UADE
UNA GRAN UNIVERSIDAD

FUNDACIÓN UADE

FACULTAD DE INGENIERIA Y CIENCIAS EXACTAS
Departamento de Tecnología Informática

INTRODUCCION A LA ALGORITMIA

Guía de Trabajos Prácticos

V01.00.00

Objetivos: Que el alumno pueda resolver problemas de forma simple y práctica.

- Plantear estrategias de resolución de problemas creando algoritmos
- Entender y saber construir un programa estructurado
- Diferenciar los distintos tipos de estructuras de control
- Identificar módulos y definir funciones
- Utilizar estructuras de datos básicas

Bibliografía sugerida:

Básica

- Joyanes Aguilar, Luis: Fundamentos de Programación. Editorial Mc.Graw-Hill. ISBN 978-84-481-6111-8. EAN 9788448161118.

Complementaria

- Pilgrin, Mark: Inmersión en Python 3 [en línea] Traducido por José Miguel González Aguilera. Disponible gratuitamente (bajo licencia Creative Commons 3.0). [consultas: 9/07/20] **http://www.jmgaguilera.com/libro/python/traducci%C3%B3n/latex/2016/08/19/inmersion-python.html**
- Marzal Varó, Andrés - Gracia Luengo, Isabel, García Sevilla, Pedro: Introducción a la programación con Python 3 [En línea] Disponible gratuitamente bajo licencia Creative Commons. [consultas: 9/07/20] **https://openlibra.com/es/book/introduccion-a-la-programacion-con-python-3**
- The Python Language Reference. [consultas: 9/07/20]. Español: **https://docs.python.org/3/reference/**
- Documentos varios en Python.org.ar [consultas: 9/07/20]. Inglés / Español: **http://www.python.org.ar/aprendiendo-python/**

Autor:

Trabajo Práctico 1: Algoritmos elementales

Objetivo:

- Resolver problemas creando algoritmos.
- Definir la estructura básica de un algoritmo: Entrada-Proceso-Salida.

Ejercicio 1: Escribir la secuencia de acciones necesarias para lograr realizar el objetivo planteado. Identificar el estado inicial y el estado final para cada situación.

a. Cruzar una calle en una esquina con semáforos.
b. Lavarse las manos.
c. Preparar un mate.
d. Destapar una botella de cerveza.
e. Pintar una pared.
f. Resolver la siguiente cuenta: 4 + 2 * 3
g. Separar el cuatro de copas de un conjunto de naipes.
h. Buscar la menor carta de un mazo de naipes, que puede no estar completo.
i. Ordenar un conjunto de cartas

Trabajo Práctico 2: Entrada y Salida de Datos – Estructura Secuencial

Objetivos:

- Conocer los distintos tipos de datos y la gestión de los datos a través de variables.
- Identificar los distintos operadores aritméticos y el orden de precedencia y cálculos básicos.
- Programas estructura secuencial.

Ejercicio 1: Las variables guardan datos de diferente tipo y permiten cambiar su valor. Dadas dos variables numéricas A y B, que el usuario debe ingresar por teclado, se pide:
Diseñar el algoritmo para resolver los siguientes problemas y escriba el pseudocodigo o diagrama de flujo.

a. Mostrar el mensaje "Hola Mundo".
b. Ingresar el nombre del usuario del programa y saludarlo. Ejemplo: Si el usuario se llama Juan, se debe mostrar el mensaje "Hola Juan".
c. Ingresar dos números y mostrar la suma y la diferencia.
d. Ingresar tres números y mostrar el promedio.
e. Ingresar el monto de una factura y calcular el IVA (21%).
f. Duplicar el valor que poseen y mostrarlo por pantalla.
g. Intercambiar los valores de ambas variables y mostrar cuánto valen al final las dos variables.

Ejercicio 2: Calcular el promedio de las notas que obtiene un alumno al rendir los dos parciales.

Ejercicio 3: Tres personas invierten dinero para fundar una empresa (no necesariamente en partes iguales). Calcular qué porcentaje invirtió cada una.

Ejercicio 4: Leer una medida en metros e imprimir esta medida expresada en centímetros, pulgadas, pies y yardas. Los factores de conversión son:

- 1 pie = 12 pulgadas
- 1 yarda = 3 pies
- 1 pulgada = 2,54 cm.
- 1 metro = 100 cm.

Ejercicio 5: Una inmobiliaria paga a sus vendedores un salario base, más una comisión fija por cada venta realizada, más el 5% del valor de esas ventas. Realizar un programa que imprima el número del vendedor y el salario que le corresponde en un determinado mes. Se leen por teclado el número del vendedor, la cantidad de ventas que realizó y el valor total de las mismas.

Ejercicio 6: Una persona quiere invertir su capital en un banco y quiere saber cuánto dinero gana en un mes si el banco paga 2% mensual. ¿Cuánto ganará en seis meses si deja su dinero invertido?

Ejercicio 7: Leer un período en segundos e imprimirlo expresado en días, horas, minutos y segundos. Por ejemplo, 200000 segundos equivalen a 2 días, 7 horas, 33 minutos y 20 segundos.

Ejercicio 8: Un banco necesita para sus cajeros de la sucursal un programa que lea una cantidad de dinero que desea retirar el cliente e imprima a cuántos billetes equivale, considerando que existen billetes de $1000, $500, $200, $100, $50, $20 y el resto en monedas. Desarrollar dicho programa de tal forma que se minimice la cantidad de billetes entregados por el cajero

Ejercicio 9: Escribir un programa para convertir un número binario de 4 cifras en un número decimal. Se ingresa como un solo número entero de cuatro dígitos.

Ayuda: Dado un número binario, se toma de a un dígito a la vez, de menor a mayor comenzando por el índice 0 y se multiplica por (2x) donde x corresponde a la posición del dígito. Estos resultados se suman.
Ejemplo: $1\ 0\ 1\ 1_2 = 1 * 2_0 + 1 * 2_1 + 0 * 2_2 + 1 * 2_3 = 11_{10}$
Índices: 3 2 1 0

Trabajo Práctico 3: Estructura condicional

Objetivos:

- Crear algoritmos combinando estructura secuencial y condicional.
- Ser capaz de escribir algoritmos creados en lenguaje Python.
- Comenzar a utilizar variables que guardan valores booleanos (True/False).

Ejercicio 1: Escribir un programa que pida al usuario dos números y muestre por pantalla su división. Si el divisor es cero el programa debe mostrar un mensaje de error indicando que no se puede realizar la operación.

Ejercicio 2: Leer un número entero e imprimir un mensaje indicando si es par, impar o es cero.

Ejercicio 3: Ingresar la edad de una persona e indicar si es mayor de edad (mayor o igual a 18 años).

Ejercicio 4: Ingresar las notas de los dos parciales de un alumno e indicar si promocionó, aprobó o debe recuperar.

- Se promociona cuando las notas de ambos parciales son mayores o iguales a 7.
- Se aprueba cuando las notas de ambos parciales son mayores o iguales a 4.
- Se debe recuperar cuando al menos una de las dos notas es menor a 4.

Ejercicio 5: Escribir un programa para una empresa que tiene salas de juegos para todas las edades y quiere calcular de forma automática el precio que debe cobrar a sus clientes por entrar. El programa debe preguntar al usuario la edad del cliente y mostrar el precio de la entrada. Si el cliente es menor de 4 años puede entrar gratis, si tiene entre 4 y 18 años debe pagar 500$ y si es mayor de 18 años, 1000$

Ejercicio 6: Leer un número correspondiente a un año e imprimir un mensaje indicando si es bisiesto o no. Se recuerda que un año es bisiesto cuando es divisible por 4. Sin embargo, aquellos años que sean divisibles por 4 y también por 100 no son bisiestos, a menos que también sean divisibles por 400. Por ejemplo, 1900 no fue bisiesto, pero sí el 2000.

Ejercicio 7: Leer tres números correspondientes al día, mes y año de una fecha e imprimir un mensaje indicando si la fecha es válida o no.

Ejercicio 8: Diseñar un programa que calcule y muestre el sueldo neto de un empleado en base a su sueldo básico y su antigüedad en años. Si es soltero se le incrementa el sueldo en 5% del salario bruto por cada año de antigüedad, mientras que si es casado se le incrementa el sueldo en 7% del bruto por cada año de antigüedad. También se le realizan los siguientes descuentos: Jubilación: 11%, Obra Social: 3%, Sindicato: 3%.
Como datos de entrada se ingresa por teclado el sueldo básico, antigüedad y estado civil ('s' o 'c'). Se debe informar: (reemplazar los 9 por los valores que correspondan)

Estado Civil: Soltero/Casado

Sueldo básico	Antigüedad	Descuentos	Importe
$ 999.99	99 años		+ 999.99
		Jubilación	- 999,99
		Obra Social	- 999,99
		Sindicato	- 999,99

	Sueldo Neto		999,99

Trabajo Práctico 4: Estructura iterativa

Objetivos:

- Crear algoritmos combinando estructura secuencial, condicional e iterativa.
- Ser capaz de escribir algoritmos creados en lenguaje Python.

Ejercicio 1: Escribir un algoritmo que muestre los primeros 25 números naturales.

Ejercicio 2: Calcular e imprimir la suma de los números comprendidos entre 42 y 176.

Ejercicio 3: Mostrar las tablas de multiplicar (entre 1 y 10) del número 4. ¿Cómo cambiaría el algoritmo para que el usuario pueda decidir la tabla de multiplicar a mostrar?

Ejercicio 4: Se desea ingresar la nota de 5 alumnos y mostrar su promedio.

Ejercicio 5: Se ingresan N números enteros. Determinar el promedio de los números pares.

Ejercicio 6: Leer N números enteros e imprimir el mayor y en qué orden fue ingresado.

Ejercicio 7: Se ingresan N letras. Determinar la cantidad de vocales ingresadas.

Ejercicio 8: Escribir un algoritmo que lea números enteros hasta que se ingrese un 0, y muestre el máximo, el mínimo (sin considerar el 0) y la media (promedio) de todos ellos.

Ejercicio 9: Ingresar números, hasta que la suma de los números pares supere 100. Mostrar Cuántos números en total se ingresaron.

Ejercicio 10: Un negocio desea saber el importe total recaudado al fin del día, desea contar con un programa que pueda ingresar el importe de cada venta realizada. Para indicar que finalizó el día se ingresa -1. ¿Cuál fue el monto total vendido y cuántas ventas se realizaron? El importe de cada venta realizada debe ser un valor positivo.

Ejercicio 11: Leer dos números A y B (enteros positivos). Calcular el producto A * B por sumas sucesivas e imprimir el resultado. Ejemplo: 4*3 = 4 + 4 + 4 (4 sumado 3 veces).

Ejercicio 12: Leer dos números naturales A y B. Calcular A^B mediante productos sucesivos y mostrar el resultado. Ejemplo: 4^3 = 4 * 4 * 4 (4 multiplicado 3 veces).

Ejercicio 13: Hacer un programa que permita ingresar un número natural n. Informar si n es un término de la serie de Fibonacci. En este caso indicar en que posición de la serie se encuentra.
Ejemplo: 1, 1, 2, 3, 5, 8 ... . Si el número ingresado es 4, el número no pertenece a la serie. Si el número es 5, este pertenece a la serie y su orden es 5.
(Ley de formación de los términos en dicha serie: los dos primeros términos son 1 y cada uno de los demás es igual a la suma de los dos anteriores).

Ejercicio 14: Hacer un programa que permita ingresar los años de fabricación de las unidades de una empresa de transporte urbano de pasajeros. Al finalizar dicho ingreso (año cero), el programa debe informar el porcentaje de vehículos que tienen menos de 5 años de antigüedad.

Trabajo Práctico 5: Sub-Algoritmos: Funciones

Objetivo:

- Diseñar algoritmos que permitan modularizar e independizar sub-algoritmos reutilizarlos.

Diseñe el sub-algoritmo planteado en cada enunciado y desarrolle la función en lenguaje Python. Debe crear un programa principal para ejecutar la función solicitada.

Crear Funciones:

Ejercicio 1: Dado un número entero, calcular su factorial. Ejemplo: fact(4) = 4*3*2*1 = 24.

Ejercicio 2: Diseñar una función para mostrar un título filas de asteriscos, la longitud de la fila de asteriscos y el texto del título se recibe como parámetro. Ejemplo: titulo("Ejercicio 3", 15)

```
***************
Ejercicio 3
***************
```

Ejercicio 3: Escribir la función obtener_mes_en_texto, que devuelva una cadena que representa un mes expresado en letras según un número entero entre 1 y 12 recibido como parámetro. Si el parámetro no es válido, devolver una cadena vacía. Ejemplo: Se invoca obtener_mes_en_texto(4) → devuelve "Abril".

Ejercicio 4: Desarrolla una función que retorne la cantidad de divisores de un número entero positivo.

Ejercicio 5: Desarrolla una nueva función que verifique si un número es un número perfecto, retorna True o False según corresponda. Un número perfecto es aquel cuya suma de divisores propios (excluyendo el propio número) es igual al número en sí.

Ejercicio 6: Desarrollar una función que reciba un y retorne True/False indicando si es bisiesto o no. Se recuerda que un año es bisiesto cuando es divisible por 4. Sin embargo, aquellos años que sean divisibles por 4 y también por 100 no son bisiestos, a menos que también sean divisibles por 400. Por ejemplo, 1900 no fue bisiesto, pero sí el 2000.

Ejercicio 7: Desarrollar una función que reciba tres números enteros positivos y verifique si corresponden a una fecha gregoriana válida (día, mes, año). Devolver True o False según la fecha sea correcta o no. Realizar también un programa para verificar el comportamiento de la función.

Ejercicio 8: Desarrollar una función para ingresar del teclado un número entero y validar que sea positivo, debe retornar el valor.

Uso de funciones existentes:

Ejercicio 9: Desarrollar un programa generar N números al azar de un dígito, positivos. N se ingresa por teclado. Para cada valor al azar, mostrar el valor creado y su factorial (utilizar la función creada en el ejercicio1).

Ejercicio 10: Desarrollar un programa que genere un número entero al azar de cuatro cifras y proponerle al usuario que lo descubra, ingresando valores repetidamente hasta hallarlo. En cada intento el programa mostrará mensajes indicando si el número ingresado es mayor o menor que el valor secreto. Permitir que el usuario abandone al ingresar -1. Informar la cantidad de intentos realizada al terminar el juego. Utilizar la función creada en el ejercicio 8 para resolver el problema.

Trabajo Práctico 6: Estructuras de datos Listas

Desarrollar los siguientes programas, no es necesario crear funciones, el objetivo es trabajar con listas:

Ejercicio 1: Crear una lista con valores al azar entre 1 y 20. Mostrar los elementos que superan el promedio de los valores ingresados.

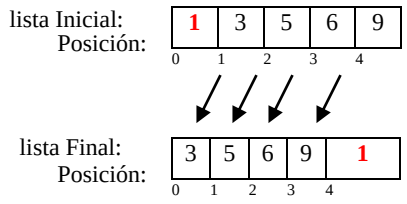
Ejercicio 2: Crear una lista con valores positivos desde el teclado hasta ingresar -1. Luego recorrer la lista para duplicar el valor de aquellos elementos que sean menores al promedio de los valores ingresados.

Ejercicio 3: Crear una lista con N valores al azar en un rango desde y hasta ingresado también desde el teclado. Imprimir el valor mínimo y el lugar que ocupa. Tener en cuenta que el mínimo puede estar repetido, en cuyo caso deberán mostrarse todas las posiciones que ocupe. No utilizar la función min, index o find de Python, debe resolver el algoritmo, puede crear sus propias funciones o resolverlo sin funciones.

Ejercicio 4: Rellenar una lista con números enteros entre 0 y 20 obtenidos al azar, La carga de datos termina cuando se obtenga un 0 como número al azar, el que no deberá cargarse en la lista. Mostrar la lista por pantalla y luego realizar resolviendo el algoritmo, sin utilizar las funciones de Python que resuelven el problema, se espera que sea capaz de crear sus propios algoritmos:

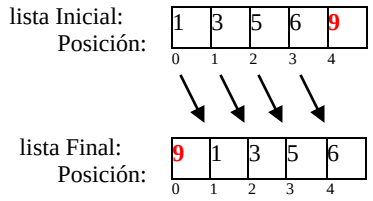
a. Cambiar el elemento que se encuentra en la posición cero de la lista a la última posición y desplazar todos los demás elementos una posición hacia delante sin perder ningún elemento. Se necesita utilizar variable/s auxiliares. No utilizar una lista auxiliar.

Ejemplo:



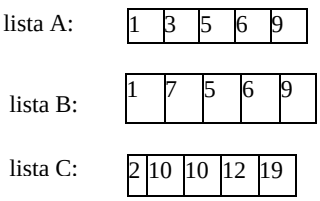
b. Rotar una posición de la lista a derecha. Igual que el punto b pero hacia la derecha.

Ejemplo:

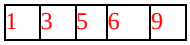


Ejercicio 5: Cargar dos listas A y B del mismo tamaño, N elementos al azar entre 1 y 99, realizar :

a. Suma A+B en una tercer lista, ejemplo.

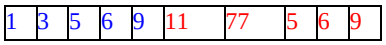


b. Concatenación de A con B, en una lista C (No usar facilidades de Python, resolver el algoritmo)



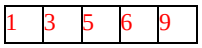
lista A

lista B



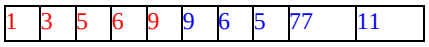
lista C

c. Concatenación de A con B invertido, en una lista C



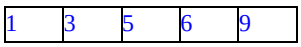
lista A

lista B



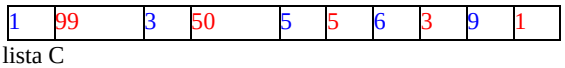
lista C

d. Crear otra lista C que contenga los elementos de A y B intercalados, comenzando por A.



lista A

lista B



lista C

Trabajo Práctico 7: Ordenamiento y Búsquedas

Ejercicio 1: Escribir una función para crear una lista con N valores ingresados desde el teclado, hasta ingresar -1. La lista no debe aceptar valores repetidos, rechazarlos si esto sucede. No utilizar el operador IN, crear su propia función existe para detectar si un elemento ya se encuentra en una lista.

Ejercicio 2: Escribir una función para crear una lista con N valores al azar en un rango desde y hasta que se recibe por parametro. Retorna la lista creada

En los siguientes ejercicios utilice una de las funciones anteriores para ingresar datos en una lista y luego resuelva:

Ejercicio 3: Escribir una función para devolver la posición que ocupa un valor pasado como parámetro, utilizando **búsqueda secuencial** en una lista desordenada. La función debe devolver -1 si el elemento no se encuentra en la lista.

Ejercicio 4: Crear tres listas y ordenarlas en forma ascendente utilizando un método para cada lista: **métodos de selección, inserción y burbujeo**. ¿Qué cambia para ordenar en forma descendente?

Ejercicio 5: Crear una lista sin elementos repetidos, ordenarla por un método a elección y mostrar por pantalla. Ingresar por teclado números positivos hasta ingresar -1, indicar en qué posición se encuentra utilizando **búsqueda binaria** si el valor se no encuentra mostrar un mensaje aclaratorio. Al finalizar las búsquedas informar cuántas búsquedas resultaron satisfactorias.

Ejercicio 6: Dada una lista ordenada de números llamada A y un nuevo número N, desarrollar un programa que agregue el elemento N dentro de la lista A, respetando el ordenamiento existente. El programa deberá detectar automáticamente si el ordenamiento es ascendente o descendente antes de realizar la inserción. No utilizar facilidades de Python para resolver, crear el algoritmo para insertar un elemento en una lista.

Trabajo Práctico 8: Ejercicios Integrales

Ejercicio 1: Leer los números de legajo de los alumnos de un curso y su nota de examen final. Cargar el número de legajo en la lista LEGAJOS y la nota en la lista NOTA_FINAL. El fin de la carga se determina ingresando un -1 como legajo. Se debe validar que la nota ingresada sea entre 1 y 10. Terminada la carga de datos, recorrer las listas e informar:

- Cantidad de alumnos que aprobaron con nota mayor o igual a 4
- Cantidad de alumnos que desaprobaron el examen. Nota menor a 4
- Promedio de nota y los legajos que superan el promedio

Luego se solicita, ordenar las listas según el número de legajo, de manera ascendente.

Ejercicio 2: Una escuela necesita conocer cuántos alumnos cumplen años en cada mes del año, con el propósito de ofrecerles un agasajo especial en su día. Desarrollar un programa que lea el número de legajo y fecha de nacimiento (día, mes y año) de cada uno de los alumnos que concurren a dicha escuela. La carga finaliza con un número de legajo igual a -1. Emitir un informe donde aparezca (mes por mes) cuántos alumnos cumplen años a lo largo del año. Mostrar también una leyenda que indique cuál es el mes con mayor cantidad de cumpleaños. Recuerde no utilizar la función max, sino crear su propia función máximo.

Ejercicio 3: Una tienda ofrece descuentos a sus clientes según la compra que realizan: Si el monto de la compra está entre 10000 y 20000, ofrece un 15% de descuento. Si el monto supera los 20000, ofrece un 20% de descuento. Desarrollar un programa que permita atender a los clientes, para cada cliente ingresa la cantidad de productos comprados y para cada producto ingresa cantidad comprada y el precio unitario, luego debe informar el monto a pagar sin decimales. A continuación, solicitar el monto que entrega el cliente para pagar e indicar cuantos billetes de 1000, 500, 200, 100, 50, 20, 10, 5 y monedas de 2 y 1 debe dar el vendedor considerando que le entregue la menor cantidad de billetes posibles. Considerar que puede ingresar un monto menor al comprado, mostrar un mensaje aclaratorio en ese caso. Al finalizar el día el dueño de la tienda desea ver un listado de todas las compras realizadas en el orden que fueron comprando y el monto total vendido.

Ejercicio 4: Una fábrica de bicicletas desea realizar un control de inventario de sus productos, la empresa fabrica N productos diferentes y quiere mantener en stock un mínimo de 5 unidades por producto. Necesita un programa que ingrese código del producto positivo y simule la cantidad en stock actual (valor entre 1 y 50). Si se vuelve a ingresar un código de producto, se debe rechazar. La carga de datos finaliza con un código de producto igual a -1.

Luego de la carga, informar:

- ✓ Código del producto y cantidad en stock actual. Ordenado por Código de producto de menor a mayor. Con el siguiente formato:

Código	**Cantidad**
123	20
243	10
450	3
8080	15

- ✓ Código del producto y cantidad en stock actual de los productos que están por debajo del stock mínimo. A continuación, indicar cuál es el total general de unidades faltantes para que todos los productos tengan al menos las 5 unidades en stock.
- ✓ Mayor cantidad en stock y todos los códigos de productos que tienen la mayor cantidad en stock.

Realizar la búsqueda de códigos de producto ingresados por teclado e informar su cantidad en stock con una leyenda si está por debajo del stock mínimo. Finalizar el ingreso de códigos para la búsqueda con -1. Informar cuántos productos de los buscados fueron encontrados.

Ejercicio 5: Desarrollar un programa para crear una lista con números al azar entre 1 y 20. Finalizar la carga al crear el número cero y contemplar además que la lista debe tener al menos 10 elementos cargados. Mostrar la lista por pantalla. Luego se pide: Informar cuántos números NO están repetidos. Desarrollar una función que reciba la lista y retorna cuántos números únicos hay en la lista.

Ejercicio 6: Una Administradora de Consorcios necesita un sistema para poder gestionar el cobro de las expensas de un edificio de departamentos de 20 unidades. En 2 listas almacena la siguiente información: número de unidad y superficie en metros cuadrados. Validar que no se ingresen números de unidades duplicadas. Cada unidad paga $100 de expensas por metro cuadrado por mes. Se pide:

a) Informar el promedio de expensas del mes.
b) Ordenar los listados de mayor a menor según la superficie. Mostrar por pantalla el listado ordenado informando Número de Unidad y Superficie en metros cuadrados.

Ejercicio 7: Una empresa de colectivos debe renovar su flota de vehículos que tengan más de 20 años de antigüedad. Para eso necesita un programa que le permita llevar el control de dicha renovación. El programa deberá realizar las siguientes tareas:

a) Ingresar el año de renovación (validar que el año sea mayor o igual al año actual).
b) Ingresar el número de interno de cada uno de los vehículos y a continuación su modelo (año de fabricación). Finalizar la carga ingresando -1 en el número de unidad.
c) Al ingresar los datos del vehículo, validar que el modelo sea mayor a 1980 y menor al año de renovación.
d) Mostrar por pantalla un listado con todos los datos de los vehículos que deben ser renovados (número de interno, modelo).
e) Informar cuántos vehículos posee la empresa y qué porcentaje de éstos debe ser renovado.
f) Informar el año de fabricación de la unidad más antigua de la empresa.

Ejercicio 8: Construir una lista llamada SECUENCIAS con N números enteros al azar entre 1 y 20. Esta lista se caracterizará porque sus valores deben encontrarse divididos en secuencias de números separadas por ceros, cuya suma no sea mayor que 20. Para eso se deberá agregar un elemento de valor 0 a fin de separar cada secuencia de la siguiente, cuidando que ninguna secuencia sume más de 20. Agregar un 0 adicional al final de la lista y mostrar la lista obtenida por pantalla.

Ejemplo:

Lista original:

5	2	9	6	4	15	3	19	12	1	5

Resultado:

5	2	9	0	6	4	0	15	3	0	19	0	12	1	5	0

Ejercicio 9: A partir de la lista SECUENCIAS generada en el ejercicio anterior, mostrar la secuencia más larga almacenada en la misma. Si hubiera varias secuencias con la misma longitud máxima deberán mostrarse todas las que correspondan.

Ejercicio 10: Yuki es una chica con suerte. Su abuela Chika es una encantadora anciana, propietaria de una granja de ovejas, experta repostera y ninja de la costura, y tiene un avellanero detrás de casa. Yuki pide avellanas continuamente, pero su sabia abuela no permite que Yuki coma demasiadas para que no caiga enferma. Chika es aficionada a los números binarios, así que propuso este juego a Yuki: ella diría un número N y Yuki escogería dos números, x e y, tales que x + y = N. Entonces, Chika daría a Yuki tantas avellanas como la cantidad de unos que tengan x e y en base 2. Por ejemplo, si Chika propone 7 y Yuki escoge 3 y 4, recibiría 2 (porque 3 en base 2 es 11) y 1 (porque 4 en base 2 es 100): 3 avellanas. Yuki quiere saber qué números debe escoger para obtener la máxima cantidad de avellanas que podría obtener de Chika dado el número que le dice su abuela.

Trabajo Práctico 9: Estructuras de datos Matrices

Ejercicio 1:
Desarrollar cada una de las siguientes funciones y escribir un programa que permita verificar su funcionamiento, imprimiendo la matriz luego de invocar a cada función:

a. Cargar números enteros en una matriz de N x N, ingresando los datos desde teclado.
c. Intercambiar dos filas, cuyos números se reciben como parámetro.
d. Intercambiar dos columnas dadas, cuyos números se reciben como parámetro.
e. Calcular el promedio de los elementos de una fila, cuyo número se recibe como parámetro.
f. Calcular el porcentaje de elementos con valor impar en una columna, cuyo número se recibe como parámetro.

NOTA: El valor de N debe leerse por teclado. Las funciones deben servir cualquiera sea el valor ingresado.

Ejercicio 2:
Para cada una de las siguientes matrices, desarrollar una función que la genere y escribir un programa que invoque a cada una de ellas y muestre por pantalla la matriz obtenida. El tamaño de las matrices debe establecerse como N x N, y las funciones deben servir, aunque este valor se modifique.

a:

1	0	0	0
0	3	0	0
0	0	5	0
0	0	0	7

b:

0	0	0	27
0	0	9	0
0	3	0	0
1	0	0	0

c:

8	8	8	8
4	4	4	4
2	2	2	2
1	1	1	1

d:

0	1	0	2
3	0	4	0
0	5	0	6
7	0	8	0