesta(matriz):
17     filas=len(matriz)
18     columnas=len(matriz[0])
19     matrizT=Array(columnas,Array(filas,0.0))
20     for i in range(0,columnas):
21         for j in range(0,filas):
22             matrizT[i][j]=matriz[j][i]
23     return matrizT
24
25 bandera=False
26 while(bandera==False):
27     filas=input_int("Ingrese las filas de la matriz: ")
28     columnas=input_int("Ingrese las columnas de la matriz: ")
29     if (filas<=0) or (columnas<=0):
30         print("Dimensiones inválidas")
31     else:
32         bandera=True
33
34 matriz=Array(filas,Array(columnas,0.0))
35 leer_matriz(matriz,filas,columnas)
36 mostrar_matriz(matriz,filas,columnas)
```

```
37
38  triangularInf=triangular_inferior(matriz,filas,columnas)
39 ▼if triangularInf==True:
40      print("La matriz es triangular inferior")
41      print("Su traspuesta es:")
42      matrizT=matriz_traspuesta(matriz)
43      mostrar_matriz(matrizT,columnas,filas)
44 ▼else:
45      print("La matriz NO es triangular inferior")
46
```

Link Repl.it: https://replit.com/@Paulonia/Martinez13866TP1-Ej8#main.py

Parte II: TAD Conjuntos

1. A partir del TAD **`Set(Conjunto)`** implementar las siguientes operaciones utilizando la estructura **Array:**

Create_Set(Array):
Descripción: Crea un TAD de tipo **Set** a partir de un arreglo recibido como parámetro.
Entrada: el Arreglo sobre el cual se quiere construir el TAD **Set**
Salida: Un **Array** que representa el **TAD** Set
Union(Array S,Array T):
Descripción: Aplica la operación UNIÓN sobre los conjuntos (Sets) S y T.
Precondición: La operación debe garantizar que no hay elementos duplicados en los arreglos. (Ver Nota más abajo)
Entrada: Dos arreglos que representan los **Sets** S y T
Salida: Un **Array** que representa un nuevo **TAD** Set
Intersection(Array S,Array T):
Descripción: Aplica la operación INTERSECCIÓN sobre los conjuntos S y T.
Precondición: La operación debe garantizar que no hay elementos duplicados en los arreglos. (Ver Nota más abajo)
Entrada: Dos arreglos que representan los **Sets** S y T
Salida: Un **Array** que representa un nuevo **TAD** Set
Difference(Array S, Array T):
Descripción: Aplica la operación DIFERENCIA sobre los conjuntos S y T.
Precondición: La operación debe garantizar que no hay elementos duplicados en los arreglos. (Ver Nota más abajo)
Entrada: Dos arreglos que representan los **Sets** S y T
Salida: Un **Array** que representa un nuevo **TAD** Set

2. Todas las operaciones del TAD conjunto deberán incluirse en un archivo (modulo) set.py

set.py ×

```
1  import algo1
2
3 ▼def check_duplicates(Array):
4      duplicado=False
5 ▼    for i in range(0,len(Array)):
6 ▼        for j in range(0,len(Array)):
7 ▼            if (Array[i]==Array[j]) and (i!=j):
8                  duplicado=True
9      return duplicado
10
```

```python
def Create_Set(Array):
  #busca los elementos repetidos del array y los elimina cambiando por None
  for i in range(0,len(Array)):
    for j in range(0,len(Array)):
      if (Array[i]==Array[j]) and (i!=j):
        Array[j]=None
  #cuenta la cantidad de elementos repetidos para sacar la dimension del nuevo array
  contadorDup=0
  for i in range(0,len(Array)):
    if Array[i]==None:
      contadorDup+=1
  n=len(Array)-contadorDup
  ArrayR=algo1.Array(n,0)
  #se llena el nuevo array ignorando los elementos None del array anterior
  cont=0
  for i in range(0,len(Array)):
    if(Array[i]!=None):
      ArrayR[cont]=Array[i]
      cont+=1
  return ArrayR

def Union(ArrayS,ArrayT):
  duplicados=check_duplicates(ArrayS)
  duplicados2=check_duplicates(ArrayT)
  if (duplicados==True) or (duplicados2==True):
    print("No es posible realizar la operación de unión")
    return
  ArrayR=algo1.Array(len(ArrayS)+len(ArrayT),0)
  for i in range(0,len(ArrayS)):
    ArrayR[i]=ArrayS[i]
  for i in range(0,len(ArrayT)):
    ArrayR[i+len(ArrayS)]=ArrayT[i]
  #Se devuelve el array resultado sin elementos repetidos
  ArrayR=Create_Set(ArrayR)
  return ArrayR

def Intersection(ArrayS,ArrayT):
  duplicados=check_duplicates(ArrayS)
  duplicados2=check_duplicates(ArrayT)
  if (duplicados==True) or (duplicados2==True):
    print("No es posible realizar la operación de intersección")
    return
  ArrayR=algo1.Array(len(ArrayS)+len(ArrayT),0)
  cont=0
  for i in range(0,len(ArrayS)):
    for j in range(0,len(ArrayT)):
      if ArrayS[i]==ArrayT[j]:
        ArrayR[cont]=ArrayS[i]
        cont+=1
  ArrayR=Create_Set(ArrayR)
  return ArrayR

```

```
62
63 def Difference(ArrayS,ArrayT):
64     duplicados=check_duplicates(ArrayS)
65     duplicados2=check_duplicates(ArrayT)
66     if (duplicados==True) or (duplicados2==True):
67         print("No es posible realizar la operación de diferencia")
68         return
69     ArrayR=algo1.Array(len(ArrayS)+len(ArrayT),0)
70     cont=0
71     for i in range(0,len(ArrayS)):
72         igual=False
73         for j in range(0,len(ArrayT)):
74             if ArrayS[i]==ArrayT[j]:
75                 igual=True
76         if igual==False:
77             ArrayR[cont]=ArrayS[i]
78             cont+=1
79     ArrayR=Create_Set(ArrayR)
80     return ArrayR
81
```

Link Repl.it: https://replit.com/@Paulonia/Martinez13866TP1-TAD#set.py