FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS Departamento de Tecnología Informática

FUNDAMENTOS DE INFORMÁTICA

Guía de Trabajos Prácticos

EJERCICIOS DE CLASE .

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 unidimensionales y bidimensionales

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: 17/02/21] 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: 17/02/21] https://openlibra.com/es/book/introduccion-a-la-programacion-con-python-3

• The Python Language Reference. [consultas: 17/02/21]. Español: https://docs.python.org/3/reference/

• Documentos varios en Python.org.ar [consultas: 17/02/21]. Inglés / Español: https://wiki.python.org.ar/aprendiendopython/

Autor:

Ricardo Thompson, Patricia Mazzitelli, Felipe D'Aquino, Verónica Galati.. Verónica Galati, Ramiro Escalante Leiva, Guillermo Maquieira, Manuel Quintana.

Equipo docente de la cátedra.

Trabajo Práctico 1: Algoritmos elementales

Objetivos:

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

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. Llegar a la vereda desde la posición en la que te encontrás.
b. Cruzar una calle en una esquina con semáforos.
c. Lavarse las manos.
d. Preparar un mate.
e. Destapar una botella de cerveza.
f. Poner una alarma en el reloj del celular.
g. Pintar una pared.
h. Comprar una entrada de cine por Internet.
i. Resolver la siguiente cuenta: 4 + 2 * 3
j. Separar el cuatro de copas de un conjunto de naipes.
k. Buscar la menor carta de un mazo de naipes, que puede no estar completo.
l. Ordenar un conjunto de naipes.

Trabajo Práctico 2: 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. • Crear los primeros algoritmos en pseudocódigo o diagrama de flujo con estructura secuencial. • Ser capaz de escribir el algoritmo en lenguaje Python.

Ejercicio 1: Diseñar el algoritmo para resolver los siguientes problemas y escriba el programa correspondiente en lenguaje Python.

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 la suma y el promedio.
e. Ingresar el monto de una factura y calcular el IVA (21%).

Ejercicio 2: Calcular el valor de las siguientes expresiones aritméticas, respetando el orden establecido, indicar

el orden que ejecuta el intérprete Python cada una de ellas. Ejemplo: 12*4+4*5 = 68 orden: multiplicación, suma:

a. 12 * 4 + 4 * 5 (68)
b. 12 * (4 + 4) * 5 (480)
c. 5 * 4 / 2 (10.0)
d. 5 * (4 / 2) (10.0)
e. 24 / 2 ** 2 (6.0)
f. (24 / 2) ** 2 (144.0)
g. – 9 ** 2 (-81)
h. (– 9) ** 2 (81)
i. 10 / 3 (3.333333…)
j. 10 // 3 (3)
k. *12 % 5 (2)*

Para los siguientes ejercicios, identifique las entradas y salidas. Diseñe el algoritmo en pseudocódigo y luego escriba el programa en lenguaje Python.

Ejercicio 3: Calcular el promedio de las notas que obtiene un alumno al rendir los dos parciales.

Ejercicio 4: Tres personas invierten dinero para fundar una empresa (no necesariamente en partes iguales). Calcular qué porcentaje invirtió cada una.

Ejercicio 5: 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 6: 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 7: 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 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 de $10, $5, $2 y $1. Desarrollar dicho programa de tal forma que se minimice la cantidad de billetes entregados por el cajero.

Ejercicio 9: 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 10: 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 (2^x) 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 pseudocódigo o diagrama de flujo combinando estructura secuencial y condicional.
- Ser capaz de escribir el algoritmo en lenguaje Python.

Para los siguientes ejercicios, identifique las entradas y salidas. Diseñe el algoritmo en pseudocódigo y luego escriba el programa en lenguaje Python.

Ejercicio 1: Ingresar dos números enteros e indicar si son iguales o distintos.

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

Ejercicio 3: Crear un programa que pida un número de mes (ejemplo 4) y escriba el nombre del mes en letras ("abril"). Verificar que el mes sea válido e informar en caso que no lo sea.

Ejercicio 4: Ingresar las notas de los dos parciales de un alumno e indicar si promocionó, aprobó o debe recuperar. Si el valor de la nota no está entre 0 y 10, debe informar un error.

- 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: Una editorial determina el precio de un libro según la cantidad de páginas que contiene. El costo básico del libro es de $50000, más $2000,20 por página con encuadernación rústica. Si el número de páginas supera las 300 la encuadernación debe ser en tela, lo que incrementa el costo en $20000. Además, si el número de páginas sobrepasa las 600 se hace necesario un procedimiento especial de encuadernación que incrementa el costo en $33600. Desarrollar un programa que calcule el costo de un libro dado el número de páginas.

Ejercicio 6: Una remisería requiere un sistema que calcule el precio de un viaje a partir de la cantidad de km que desea recorrer el usuario. Tiene la siguiente tarifa:

- Viaje mínimo $5000
- Si recorre entre 0 y 10km: $2000/km
- Si recorre 10km o más: $1500/km

Ejercicio 7: 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.

Trabajo Práctico 4: Estructura iterativa

Objetivos:

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

Para los siguientes ejercicios, identifique las entradas y salidas. Diseñe el algoritmo en pseudocódigo y luego escriba el programa 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: Leer 10 números enteros e imprimir el promedio, el mayor y en qué orden fue ingresado el mayor valor, si se ingresó más de una vez debe informar el primer ingreso.

Ejercicio 5: 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. Piense cómo debe inicializar las variables.

Ejercicio 6: Ingresar números, hasta que la suma de los números pares supere 100. Mostrar Cuántos números en total se ingresaron.

Ejercicio 7: 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 8: Se desea analizar cuántos autos circulan con patente que tiene numeración par y cuántos con numeración impar en un día. Le solicitan escribir un algoritmo que permita ingresar la terminación de la patente (entre 0 y 9) hasta ingresar -1 e informar cuántas se ingresaron con numeración par y cuántas con numeración impar.

Ejercicio 9: 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 10: 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 11: Cada cliente que va al banco Express, indica su número de documento y aguarda a ser atendido, cuando finaliza la atención del día se ingresa -1 para indicar que no hay más clientes para ser atendidos. El banco desea saber quién fue el primer cliente atendido y quién fue el último.

Ejercicio 12: Ingresar un número n, validar que sea positivo. Luego:
a) Mostrar los primeros n números impares hasta el número ingresado.
Ejemplo:

- Si se ingresa 5, se debe mostrar 1 3 5
- Si se ingresa 8, se debe mostrar 1 3 5 7
- Si se ingresa -5, se debe pedir otro número.

b) Informar la suma de esos números impares.

Trabajo Práctico 5: Ejercicios Integradores

Objetivos:

- Plantear estrategias de resolución de problemas creando algoritmos.
- Entender y saber construir un programa estructurado.
- Diferenciar los distintos tipos de estructuras de control.

En los ejercicios siguientes identifique las entradas y salidas, diseñe el algoritmo utilizando pseudocódigo o diagrama de flujo.
Describa con datos de ejemplos al menos tres conjuntos de datos para las entradas posibles y la salida que se debe obtener en cada uno de los ejemplos.
Desarrolle el programa en lenguaje Python.

Ejercicio 1: Leer números que representan edades de un grupo de personas, finalizando la lectura cuando se ingrese el número 999. Imprimir cuántos son menores de 18 años, cuántos tienen 18 años o más y el promedio de edad de ambos grupos. Descartar valores que no representan una edad válida. (Se considera válido una edad entre 0 y 100).

Ejercicio 2: Escribir un algoritmo que permita ingresar los números de legajo de los alumnos de un curso y su nota de examen final. El fin de la carga se determina ingresando un -1 en el legajo. Informar para cada

alumno si aprobó o no el examen considerando que se aprueba con nota mayor o igual a 4. Se debe validar que la nota ingresada sea entre 1 y 10. Terminada la carga de datos, informar:

- Cantidad de alumnos que aprobaron con nota mayor o igual a 4.
- Cantidad de alumnos que desaprobaron el examen. Nota menor a 4.
- Porcentaje de alumnos que están aplazados (tienen 1 en el examen).

Ejercicio 3: Una empresa aplica el siguiente procedimiento en la comercialización de sus productos:

- Aplica el precio base a la primera docena de unidades.
- Aplica un 10% de descuento a todas las unidades entre 13 y 100.
- Aplica un 25% de descuento a todas las unidades por encima de las 100.

Por ejemplo, supongamos que vende 230 unidades de un producto cuyo precio base es 100. El cálculo resultante sería:

*100 * 12 + 90 * 88 + 75 * 130 = 18870, y el precio promedio será 18870 / 230 = 82,04*

Escribir un algoritmo que lea la cantidad solicitada de un producto y su precio base, y muestre el valor total de la venta y el precio promedio por unidad. El fin de carga se determina ingresando -1 como cantidad solicitada.
Al finalizar informar:
a) Cantidad de ventas realizadas total.
b) Cantidad de ventas que se aplicaron un 10% de descuento.
c) Cantidad de ventas que SOLO se aplicó el precio base, no se le realizaron descuentos.

Ejercicio 4: Una empresa factura a sus clientes el último día de cada mes. Si el cliente paga su factura dentro de los primeros 10 días del mes siguiente, tiene un descuento de $120 o del 2% de la factura, lo que resulte más conveniente para el cliente. Si paga en los siguientes 10 días del mes deberá pagar el importe original de la factura, mientras que si paga después del día 20 deberá abonar una multa de $150 o del 10% de su factura, lo que resulte mayor. Escriba un algoritmo que lea el número del cliente y el total de la factura, y emita un informe donde conste el número del cliente y los tres importes que podrá abonar según la fecha de pago.

Ejercicio 5: El factorial de un número entero N mayor que cero se define como el producto de todos los enteros X tales que 0 < X <= N. Desarrollar un programa para calcular el factorial de un número dado hasta ingresar -1. Deberán rechazarse las entradas inválidas (menores que 0). Al finalizar informar cuántas veces se calculó el factorial.

Ejercicio 6: Leer tres números D, M y A correspondientes al día, mes y año de una fecha, y un número entero N que representa una cantidad de días. Realizar un programa que imprima la nueva fecha que resulta de sumar N días a la fecha dada. Tener en cuenta los años bisiestos tal como se detalla en el ejercicio 9 de la práctica 3.

Ejercicio 7: Una empresa cuenta con N empleados, divididos en tres categorías A, B y C. Por cada empleado se lee su legajo, categoría y salario. Se solicita elaborar un informe que contenga:

- Importe total de salarios pagados por la empresa.
- Cantidad de empleados que ganan más de $20000.
- Cantidad de empleados que ganan menos de $5000, cuya categoría sea “C”.
- Legajo del empleado que más gana.
- Sueldo más bajo.
- Importe total de sueldos por cada categoría.
- Salario promedio.

Ejercicio 8: Ingresar por teclado la cantidad de términos a generar de la siguiente serie:

1 7 19 43 91 187 379 763 1531 3067 6139
El primer término es el 1 y cada término se genera como el doble del término anterior más 5. Mostrar la serie por pantalla e informar la suma de los términos generados.

Trabajo Práctico 6: Sub-Algoritmos: Funciones

Objetivo:

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

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

Ejercicio 1: Diseñar una función que reciba dos parámetros numéricos enteros, calcule y devuelva el resultado de la multiplicación de ambos utilizando sólo sumas.
Desarrollar un programa principal para crear la siguiente serie numérica de N términos, comienza en uno y cada siguiente término se obtiene multiplicando el anterior por la ubicación: 1 2 6 24 120
*2 *3 *4 *5

Ejercicio 2: Diseñar una función que reciba dos números enteros como parámetros enteros A y B, y permita obtener A^B mediante multiplicaciones sucesivas.
Desarrollar un programa principal para generar N veces dos valores al azar en un rango desde hasta ingresado por teclado y calcular A^B, mostrar por pantalla los valores creados y el resultado de la operación.

Ejercicio 3: 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) muestra:

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

Desarrollar un programa principal para mostrar el título: "Aprendiendo Funciones", del ejercicio 4 al 8 agregar un título al iniciar el programa relacionado a lo que va a resolver el problema.

Ejercicio 4: Desarrollar dos funciones
1) Función para sumar los dígitos de un número. Recibe un número y retorna la suma de los dígitos. (NO utilizar cadenas de caracteres str, para lograr el objetivo)
2) Extraer un dígito de un número. La función recibe como parámetros dos números enteros, uno será del que se extraiga el dígito y el otro indica qué cifra se desea obtener. La cifra de la derecha se considera la número 0. Retornar el valor -1 si no existe el dígito solicitado. Tener en cuenta que el número puede ser positivo o negativo. Ejemplo: extraerdigito(12345, 1) devuelve 4, y extraerdigito(12345, 8) devuelve -1.
Desarrollar un programa para generar valores al azar de 5 dígitos hasta que el dígito central sea cero. Mostrar por pantalla este número y la suma de sus dígitos utilizando ambas funciones creadas y no olvidar mostrar un título al inicio utilizando la función del ejercicio 3

Ejercicio 5: Desarrollar dos funciones:
1) Diseñar una función que solicite por teclado un número y lo retorne solo si el número ingresado es natural, caso contrario la función deberá seguir solicitando el número. 2) Función para sumar los primeros N números naturales de un valor. Retorna la suma. Desarrollar un programa principal para ingresar una cantidad de valores naturales (la cantidad se solicita al usuario). Para cada valor informar la suma de los primeros N valores naturales. Al finalizar informar cuántos valores se ingresaron y cuál es el mayor valor ingresado.

Ejercicio 6: **Hora de jugar**: Una calculadora tiene cuatro operaciones básicas (a saber: sumar, restar, multiplicar, dividir). Desarrolle una función para realizar cada operación, que reciba como parámetros dos números ingresados por el usuario y devuelva el resultado de la operación. Resuelva la división por restas sucesivas (investigar cómo se resuelve).
Desarrollar un programa principal con un menú que permita realizar una operación y posea una opción para Salir. Luego de cada operación realizada se debe volver a presentar el menú.

Trabajo Práctico 7: Estructuras de datos Listas o Arreglos

Objetivos:

- Conocer y utilizar estructuras de datos – Listas en Python, conocidas como arreglos o vectores en otros lenguajes de programación.

Desarrollar las siguientes funciones para crear listas con valores según lo solicitado:

Ejercicio 1: Escribir una función que solicite ingresar una serie de números entre a y b y guardarlos en una lista. En caso de ingresar un valor fuera de rango el programa mostrará un mensaje de error y solicitará un nuevo número. Para finalizar la carga se deberá ingresar -1. La función no recibe ningún parámetro, y devuelve la lista cargada (o vacía, si el usuario no ingresó nada) como valor de retorno.

Ejercicio 2: Escribir una función para crear una lista con N números al azar en un rango de valores que se recibe por parámetro. La función devuelve la lista cargada (o vacía si el rango indicado no es válido).

En los siguientes ejercicios utilice una de las funciones anteriores para ingresar datos en una lista y luego resuelva diseñando y creando nuevas funciones y un programa principal:

Ejercicio 3: Calcular la suma de los números de una lista.

Ejercicio 4: Desarrollar un algoritmo que permita crear al azar 5 números pertenecientes a la lista A y otros 5 números pertenecientes a la lista B. Crear una lista C, donde cada posición es el resultado de la suma del número en la misma posición en la lista A con el número en la misma posición en la lista B. Ejemplo: Se crea A = [1, 2, 3, 4, 5] y B = [4, 7, 1, 3, 6] → C = [5, 9, 4, 7, 11]

Ejercicio 5: Rellenar una lista con números enteros entre 0 y 100 obtenidos al azar e 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. La carga de datos termina cuando se obtenga un 0 como número al azar, el que no deberá cargarse en la lista.

Ejercicio 6: *Determinar si una lista es capicúa.*

Ejercicio 7: 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.

Ejercicio 8: 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 9: 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 10: Utilizando **búsqueda binaria** sobre una lista ordenada realizar la búsqueda de valores e informar si se encuentran o no en la lista, finalizar las búsquedas con -1 e informar cuantas búsquedas fueron exitosas y en cuantas no se encontró el valor buscado.

Ejercicio 11: Crear una lista de N números generados al azar entre 0 y 100 pero sin elementos repetidos. Validar que N sea menor o igual a 100.

Trabajo Práctico 8: Listas de Listas - Matrices

Objetivos:

• Conocer una nueva estructura de datos que permite relacionar la información. Listas de Listas – Matrices.

Ejercicio 1: Crear una matriz de 3x4 (3 filas y 4 columnas) con valores creados al azar entre 1 y 10. Mostrar la matriz creada respetando el formato de 3 filas y 4 columnas por pantalla.

Ejercicio 2: Generar una matriz cuadrada, ingresando por teclado la cantidad de filas/columnas. Informar

cual es la suma de los elementos ubicados en el triángulo superior de la matriz

Ejercicio 3: Ingresar por teclado la cantidad de filas y de columnas de una matriz. Generarla con valores al azar comprendidos entre a y b. Mostrar la suma de los valores ubicados sobre la diagonal principal

Ejercicio 4: Indicar las coordenadas del mayor valor encontrado en una matriz n*m, generada con valores aleatorios entre 100 y 1000.

Ejercicio 5: Desarrollar una función para crear una matriz de NxM (n filas y m columnas). La función recibe n y m por parámetro, la completa con valores al azar entre a y b también por parámetro y retorna la matriz creada. Si no hay valores entre a y b o alguna de las dimensiones es negativa, retornar la matriz vacía. Desarrollar un pequeño programa principal crear una matriz al azar utilizando la función y mostrar por pantalla la matriz y la suma de cada fila y cada columna.

Ejercicio 6: Utilizando la función creada en el ejercicio 2, desarrollar un programa para crear dos matrices de 3x3 con valores al azar entre dos números ingresados por teclado. Verificar que el rango sea válido, caso contrario solicitar nuevamente ambos valores. Calcular la suma de sus elementos y mostrar la matriz resultante.

Ejercicio 7: Ingresar una cantidad de números, debe ser múltiplo de 4. Luego crear una matriz que contenga 4 elementos por fila, hasta completar la cantidad de elementos indicada. Mostrar la matriz e informar cuántas filas se crearon. Puede crear los valores al azar para crear la matriz.

Ejercicio 8: Desarrollar una función que genere la siguiente matriz NxM

1	3	5	7
2	4	6	8
9	11	13	15
10	12	14	16

Escribir un programa que utilice la función y muestre por pantalla la matriz obtenida.

Ejercicio 9: *Crear una matriz cuadrada y luego crear una lista que contenga la suma de cada una de las filas de la matriz sin valores repetidos, o sea, dos filas suman igual, el valor debe estar una sola vez en la lista. Mostrar la lista por pantalla la lista ordenada de menor a mayor*

Ejercicio 10: Ingresar valores desde el teclado en una matriz de NxM y mostrar la matriz creada. Ordenar cada una de las filas por el método de selección. Mostrar nuevamente la matriz.

Trabajo Práctico 9: Resolver Problemas de forma simple y práctica

Objetivos:

- Plantear estrategias de resolución de problemas creando algoritmos.
- Diferenciar los distintos tipos de estructuras de control.

Ejercicio 1: Leer los números de legajo de los alumnos de un curso y su nota de examen 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 mostrar un listado de manera ascendente según el número de legajo. Ingresar números de legajo hasta que se encuentre e informar la nota de examen final utilizando búsqueda binaria.
Resolver de dos formas: Utilizando dos listas - Utilizando una matriz de dos filas.

Ejercicio 2: 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 un valor fijo para todo el edificio 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 3: **Hora de jugar:** 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, haciendo que el usuario ingrese su número de documento si mejoró la mejor marca de intentos obtenida hasta el momento. Luego mostrar la lista ordenada de los 5 mejores puntajes (indicando también a quién pertenecen) y preguntar si se desea jugar otra vez, reiniciando el juego en caso afirmativo.

Ejercicio 4: Modificar el programa anterior para que las pistas brindadas por el programa no sean del tipo "es mayor" o "es menor" sino "M dígitos correctos y N dígitos aproximados". Se considera que un dígito es correcto cuando tanto su valor como su posición coinciden con los del número secreto, mientras que un dígito es aproximado cuando coincide el valor, pero no su posición. Ejemplos:

Número secreto: 5739

- Intento 1: 1234 -> 1 correcto
- Intento 2: 5678 -> 1 correcto y 1 aproximado
- *Intento 3: 9375 -> 4 aproximados*

Ejercicio 5: 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.

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