Universidad de Buenos Aires

Facultad de Ingeniería

Guía de Ejercicios

7540 – 9514 Algoritmos y Programación I

Índice

Introducción

La intención de esta guía es que aprendas a resolver problemas mediante algoritmos bajo programación estructurada. Los mismos se pueden implementar en el lenguaje que estés estudiando, ya sea Pascal, C, Python o cualquier otro que soporte el paradigma estructurado.

Los problemas tienen un orden progresivo en dificultad y están divididos en secciones que serán los temas que se van a ir viendo en la teórica. Cada nueva sección incluye a las anteriores, de esta forma, en la sección de vectores y matrices, por ejemplo, aunque no se mencione hay que manejar y aplicar funciones y procedimientos. Por esto no recomendamos pasar a una nueva sección sin tener bien en claro la anterior.

¿Cómo tenés que trabajar con esta guía para que dé sus resultados?

La idea es que comiences por el ejercicio 1 de la primera sección y te dejes llevar por las recomendaciones que se indican y las respetes al pie de la letra. Es preferible estar analizando un buen rato un solo ejercicio que hacer varios en forma descuidada.

Lo ideal es hacer todos los ejercicios. Aunque, la realidad, es que pueden llevar mucho tiempo del que no siempre se dispone. Si este es el caso, podés ir eligiendo los ejercicios que querés hacer, cuidando que, por lo menos, representen el 60% del total de la guía.

Del total de los ejercicios que hagas, se recomienda que, por lo menos, un 30% los hagas en papel antes de codificarlo en alguna máquina. ¿Por qué? Porque cuando uno lo escribe en papel piensa más que cuando lo hace en la máquina. El deseo de resolverlo en forma rápida hace que no le dediques al problema el tiempo que realmente necesita. Por otra parte, es habitual que cuando se escribe en una máquina, al detectar que no funciona correctamente en alguna prueba, toque el código en forma indiscriminada, sin pensar mucho lo que está haciendo, solo manejándose por los resultados de la salida, de manera “prueba y error”. Esto no es para nada recomendable.

También, es bueno que, después de resolver algún ejercicio en papel, hagas un seguimiento de escritorio. Es decir, tomes algunos valores de entrada y sigas el algoritmo anotando los valores que van tomando las variables a cada paso.

Otra cosa que recomendamos es que seas prolijo a la hora de hacer los ejercicios: los tengas todos en un cuaderno y los escritos en máquina estén en un directorio aparte para que los puedas ubicar fácilmente. No sobre-escribas ejercicios. Si hacés distintas versiones de un mismo ejercicio, guardalas todas indicando v0, v1, etc. Cada tanto es bueno rever lo que se hizo, mirar los primeros ejercicios hechos cuando se está por la

mitad de la guía sirve para medir el avance realizado. Además, en muchos ejercicios se piden cosas que, en parte, ya se pidieron anteriormente. Todo lo que puedas reutilizar será bienvenido porque, por un lado, te ahorra tiempo y, por otro, te acostumbra a evitar re escribir código, lo cual es uno de los objetivos que se irán persiguiendo en todo el curso.

Cuando se resuelve un problema, la primera meta a conseguir es que tu algoritmo funcione y cumpla lo que se pide. Pero no te tenés que conformar con eso. De a poco empezá a buscar calidad en tu código. Hacete alguna de estas preguntas:

- ¿Podría haber logrado la solución en forma más eficiente? ¿Con menos líneas de código? ¿Con menos ciclos? ¿Usando menos variables?
- ¿Los nombres de las variables, funciones y procedimientos son claros y representativos?
- ¿La salida por pantalla es clara y agradable a la vista? ¿Se pide intervención del usuario innecesariamente? ¿El usuario sabrá qué datos tiene que ingresar y qué datos le muestran?
- ¿El código es legible? ¿Dejé sangrías correctamente en cada bloque?
- ¿Podía haber modularizado mejor el código? ¿Cada función y procedimiento se dedica a resolver una sola cosa?

El aprendizaje de resoluciones algorítmicas es como cualquier otro aprendizaje: se necesita práctica, tiempo de dedicación y que sea progresivo. A una persona que jamás entrenó en su vida no se le puede pedir que corra un maratón. En cambio, si uno empieza corriendo pequeñas distancias y las va aumentando en forma progresiva, al final podrá hacerlo.

¡Éxitos!

Estructuras consecutivas o secuenciales

- *Los siguientes seis ejercicios escribilos en papel, en pseudocódigo.*
- *Para el ejercicio 4 o 5 hacer un diagrama de flujo.*

1. Leer un número por teclado e imprimirlo en pantalla con el siguiente cartel: "Número ingresado = " número.

2. Pedir al usuario que ingrese dos números por teclado e imprimir

- La suma de ambos
- La resta (el primero menos el segundo)
- La multiplicación
- La división entera (suponer que el segundo número no es cero).
- La división con decimales.

3. Escribir un programa que lea el nombre de una persona y, luego, lo salude.

4. Dado el radio R de una esfera, calcular e imprimir su superficie y su volumen.

5. Leer la base y la altura de un rectángulo, calcular el perímetro y la superficie.

6. Leer dos números A y B e intercambiar el valor de sus variables.

- *Ahora, los mismos ejercicios, escribilos en el intérprete de Python y probalos.*
- *Para los ejercicios 4 y 5 escribir (en papel) algunos datos de entrada para R, la base y la altura y escribir la salida esperada (con tres datos de entrada alcanzan). Al escribirlos y correrlos en Python ingresá estos datos y verificá la salida.*

Estructuras selectivas o condicionales

- *Los primeros seis ejercicios escribilos en papel, en primer lugar, luego escribilos en máquina y probalos.*

1. Leer un número real y decir si es mayor, menor o igual a cero.

2. Leer dos números reales e imprimir el mayor de ellos.

3. Dadas las horas de partida y de llegada de un móvil, expresadas en horas, minutos y segundos, calcular su velocidad promedio sabiendo que la distancia recorrida es D (D es un valor en km que se ingresa por teclado).

4. Escribir un algoritmo que determine si un número entero, que ingresa un usuario, es par.

- *Para el ejercicio 5 escribir en papel algunos datos de entrada y escribir la salida esperada (con tres datos de entrada alcanzan). Al correrlo en Python ingresar estos datos y verificar la salida.*
- *Hacer también un diagrama de flujo.*

5. Escribir un algoritmo que determine si un número entero, que ingresa un usuario, es par.

6. Escribir un algoritmo que determine si un número N (entero), que ingresa un usuario, es divisible por M.

7. Leer dos números y, luego, una opción (puede ser suma, resta, multiplicación o división), según la opción elegida realizar el cálculo.

- *Una vez escrito el ejercicio 6, andá al siguiente link: Pregunta y respondé las preguntas.*

8. Formar un menú con 4 opciones y al elegir una de ellas mostrar un cartel diciendo qué opción se eligió o si fue una opción incorrecta. (+)
    - Opción 1
    - Opción 2
    - etc

9. Pasar un período expresado en segundos a un período expresado en días, horas, minutos y segundos.

- *Una vez escrito el ejercicio anterior, andá al siguiente link: Pensá.*

10. Hacé el seguimiento, en papel, de los siguientes códigos y decidí si hacen lo mismo. Indicá qué imprimen:

```
x = 3
if (x < 5):
   x = x + 1
if (x == 5):
   x = x + 2
if (x > 5):
   x = x + 3
print(x)
```

```
x = 3
if (x < 5):
   x = x + 1
elif (x == 5):
   x = x + 2
else:
   x = x + 3
print(x)
```

11. Escribí los dos códigos del ejercicio anterior en una máquina y probá que la salida sea la que esperabas.

12. Volvé a hacer el seguimiento, primero en papel y, luego, verificando en máquina, de los códigos del ejercicio 10 pero cambiando x de 3 a 4 en la primera línea:

```
x = 4
```

- Ahora que finalizaste esta sección probá, en la máquina todas las dudas que tengas, todas las preguntas que te hiciste, experimentá y escribilas a ver qué pasa. Por ejemplo, ¿qué pasa si se pide el ingreso de un número entero y el usuario ingresa un real? Etc. Anotá lo que sucede.

Estructuras repetitivas

- Resolvé los ejercicios en papel (hasta el 5). Después escribilos en el intérprete y probalos.

1. Imprimir por pantalla una lista de 20 números consecutivos, los cuales comienzan con un número ingresado por teclado.

2. Leer un número N y calcular su factorial.

3. Leer una serie de números enteros, terminando la serie con un cero. Imprimir los datos a medida que se los ingresa junto con la suma parcial de los mismos.

4. Dada una serie de números enteros, determinar el valor máximo, mínimo y las posiciones en que éstos se encontraban en la serie. El programa deberá ir preguntando si hay más números para ingresar.

- *Después de escribir este ejercicio andá al siguiente link: verificar.*

5. Leer un valor N y, luego, N números enteros. Se pide imprimir el mayor, el menor y las veces que aparece cada uno.

6. Leer A y B, enteros. Calcular C = A x B mediante sumas sucesivas e imprimir el resultado.

- Antes de escribir el ejercicio siguiente hacé un diagrama de flujo. Luego, escribilo y probalo.

7. Leer A y B, enteros. Calcular C = A^B mediante multiplicaciones sucesivas e imprimir el resultado. Tener en cuenta que son números enteros, por lo tanto pueden tomar valores positivos, negativos o cero.

8. Escribir un algoritmo que indique si un número entero, ingresado por un usuario, es primo.

- *Después de escribir el ejercicio anterior andá al siguiente link: investigar.*

9. Dada una serie de números enteros terminada en cero, imprimir los tres mayores.

10. Dada una serie de nombres y salarios respectivos, determinar el salario máximo, el mínimo y la persona que percibe cada uno. No se deben guardar todos los datos.

Funciones

- *Nota: si desean hacer todos los ejercicios, para no atrasarse, pongan un máximo de 20 minutos para cada uno. Cuando empiecen a hacer uno, tomen el tiempo (acá hay un cronómetro: http://cronometro-online.chronme.com/) y si a los 20 minutos no terminaron, márquenlo como pendiente y sigan con otro.*
- *Otra opción es que le dediquen media hora a cada uno pero haciendo la mitad de los ejercicios, solo los impares o los pares.*

1. Hacer una función que, dado los coeficientes de un polinomio de segundo grado (3 números reales), indique si tiene o no raíces reales (devuelve verdadero o falso).

2. Hacer una función que invocada como expo(x,n) devuelva el valor de x a la n, donde x es un número real y n entero (puede ser negativo). Resolverla con multiplicaciones sucesivas.

3. Hacer una función que devuelva las raíces reales de un polinomio de segundo grado y, además, indique si tiene o no raíces reales (una booleana).

4. Hacer una función que indique si un número grande es primo o no. Utilizar lo realizado en el ejercicio 8 de la práctica anterior.

5. Hacer un programa que liste todos los números primos desde 2 hasta un número ingresado por el usuario utilizando la función realizada en el ejercicio anterior.

- *Después de escribir este ejercicio andá al siguiente link: compará.*

6. Escribir un programa, utilizando funciones, que descomponga un número en sus factores primos. Agregale comentarios al programa donde lo creas necesario.

- De acá en adelante, todos los programas deberán tener, como mínimo, una función.
- Antes de escribir el ejercicio 7 hacé un diagrama MODULAR.

7. Realizar un algoritmo que lea una serie de números reales y verifique si están ordenados en forma ascendente, descendente o si no están ordenados, informando por pantalla.

8. Dada una serie de datos de la forma mes (1 a 12, no vienen ordenados), cantidad recaudada (en pesos) y costo total (en pesos), hacer un algoritmo que calcule e

imprima cuál fue el mes que arrojó mayor ganancia. La serie termina con mes igual a cero. No se deben guardar los datos.

9. Se leen 300 datos (usar constantes para poder achicar esta cantidad) que representan el peso de la misma cantidad de niños que hay internados en un hospital. Se desea confeccionar la siguiente tabla:
- Entre 0,000 y 10,000 kg. hay ..... niños.
- Entre 10,001 y 20,000 kg. hay ..... niños.
- Entre 20,001 y 30,000 kg. hay ..... niños.
- Más de 30,000 kg. hay ..... niños.

- Nota: Probar el ejercicio modificando 300 por 15, tratar de repartir los valores en diferentes rangos. Verificar. Luego, utilizando la función random, simular el ingreso de los 300 datos.

10. La relación entre temperaturas Celsius y Fahrenheit está dada por: C = 5/9 * (F-32) Escribir un algoritmo que haga una tabla de valores Celsius-Fahrenheit, para valores entre OF y 200ºF, con intervalos de 10º.

11. Contar la cantidad de letras de un telegrama que termina en punto sin utilizar funciones de string, salvo la que indica la longitud

12. Contar la cantidad de palabras, separadas por uno o más espacios, de un telegrama que termina en punto. No utilizar funciones de string salvo la que indica la longitud.

13. Dado un texto terminado en punto, determinar cuál es la vocal que aparece con mayor frecuencia.

14. Dado un texto terminado en "." se pide determinar cuántas veces aparece determinada letra que se indica por teclado. Sin utilizar funciones de string, salvo len.

15. Dado un texto terminado en "." averiguar qué cantidad de letras tiene la palabra más larga.

16. Leer dos letras de teclado y luego un texto terminado en ".". Se pide determinar la cantidad de veces que la primera letra precede a la segunda en el texto. No utilizar funciones de string salvo la que indica la longitud.

17. Leer N y luego N lotes de números reales que terminan con un valor 0, y calcular la media individual de cada lote, junto con la media total de todos los números ingresados. No se deben guardar todos los datos.

18. Escribir un algoritmo que lea un número real cualquiera y lo redondee con dos decimales. Verificar con distintas entradas.

19. Hacer una función que devuelva el máximo común divisor y el mínimo común múltiplo entre dos enteros.

20. Escribir un programa principal, con un menú de opciones, para que invoque a las funciones de los puntos 3, 4 y 19.

- Escribir en el programa del punto anterior comentarios al principio de cada función indicando qué es lo que hace, qué recibe y qué devuelve. Agregar comentarios en el programa principal.
- Verificar que los nombres de las variables y las funciones sean correctos.

21. Hacer el seguimiento en papel del siguiente código, luego correrlo en máquina.

```
def f(x, y):
      x = x + 2
      y = y + 10
      print(x, y)

x = 3
y = 8
print(x, y)
f(x, y)
print(x, y)
f(y, x)
print(x, y)
```

22. Hacer el seguimiento en papel del siguiente código, luego correrlo en máquina.

```
def f(x, y):
      while (x < y):
            x = x + y
      x = x + 2
```

```
        y = y + 10
        print(x, y)

x = 3
y = 8
print(x, y)
f(x, y)
print(x, y)
f(y, x)
print(x, y)
```

Tuplas, listas y strings

1. Desarrollar una función que devuelva en un vector (una lista) los números primos entre 2 y 200. Reutilizar lo que ya se escribió y probó.

2. Dados dos vectores A y B, de N elementos cada uno, se desean calcular:
   a. el vector suma.
   b. el producto escalar.

3. Por cada alumno que rindió un examen de una materia se lee el número de legajo y la nota obtenida. Se desea saber la cantidad de alumnos que rindieron el examen, el porcentaje de alumnos que obtuvieron cada nota, y el (o los) legajos de la nota más alta.

4. Escribir una función que reciba una lista y un valor y devuelva verdadero (True) si el valor está en la lista, falso (False) en caso contrario. Hacerlo sin usar *in* ni *count*.

5. Escribir la misma función del punto anterior usando count.

6. Escribir la misma función del punto anterior usando in.

7. Escribir una función que reciba una lista y un valor y devuelva la posición en que encuentra al valor en la lista, si el valor estuviera repetido devolver la primera aparición, si no estuviera devolver –1. Escribirla sin utilizar funciones como *in*, *count*, *index*, etc.

8. Escribir la misma función del punto anterior usando funciones específicas de Python.

9. Se lee un vector X de N elementos (enteros). Escribir un algoritmo que devuelva un vector que tenga todos los elementos de X, pero sin elementos repetidos.

10. Se leen dos vectores A y B, de N y M elementos respectivamente. Construir un algoritmo que halle los vectores unión e intersección de A y B. Desarrollarlo sin usar conjuntos (set) en Python.

11. Si los números de un vector representan los coeficientes de un polinomio (de grado no mayor a 10), escribir un algoritmo que calcule la especialización de ese polinomio con un número que elige el usuario.

12. Escribir un algoritmo que halle una matriz C como suma de dos matrices A y B. La dimensión de las matrices de M × N se lee como dato (suponer un MAX para fila y columna).

13. Escribir un algoritmo que halle un vector cuyos elementos son la suma de los elementos de cada fila de una matriz previamente ingresada.

14. Escribir un programa que calcule la traza de una matriz cuadrada. Recordar que la traza de una matriz es la suma de los elementos de su diagonal principal.

15. Escribir un algoritmo que determine si una matriz cuadrada ingresada es la matriz identidad. Optimizar el código.

16. Escribir un algoritmo que construya un vector con los valores mínimos de cada una de las filas de una matriz.

17. Definir una función que dada una fecha en formato DD/MM/AA, verifique si es correcta o errónea. Ejemplo: El 31/02/18 es una fecha errónea. Investigar cómo se calcula si un año es bisiesto o no.

18. Escribir una función que, dada una palabra, determine si es capicúa o no.

19. Escribir una función que ordene alfabéticamente una lista de N nombres. Escribirlo sin utilizar *sort* ni *sorted*.

20. Hacer el mismo punto anterior usando la función *sorted*.

21. Hacer el mismo punto anterior usando el método *sort*.

22. Escribir una función que, dado dos vectores (uno contiene nombres y el otro número de legajos), ordene el primero y el segundo en paralelo. ¿Se puede utilizar *sort* o *sorted* en este ejercicio?

23. Escribir una función que detecte, en un string, una operación que se desee realizar, ejemplo: “59 + 30” y devuelva el valor o un mensaje que no se puede realizar la operación. Las operaciones pueden ser +, – , x o /. Los valores deben estar separados por un blanco.

Registros y Tablas (Listas y Diccionarios)

1. El Servicio Meteorológico Nacional tiene registrado, mes por mes, la cantidad de lluvia caída por provincia, en mm (entero) y el promedio de humedad relativa (real).
   Estos datos los guarda en una matriz de 24 (23 provincias más CABA) x 12 (meses).
   Se pide, hacer una aplicación en Python que realice lo siguiente:
   a. Indicar en qué mes llovió más y la cantidad.
   b. Indicar si la provincia donde más llovió es la que tiene mayor humedad relativa.
   c. Hacer un listado de las 10 provincias donde más llovió, ordenado de mayor a menor por cantidad de agua caída. Indicando: nombre de provincia y cantidad de agua caída
   Nota: simular la carga de forma aleatoria (utilizando la función random).

2. Un artículo se comercializa en 5 sucursales. Se leen N (N lo carga el usuario) datos, con el siguiente formato:
   - sucursal: Número de sucursal (de 1 a 5)
   - mes: Número de mes (de 1 a 12)
   - cantidad: Cantidad del producto vendida.
   - monto_total: Monto recaudado (los precios del producto difieren por cada sucursal y por cada mes).

   Se pide hacer un programa en Python que:
   a. Vuelque los datos en una matriz de 5 (sucursales) x 12 (meses). Cada celda contendrá un registro con la información cantidad – monto_total.
   b. A partir de la matriz armar un listado, ordenado de mayor a menor por monto, indicando sucursal, mes y cantidad vendida.
   c. Indicar si la mayor cantidad vendida corresponde al precio unitario más bajo.
   Nota: generar los datos de forma aleatoria.

3. Escribir un programa que recorra una lista de números, que puede ser cargada por teclado o en forma aleatoria, calculando
   a. Para cada uno el cuadrado (x ** 2) y genere otra lista con dichos números.
   b. Unir las 2 listas y ordenarlas por valor. Imprimir la misma sin duplicados.

4. Armar una lista con las primeras 7 letras del alfabeto, pasar c d y e a mayúsculas. Borrar las ultimas 2 letras, e imprimir el resultado y la cantidad de elementos de la misma.

5. Imprimir los elementos de una lista con su posición:

a. Sin usar la función *enumerate*.
b. Usando la función *enumerate*.

6. Procesar una lista de números enteros, e imprimir para cada uno de ellos: el número que está procesando y la suma parcial de los mismos. Además, indicar True si el número es mayor al número anterior y False de lo contrario. Terminar cuando se hayan procesado todos los números o cuando la suma parcial haya alcanzado un valor mayor o igual a 100.

7. Suponiendo que cuenta con una lista en la que en cada posición tiene la información de un alumno en un registro:

   [{'nombre': 'XX', 'legajo': 1, 'nota': 4, 'grupo': 1}, ...]

   - Se desea imprimir el nombre y legajo de todos los alumnos aprobados.
   - Asumiendo que la lista se encuentra ordenada por número de grupo, se pide indicar aquellos grupos para los cuales todos sus integrantes hayan aprobado el parcial recorriendo sólo una vez la lista.

8. Se cuenta con dos listas de números ordenadas de forma creciente y se desea obtener una nueva lista ordenada que contenga todos los números, pero sin ordenarla nuevamente.

9. Procesar una lista de números y generar un diccionario con dos claves llamadas "par" e "impar". Al terminar de procesar la lista el diccionario debe tener todos los números que procesó agrupados en pares e impares.
   Por ejemplo, si contamos con la lista [1, 5, 2, 6, 9, 3, 8], el diccionario que se obtenga debería ser: {"par": [2, 6, 8], "impar": [1, 5, 9, 3]}

10. Procesar una lista de strings e ir guardando en un diccionario la cantidad de ocurrencias de cada palabra (distinguir mayúsculas y minúsculas). Por ejemplo, parea la lista:
    ['Otra', 'posible', 'clasificacion', 'radica', 'en', 'si', 'una', 'variable', 'puede', 'cambiar', 'el', 'tipo', 'de', 'dato', 'que', 'se', 'puede', 'almacenar', 'en', 'ella', 'entre', 'una', 'sentencia', 'y', 'la', 'siguiente', '(', 'tipado', 'dinamico', ')', 'O', 'si', 'en', 'la', 'etapa', 'de', 'definicion', 'se', 'le', 'asigna', 'un', 'tipo', 'de', 'dato', 'a', 'una', 'variable', 'y', 'por', 'mas', 'que', 'se', 'puede', 'cambiar', 'el', 'contenido', 'de', 'la', 'misma', 'no', 'cambie', 'el', 'tipo', 'de', 'dato', 'de', 'lo', 'que', 'se', 'almacena', 'en', 'ella', '(', 'tipado', 'estatico', ')']

    El resultado sería:
    {'el': 3, 'en': 4, 'etapa': 1, 'por': 1, 'Otra': 1, 'contenido': 1, 'almacenar': 1, 'sentencia': 1, 'le': 1, 'tipo': 3…]

11. Se tienen dos diccionarios, uno de precios de productos y otro de stock de productos. Ambos tienen las mismas claves. Se pide
    a. acceder a los dos diccionarios simultáneamente calculando la multiplicación del precio por la cantidad de artículos para cada producto, e imprimir los casos en que dicho monto supere los $100.000.
    b. Calcular el monto total para el total del stock.

12. Se pide que ingresen por teclado pares de equipo – puntos ganados, el mismo par se puede ingresar varias veces. Se pide
    a. generar una tabla de puntos acumulados para cada equipo.
    b. Mostrar la tabla de posiciones (equipo – total de puntos) ordenada por el total de puntos en forma descendente.

13. Dada una lista de empleados, donde cada elemento es, a su vez, una lista que contiene: legajo, nombre, sexo, sueldo. Nota: un empleado puede tener más de un registro ya que puede cobrar adicionales. Se pide:
    a. Armar un diccionario con clave legajo y el resto de los datos como valor, unificando los datos: se deben sumar los sueldos para un mismo empleado.
    b. Listar legajo – nombre – sueldo, ordenado por legajo.
    c. Listar legajo – nombre – sueldo, ordenado por sueldo, de mayor a menor.
    d. Indicar el mayor sueldo y a quién corresponde.
    e. Indicar si el promedio de sueldos de las mujeres es menor que el promedio de sueldos de los hombres.

Ejercicios tipo parcial

1. Solicitar al usuario el ingreso de un texto. A continuación:
    a. Mostrar el texto, pero ordenado por palabra y todo en mayúsculas.
    b. Informar cuantos caracteres tiene la palabra más larga.
    c. Generar una nueva lista sin palabras repetidas.
    d. Armar un ranking de palabras, informando palabra y cantidad de ocurrencias, ordenado por la cantidad de ocurrencias.

2. Escribir una función que, dado un texto que se pasa por parámetro, lo imprima al revés y sin ninguna vocal. Las vocales pueden estar en minúsculas o mayúsculas.

3. Solicitar al usuario el ingreso de un texto. El mismo debe contener al menos 100 palabras, de lo contrario deberá exigirle que ingrese más palabras y adicionarlas a las ya ingresadas hasta superar el mínimo establecido. Considere que el usuario solo ingresa palabras separadas por blancos sin ningún otro tipo de caracteres. Muestre una lista de las palabras ingresadas, ordenada alfabéticamente, sin repetir palabras.

4. Ingresar, en un diccionario, pares de datos con una clave que será el nombre de un partido político y un valor que será la cantidad de votos obtenidos en una provincia, una misma clave se puede ingresar varias veces. Se pide:
    a. Calcular el total de votos para cada partido e imprimirlo sin ningún orden.
    b. Imprimir el listado anterior ordenado de mayor a menor por cantidad de votos.

5. Escribir una función que, dado un texto que se pasa por parámetro, lo devuelva cambiando la letra “m” por “eme”, y “M” por “EME”.

6. Solicitar al usuario el ingreso de un texto. Considerar que el usuario solo ingresa palabras separadas por blancos, sin ningún otro tipo de caracteres. Mostrar una lista de las palabras capicúas ingresadas, ordenadas alfabéticamente, sin repetirlas. Una palabra capicúa es la que es exactamente igual al invertirla.

7. Ingresar en un diccionario pares de datos con una clave que será el nombre de una ONG (Organización No Gubernamental) y el monto que será una donación. Una misma clave se puede ingresar varias veces. Se pide:
    a. Calcular el total recaudado para cada ONG e imprimirlo sin importar un orden.
    b. Imprimir el listado anterior ordenado de mayor a menor por cantidad recaudado, indicando: cantidad – ONG

Recursividad

1. Hacer una función recursiva que devuelva la sumatoria desde 1 hasta un número n que se pase por parámetro. Probarla con un pequeño programa. **(+)**

2. Hacer una función recursiva que devuelva la suma de los elementos de una lista que se pasa por parámetro.

3. Hacer una función recursiva que devuelva el valor de la sucesión de Fibonacci de un entero n. Nota: la serie de 0, devuelve 0, y la de 1 devuelve 1. Luego, devuelve la suma de los dos términos anteriores. Luego, hacerla de manera iterativa (mediante un ciclo). Comparar el rendimiento (tomar los tiempos) con la Serie en 5, 15, 30 y 60.

4. Hacer una función recursiva que invierta un string que se pase por parámetro, y lo devuelva. Por ejemplo: CASA – ASAC. Probarla con un pequeño programa.

Archivos

1. Dado un archivo “prueba.txt” de formato CSV (valores separados por coma), en donde hay 3 campos: número de legajo, nombre y apellido, nota (de tipo entero), se pide: leerlo e imprimirlo por pantalla con el mismo formato que está en el archivo.
   Nota: escribir un archivo para probar el programa.

2. Dado el mismo archivo del punto anterior, leerlo por completo e ir generando, a medida que se lo lee, un nuevo archivo “prueba2.txt” con el mismo formato, pero, en lugar de nota irá: R (recursa) si la nota es menor a 4, A (aprobado) si está entre 4 y 7, D (distinguido) si es 8 o 9, y S (sobresaliente) si es 10.

3. Dado un archivo MAESTRO.txt y uno NOVEDAD.txt con el formato que se encuentra en el video de ejemplo:
   MAESTRO
   Legajo, nombre, sueldo
   NOVEDAD
   Legajo, nombre, sueldo, tipo de novedad
   Ambos ordenados por número de legajo de menor a mayor.
   Se pide hacer un apareo, teniendo en cuenta que, la cantidad de campos del archivo NOVEDAD puede no ser correcta (más o menos campos), en estos casos, la línea irá al archivo de ERRORES.

4. Dados dos archivos de ventas de SUCURSAL_1.txt y SUCURSAL_2.txt como los del ejemplo del video:
   Fecha (año-mes-dia),Sucursal (1 o 2),monto
   Ambos ordenados por fecha de menor a mayor.
   Se pide hacer un merge, teniendo en cuenta que la cantidad de campos (en cualquiera de los archivos) puede no ser la correcta (más o menos campos). En ese caso enviar esa línea al archivo ERRORES.txt

5. Repetir el ejercicio anterior pero:
   a. Con 3 archivos de sucursales.
   b. Con 4 archivos de sucursales.
   c. Con n archivos de sucursales (n es menor a 100), para esto hacer una función que haga un merge entre dos archivos y se llame n – 1 veces. Los archivos de las sucursales se llamarán: suc01.txt, suc02.txt, suc03.txt, etc.

6. Rehacer el ejercicio 1 pero con un archivo binario: “pruebas.dat”

7. Rehacer el ejercicio 2 pero con archivos binarios: “pruebas2.dat”

8. Rehacer el ejercicio 3 pero con archivos binarios: MAESTRO.DAT y NOVEDAD.DAT.

9. Rehacer el ejercicio 4 pero con archivos binarios SUCURSAL_1.DAT y SUCURSAL_2.DAT.

Anexo

Pregunta

Revisá el código y respondé

- ¿Los nombres de las variables son correctos?
- ¿Dejé los 4 espacios de sangría en los bloques internos?

Volver a la guía

Pensá

¿Son correctos los nombres de las variables? Una persona que desconozca el enunciado, ¿se dará cuenta qué significa cada una de las variables?

Si la respuesta a la pregunta es NO, reescribí el algoritmo cambiando los nombres de las variables para que cualquier persona que lo lea, lo pueda entender.

Volver

Verificar

- Probá el ejercicio anterior con números positivos y negativos.
- Sólo con negativos.
- Con un solo número.
- Con dos números distintos.
- Con dos números iguales.
- Con el máximo en la primera posición y el mínimo en la última.
- Con el máximo en la última posición y el mínimo en la primera.
- Con el máximo y el mínimo en cualquier posición, salvo la primera y la última.
- Verificá que funcione correctamente en todos los casos.
- Al finalizar graficá un diagrama de flujo.

Volver

Investigar

- ¿De qué otra forma se podría haber resuelto el ejercicio anterior? Buscá en Internet otros algoritmos que pueden ser de utilidad. Elegí alguno e implementalo como ejercicio 8 – v2.

- A ambos ejercicios agregá un contador que se irá incrementando, cada vez que hace algún cálculo o comparación. Al final imprimí esos contadores y compará el resultado, ingresando un mismo número en ambos algoritmos.

- Probá los ejercicios con números grandes.

Volver

Compará

- El ejercicio anterior (ejercicio 5) probalo con dos funciones distintas: una basada en la primera versión del ejercicio 8 y, la otra, con la segunda. Dejá los contadores que te decían cuántas operaciones hacía el algoritmo.

- Probá ambos algoritmos con el número 100 y compará: los contadores y el tiempo que tardó en ejecutarse uno y otro. Medilo con un cronómetro.

Acá hay uno: http://cronometro-online.chronme.com/

- Ahora hacé lo mismo con el número 1000.

- Ahora con 10000. ¿Qué notás?

- Ahora con 100000. ¿Qué pasó?

- ¿Alguna conclusión?

Volver