Universidad Nacional de Río Cuarto
Facultad de Cs. Exactas, Fco-Qcas y Naturales - Dpto de Computación
Asignatura: **INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN**
Código: 3300
Año: 2022

Práctico Nº 7

Tema: Tipo de Dato Estructurado Homogéneo: Arreglo Duración: 6 clases

1) A continuación se muestran distintos algoritmos y el procedimiento de carga de un arreglo. En este arreglo se almacenan 31 elementos, cada uno corresponde a la temperatura máxima diaria de una localidad determinada, durante el mes de Marzo (que tiene 31 días) del año 2020. Lea cada uno de los algoritmos y deduzca que hace cada uno, al final están las preguntas. Revise detenidamente cómo se definen las variables y cómo se construyen los ciclos.

```
Algoritmo QueHace?
Léxico
   Long=31
   TNums = arreglo[1..Long] de Z
   x ∈ TNums
   cant ∈ (0..Long) // cantidad de números cargados en el arreglo
   i ∈ Z
   mge ∈ Cadena
   my, n ∈ Z

Acción Cargar(resultado q ∈ TNums, c ∈ (0..Long))
Léxico local
   i ∈ Z
   mg ∈ Cadena
Inicio
   mg <-- "¿Cuantos números va a cargar?"
   Salida: mg
   Entrada: c
   para (i=1, i<=c, i <-- i +1) hacer
      Entrada: q[i]
   fpara
Facción

Acción QueHaceA(dato q ∈ TNums, c ∈ (0..Long))
Léxico local
  i ∈ Z
Inicio
  i<-- 1
  según
    i > c: mge <-- 'arreglo vacío'
           Salida: mge
    i <= c: mientras i <= c hacer
                  si q[i] >30 entonces
                   Salida: q[i]
                  fsi
                  i<-- i + 1
            fmientras
   fsegún
```

```
  Faccion

Acción QueHaceB(dato q ∈ TNums, c ∈ (0..Long), resultado may ∈ Z )
Léxico local
   i ∈ Z
Inicio
  i <-- 1
  según
    i > c: mge <-- 'arreglo vacío'
                Salida: mge
    i <= c: may <-- q[1]
                mientras i <= c hacer
                    si q[i] < may entonces
                      may <-- q[i]
                    fsi
                   i<-- i + 1
                fmientras
  fsegún
Faccion

Acción QueHaceC(dato q ∈ TNums, c ∈ (0..Long))
Léxico local
  i ∈ Z
Inicio
   i <-- 1
  según
    i > c:    mge <-- 'arreglo vacío'
              Salida: mge
    i <= c: mientras i <= c hacer
                 si q[i] < 0 entonces
                   Salída: q[i]
                 fsi
                i <-- i + 1
              fmientras
  fsegún
 Facción

Acción QueHaceD (dato q ∈ TNums, c ∈ (0..Long), resultado num ∈ Z )
Léxico local
   i ∈ Z
Inicio
  i <-- 1
  según
    i > c:    mge <-- 'arreglo vacío'
             Salida: mge
    i <= c: num <-- 0
             mientras i <= c hacer
                si q[i] > 21 entonces
                  num <-- num + 1
                fsi
                i <-- i + 1
              fmientras
   fsegún
```

Facción

```
Inicio // inicio del algoritmo principal
   Cargar (x, cant)
   QueHaceA(x, cant)
   QueHaceB(x, cant, my)
   QueHaceC(x, cant)
   QueHaceD(x, cant, n)
Fin // fin del algoritmo
```

a) ¿En la acción QueHaceA que muestra por la salida ?
b) ¿En la acción QueHaceB que almacena en la variable my?
c) ¿En la acción QueHaceC que muestra por la salida?
d) ¿En la acción QueHaceD que almacena en la variable n?

Algunas de las posibles respuestas a estas preguntas son: muestra los positivos, muestra los números menores a 30, almacena el número más grande, calcula cuántos números son mayores a 30, cuántos números son negativos, cuantos números son iguales a 21, etcétera (estas respuestas no son necesariamente las correctas, sino un ejemplo de lo que pedimos como respuesta, una idea concreta y breve de lo que calcula cada acción.

Ej. 2) Un espía, con el objeto de descifrar una clave secreta necesita contar la cantidad de vocales que hay en una frase, la cuál se encuentra almacenada en un arreglo de caracteres de 100 elementos.. Diseña un algoritmo que ayude al espía a determinar cuántas vocales hay en la frase.
Utilice:
N= 100
Tclave = arreglo [1..N] de Caracter
clave ∈ Tclave
cant ∈ (0..N)
// para practicar, definimos el arreglo y la cantidad por separado. A partir del próximo ejercicio comenzaremos // a usar un registro (TData, como lo llamamos en la teoría) que contiene el arreglo y la cantidad.

Ej. 3) En un Bingo se registran los números que van saliendo hasta que haya un ganador, o se saquen 30 números. Luego se debe informar los números que salieron. Diseñar un algoritmo que permita ir almacenando los números que salen en una estructura de datos adecuada y si hay un ganador que se detenga la carga de números. Posteriormente todos los números que fueron almacenados deben ser mostrados, la condición de ganador se evalúa mediante la pregunta ¿Hay ganador?, N no hay ganador, S hay ganador.

Ej. 4) Se necesita diseñar un algoritmo que permita almacenar en una estructura, a lo sumo 10 vocales. Por cada dato ingresado, si no es vocal se descarta, sólo se almacenarán las vocales (hasta 10 a lo sumo). Pero la cantidad de caracteres obtenidos no debe superar los 20. Luego informar la cantidad de vocales almacenadas.
Nota: se recomienda modularizar (utilizar acciones y/o funciones según sea lo más conveniente).

Ej. 5) Un profesor necesita registrar hasta 200 notas (entre 1 y 10) de un examen y luego calcular el promedio. Complete la acción promedioNotas y el cuerpo principal del siguiente algoritmo:

```
Algoritmo Notas
Léxico
  Max = 200
  TElem = (1..10)
  TNumeros = arreglo[1..Max] de TElem
  TData = <a ∈ TNumeros, cant ∈ (0..Max)>
  misNotas ∈ TData
```

```
  promedio ∈ R

  Acción cargarNotas(resultado notas ∈ TData)
  Léxico local
    i ∈ Z
  Inicio
  // cantidad de notas a cargar
    Entrada: notas.cant
    para (i <-- 1, i<=notas.cant, i <-- i+1) hacer
      // obtener cada nota
      Entrada: notas.a[i]
    fpara
  Facción

  Acción promedioNotas(……)
  Léxico local
    ...
  Inicio
    ...
  Facción

Inicio
  cargarNotas(misNotas)
  ...
Fin
```

Ej. 6) En un sistema contable se almacenan las ventas totales mensuales de sus vendedores en un arreglo. Se solicita una función que reciba el arreglo y calcule la suma de las ventas que contiene.
Utilice:
Max = 60
Telem = R
Tventas = arreglo [1..Max] de Telem
TData = <v ε Tventas, cant ε (0..Max)>

Ej. 7) Para un software de una lotería nos solicitan que proveamos un conjunto de hasta 30 números enteros comprendidos entre 0 y 999, generados aleatoriamente. Implementa una solución modular que almacene los números en un arreglo. (Usar un TData adecuado al problema planteado)

Nota: para resolver este ejercicio se asume la existencia de una cierta función llamada random(n). El parámetro **n** debe ser un valor entero. Si se pasa un valor n como parámetro, la función devolverá un número entero pseudoaleatorio en el rango de 0 a n-1. Así por ejemplo random (10) dará por resultado un número entero comprendido entre 0 y 9. Para obtener números pseudoaleatorios en un intervalo determinado, por ejemplo en el rango de 3 a 8 inclusive, llame a la función de la siguiente forma: random (6) + 3

Ej. 8) Un profesor guarda en una lista en papel las calificaciones de **Nmax** alumnos. Con este listado calcula a mano:

a) El promedio general del grupo

b) Número de alumnos aprobados y número de alumnos reprobados. Se considera aprobado a una nota igual o mayor a 5

c) Porcentaje de alumnos aprobados y reprobados

¿Puedes ayudarlo a realizar estos cálculos utilizando un arreglo que contenga todas las calificaciones?

Ej. 9) Desarrollar una función llamada capicúa: que reciba un arreglo y la cantidad de elementos que tiene cargados (Utilice un TData adecuado). La función debe devolver verdadero si los elementos equidistantes de los extremos del arreglo son iguales (es decir, se lee igual de izquierda a derecha y de derecha a izquierda). Ejemplo de casos en que debería devolver verdadero:

N	E	U	Q	U	E	N

A	C	U	R	R	U	C	A

Ej. 10) Desarrollar una función llamada espejo: que reciba un arreglo y la cantidad de elementos que tiene cargados (Utilice un TData adecuado) , y devuelva un arreglo pero con los elementos en posiciones opuestas a la original tomando como referencia la primera posición.

Ejemplo: si el arreglo fuese

P	E	R	R	O

La aplicación de la función daría como resultado:

O	R	R	E	P

Ej. 11) Desarrollar una acción que dado un arreglo A ya cargado, que contiene **t** números enteros positivos menores o iguales a 10, copie en otro arreglo B todos los valores del arreglo de A, ignorando los valores duplicados que se encuentran en A. Mostrar la cantidad de elementos copiados en el arreglo B y los elementos del mismo.

Ejemplo:

A

1	2	2	5	4	3	2	4	1

B

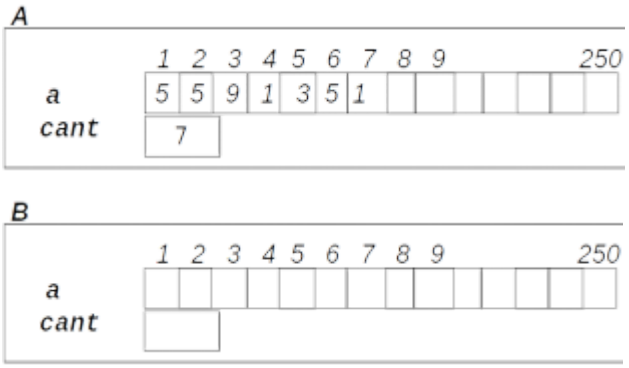
1	2	5	4	3

Se deberá informar: Cantidad total de elementos: 5 – Elementos: 1 2 5 4 3

Nota: defina los dos arreglos de la misma dimensión.

CLASE 3 de 5 Arreglo Unidimensional

Ej. 12) Dados estos dos registros TData:



Y el siguiente algoritmo, responda cuál es la secuencia de números que se almacena en **b** y cuántos son.

```
Algoritmo NoRepetidos
Léxico
  Max = 250
  TElem = (1..10)
  TNumeros = arreglo[1..Max] de TElem
  TData = <a ∈ TNumeros, cant ∈ (0..Max)>
  a, b ∈ TData
  Acción Cargar(resultado z ∈ TData)
  Léxico local
    i, t ∈ Z
   Inicio
    // cantidad de números a cargar
    repetir
      Entrada: t
    hasta que (t>=0) y (t <= Max)
    para (i <-- 1, i<=t, i<--i+1) hacer
      // obtener cada número
       Entrada: z.a[i]
    fpara
    z.cant <-- t
  Facción
  Acción CargaEnOtro(dato q ∈ TElem, dato-resultado z ∈ TData)
  Léxico local
    i ∈ Z
    esta ∈ Lógico
  Inicio
    i <-- 1
    esta <-- Falso
    mientras (i <= z.cant) y (no esta) hacer
     si q = z.a[i] entonces
       esta <-- Verdadero
     fsi
     i <-- i + 1
   fmientras
   si (no esta) entonces
     z.a[i] <-- q
     z.cant <-- i
```

```
  fsi
 Facción
 Acción Mostrar (dato z ∈ TData)
 Léxico local
  i ∈ Z
 Inicio
  i <-- 1
  para (i<--1, i <= z.cant, i<-- i + 1) hacer
    Salida: z.a[i]
  fpara
  Salida: z.cant
 Faccion

Inicio // inicio algoritmo
  Cargar(a)
  b.cant <-- 0
  para (i<--1, i <= aA.cant, i<-- i + 1) hacer
    CargarEnOtro(a.a[i], b)
  fpara
  Mostrar(b)
Fin // fin algoritmo
```

Ej. 13) Dado un arreglo A ya cargado con t números enteros positivos menores o iguales a 10, y ordenado de menor a mayor, desarrolle una acción que permita insertar un elemento, en el lugar que le corresponde para mantener el orden creciente.

Ej. 14) Desarrollar un algoritmo que permite cargar un arreglo de caracteres, y luego devuelva el arreglo eliminando los espacios en blanco, para esto debe desplazar los caracteres situados a la derecha del espacio en blanco, un lugar hacia la izquierda. Ejemplo sea A un arreglo de caracteres A=(c,a,s,a, ,a,m,a,r,i,l,l,a, , c,a,s,a, , a,z,u,l), debe quedar una vez eliminado los espacios en blanco: A=(c,a,s,a,a,m,a,r,i,l,l,a,c,a,s,a,a,z,u,l). Para resolver crear las acciones: CargarFrase(), EliminaBlancos() y MostrarFrase(), cada una adecuadamente parametrizada. Puede rehusar acciones creadas en ejercicios anteriores.

Ej. 15) Para administrar una lista de personas se requiere hacer diversas acciones que permitan ese trabajo. Para almacenar los nombres se utilizará un registro que contiene dos campos, uno con un arreglo donde se guardan los nombres y otro con la cantidad actual de nombres almacenados en el arreglo (Utilice un registro con un campo arreglo y otro campo con la cantidad de datos que se han cargado al arreglo). La cantidad máxima que se podrá almacenar será de 1000 nombres. Para poder administrar la lista se deben desarrollar las siguientes acciones y/o funciones:

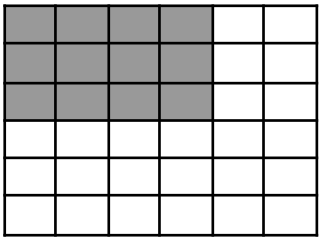
a) Una función llamada Vacia que reciba como parámetro el registro (con el arreglo de nombres y la cantidad de nombres cargados), y devuelva verdadero si la lista está vacía
   y sino debe devolver falso.
b) Una función llamada Llena que reciba como parámetro el registro (con el arreglo de nombres y la cantidad de nombres cargados), y devuelva Verdadero si el arreglo está completamente lleno (es decir si ya tiene 1000 nombres cargados) y sino debe devolver Falso.
c) Una acción que permita insertar un nombre en el arreglo. Para ello se pasará como parámetros el registro (con el arreglo de nombres y la cantidad de nombres cargados), y el nuevo nombre a insertar. El nuevo nombre se inserta siempre al final de la lista. Después de ejecutada la acción, la cantidad debe quedar incrementada en una unidad.
d) Una acción que permita suprimir al primer nombre de la lista, no importa cuál es. Simplemente suprime el primero cada vez que la acción es ejecutada. El parámetro que debe recibir es el registro (con el arreglo de nombres y la cantidad de nombres cargados). Después de ejecutada la acción, la cantidad debe quedar decrementada en una unidad.

e) Una acción que permita mostrar todos los nombres de la lista. Los parámetros que debe recibir son: el arreglo y la cantidad de nombres que tiene cargados.

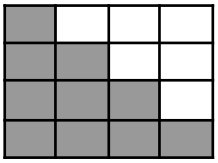
CLASE 4 de 5 Arreglo Bidimensional

Ej. 16) Dado un arreglo bidimensional **K** cargado con **n*n** números enteros (n>2):

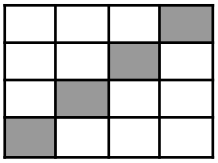
a- Calcular la suma de los datos de la zona sombreada (mitad de filas por mitad de columnas mas uno).



b- Calcular la suma de la zona sombreada e informarla:



c- Imprimir los datos de la zona sombreada en el orden indicado por la flecha:



Ej.17) Corazón = Cáscara: Dada una Matriz de [n x m] de enteros, con n >= 3 y m >=3, determinar si la sumatoria de las celdas límites (cáscara) es igual que la sumatoria las celdas del interior (todas menos las límites). Por ejemplo, para la siguiente matriz de (4x4)

	1	2	3	4
1	1	2	1	1
2	1	2	3	-1
3	-1	4	2	1
4	3	1	1	1

La respuesta es verdadero, ya que las celdas límites suman 11 al igual que las del interior.

Ej. 18) Desarrolle un algoritmo que sume dos matrices y guarde el resultado en una tercera. Las matrices para que puedan sumarse deben tener las mismas dimensiones. Ejemplo: m1= ((12,13,14,10), (15,16,17,10), (18,19,20,10)) y m2 = ((2,1,1,1), (2,1,1,1), (2,1,1,1))

Resultado: m3= (14,14,15,11), (17,17,18,11), (20,20,21,11) los elementos de m3[i,j]= m1[i,j] + m2[i,j]

Ej. 19) Desarrolle un algoritmo que multiplique una matriz por un vector (arreglo unidimensional). Para que se pueda realizar este producto el vector debe tener tantos elementos como columnas tiene la matriz: Así el producto entre A de nxm y B de dimensión m da como resultado un vector C de n elementos. Ejemplo: A=

((12,13,14,10), (15,16,17,10), (18,19,20,10)) y B = (2,1,1,1), Resultado: C= (12x2+13x1+14x1+10x1,15x2+16x1+17x1+10x1,18x2+19x1+20x1+10x1) = (61,73,85).

Nota: el producto de dos matrices solo es posible si la primer matriz tiene tantas columnas como filas tiene la segunda. Así el producto de dos matrices X de m x n, por una matriz Y de n x p, da como resultado una matriz Z de m x p. Cada elemento de Z será: Zjk= Xj1 Y1k + Xj2 Y2k+...+Xjn Ynk

Ej. 20) Suma de vecinos**:** Dada una matriz de enteros, determinar si existe una posición (informando su posición) en la misma tal que el valor en dicha posición es igual a la suma de sus vecinos rectilíneos (superior, inferior, izquierdo y derecho). Cabe aclarar que las posiciones que se encuentran en algunos de los límites (o esquinas) de dicha matriz, sólo deben considerarse las posiciones existentes. En caso de existir más de una, simplemente aclare cuál retorna. Por ejemplo, para la siguiente matriz de (5x4)

	1	2	3	4	5
1	3	1	8	0	0
2	1	4	1	0	3
3	0	1	0	1	6
4	0	0	0	0	2

Posibles resultados son
[2,2]=4, ya que sus vecinos rectilíneos ([1,2]=1 + [2,1]=1 + [3,2]=1 + [2,3]=1) = 4
[3,5]=6, ya que los 3 (está en el límite) vecinos rectilíneos ([4,3]=1 + [5,2]=3 + [5,4]=2) = 6

Ej.21) Sopa de letras: Dada una Matriz de [n x m] caracteres y un Arreglo de l caracteres, con l <=n y l<=m, determinar si la palabra que forma el arreglo está presente en la matriz al menos una vez, de forma vertical u horizontal (cabe destacar que a diferencia de la sopa de letra original, no debe considerar que la palabra pueda estar en forma diagonal o al revés). Se debe informar entre que posiciones está la palabra, si es que está. Por ejemplo, para la siguiente matriz de (5x4) y arreglo:

	1	2	3	4	5
1	i	a	t	q	u
2	g	q	u	e	r
3	t	u	h	i	a
4	v	e	u	e	i

q	u	e
1	2	3

Un posible resultado es desde [2,2] hasta [2,4]

Ej. 22) Dada un arreglo de 365 elementos, llamado lluvia10 [1..12, 1..31], cuyos valores corresponden a las precipitaciones diarias en milímetros ocurridas a lo largo del año 2010 en una determinada ciudad; desarrollar un algoritmo que calcule:

a) Cuál es la lluvia promedio por día a lo largo del año 2010 (resultado: un solo valor, se puede resolver con una función)
b) Cuál es la lluvia promedio mensual en cada uno de los meses del año 2010. (resultado: un arreglo de

12 elementos)

c) Cuál es la máxima precipitación y en que día y mes ocurrió.
d) Cuál es la mínima precipitación y en que día y mes ocurrió.
e) Dada una cantidad de días, determinar cuál es en el año el período de dicha cantidad de días con menos precipitaciones y cuál es el período con más precipitaciones.
f) Dado un promedio p, determinar cuál es el período del año más largo que lo contiene con una diferencia menor a 10 mm.

Plan mínimo de ejercicios a realizar en clases:

1era Clase: 1) 2) 3)
2da Clase: 5) 6) 7) pasar a C el ej. 5)
3ra Clase: 9) 10) 12) pasar a C ej. 7)
4ta Clase 13) 15.a) 15.c) y 15.e) pasar a C ej 13)
5ta Clase 16) 17) 18) pasar a C ej 17)
6ta Clase 20) 21) 22.a), 22.b) pasar a C ej. 20