

Algoritmos y Programación III

Práctica N°0 – Introducción, repaso y recursividad

Notas preliminares

- **Fecha de Entrega:** viernes 23 de marzo.
- No se aceptarán soluciones que no compilen.
- Los ejercicios marcados con ★ son de entrega **obligatoria**.
- Las entregas son **en grupos de hasta dos personas**.
- La entrega de los ejercicios se hará en papel (imprimiendo los .java que se entregan) y por mail a **ayp3.unpaz@gmail.com**, en un archivo .zip con los archivos java correspondientes.

1. Variables, expresiones y tipos

Ejercicio 1

Escribir el programa “¡Hola, mundo!”.

Ejercicio 2

Escribir un programa que te pregunte tu nombre y a continuación imprima un saludo del estilo “Hola [nombre]”. Para leer un texto por teclado, utilizar el método `Console.pedirTexto()` de la librería de la materia.

Ejercicio 3

Escribir un programa que te pregunte por dos números, y a continuación imprima un mensaje del estilo “La suma es: ” y el valor de la suma de ambos números. Para leer un entero de pantalla, utilizar el método `Console.pedirEntero()` de la librería de la materia.

Ejercicio 4

¿Qué valor tienen las siguientes expresiones?

- $1/2$
- $1.0/2.0$
- $1.0/2$
- $1/2.0$
- $"1"/"2"$
- $1+2$
- $"1"+"2"$
- $16/2*4$
- $16/(2*4)$

Ejercicio 5

Escribir un programa que te pregunte por dos números, y a continuación imprima un mensaje del estilo “El promedio es: ” y el valor del promedio de ambos números.

2. Métodos y condicionales

Ejercicio 6

Escribir un método **static void imprimirSuma(int a, int b)** que al igual que el ejercicio 4 imprima la suma de los dos parámetros. Modificar el programa de dicho ejercicio para que utilice este método.

Ejercicio 7

Análogamente al ejercicio anterior, escribir un método **static void imprimirPromedio(int a, int b)** que imprima el promedio de los dos parámetros.

Ejercicio 8

Escribir un método **static void ponerNota(double x, double y)** que toma dos números decimales y los promedia. En caso que el promedio sea mayor o igual a 7, deberá imprimir ‘‘Promocionado’’, si es mayor o igual a 4 pero menor que 7, imprime ‘‘Aprobado’’ y si es menor que 4 imprime ‘‘Debe recuperar’’. Probarla llamándola desde el main con distintos números. Luego, pedirle ámbos números al usuario con el método `Console.pedirDecimal()` para pasárselos a `ponerNota`.

Ejercicio 9

Escribir un método **static void imprimirFecha(int dia, int mes, int anio)** que imprime la fecha pasada como parámetro en formato del estilo ‘‘5 de Julio de 2030’’.

3. Métodos con resultados y recursividad

Ejercicio 10

- a) Escribir un programa que te pregunte por un número e imprima un mensaje de tipo ‘‘[x] es par’’ si el número es par (reemplazando [x] por el número en cuestión). En caso contrario, debe imprimir ‘‘[x] es impar’’.
- b) Crear un método **static boolean esPar(int n)** que devuelve **true** si *n* es par, y **false** en caso contrario.
- c) Modificar el **main** del programa para que utilice este método.

Ejercicio 11

Escribir un método **static int cantCifras(int n)** que devuelve la cantidad de cifras de *n*. Probarlo adecuadamente llamándola desde el main.

Ejercicio 12

Escribir un método **static boolean esDivisible(int n, int m)** que devuelve **true** si *n* es divisible por *m* y **false** en caso contrario. Probarlo adecuadamente llamándola desde el main.

Ejercicio 13

Escribir un método **static int sumatoria(int n)** que devuelve la sumatoria de los números desde 1 hasta n .

Ejercicio 14

Escribir un método **static int sumatoriaPares(int n)** que devuelve la sumatoria de los números pares desde 2 hasta n . Nota: se puede usar el método del ejercicio 10.

Ejercicio 15

Escribir un método **static double potencia(double x, int a)** que toma un número racional x y un entero a y calcula x^a .

Ejercicio 16

Escribir un método **static double factorial(int n)** que toma un entero positivo n y calcula $n!$ (el factorial de n) que se define como el producto de todos los naturales desde 1 hasta n . Por ejemplo $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$. Ojo: por definición, el factorial de 0, es 1 ($0! = 1$).

Ejercicio 17 ★

Si no lo hizo de esa manera en los ejercicios anteriores, escribir las versiones recursivas de los siguientes métodos:

- a) **sumatoria: static int recurSumatoria(int n)**
- b) **sumatoriaPares: static int recurSumatoriaPares(int n)**
- c) **potencia: static int recurPotencia(int n)**
- d) **factorial: static int recurFactorial(int n)**

4. Iteración y Cadenas

Ejercicio 18 ★

Escribir las versiones iterativas de los siguientes métodos:

- a) **sumatoria: static int iterSumatoria(int n)**
- b) **sumatoriaPares: static int iterSumatoriaPares(int n)**
- c) **potencia: static int iterPotencia(int n)**
- d) **factorial: static int iterFactorial(int n)**

Ejercicio 19

- a) Escribir un programa que pida por pantalla un texto y a continuación lo imprima de atrás para adelante. Para obtener las letras de una cadena de caracteres pueden utilizar el método **charAt** de **String**. Por ejemplo, **cadena.charAt(0)** devuelve el primer carácter del **String** **cadena**.
- b) Mover el código que imprime la cadena al revés a un método **static void imprimirReversa(String cadena)**
- c) Escribir un método **static String reversa(String cadena)** que dado un **String**, devuelve otro **String** con los caracteres invertidos. Por ejemplo, **reversa("hola")** debería devolver el **String** "aloh".
- d) Modificar el método **imprimirReversa** para que utilice el método definido en el punto anterior.

Ejercicio 20

Escribir un método **static int cantidadApariciones(String s, char c)** que dada una cadena y un caracter, cuenta la cantidad de veces que aparece **c** en **s**.

Ejercicio 21 ★

Escribir un método **static int cantidadVocales(String s)** que dada una cadena que contiene sólo letras minúsculas sin acentuar, devuelve la cantidad de vocales en dicha cadena. Nota: se puede utilizar el método definida en el ejercicio anterior.

Ejercicio 22

Una palabra se dice que es “abecedaria” si las letras en la palabra aparecen en orden alfabético. Por ejemplo, las siguientes son todas palabras abecedarias del idioma castellano.

adiós, afín, afinó, ágil, bello, celos, cenó, chinos dijo, dimos, dios, fijos, finos, hijos, hilos, himno

1. Describí un algoritmo para decidir si una palabra dada es abecedaria, asumiendo que la misma contiene sólo letras minúsculas.
2. Implementar el algoritmo en un método **static boolean esAbecedaria(String s)**.
3. ¿Funciona el algoritmo si le pasamos como parámetro “ágil”? En caso negativo, ¿por qué te parece que puede ser? ¿Cómo lo solucionarías?

Ejercicio 23 ★

Escribir el método **static boolean esCapicua(String s)** que dada una cadena, devuelve **true** si la cadena es igual de atrás hacia adelante o de adelante hacia atrás. En caso contrario, devuelve **false**.

Ejercicio 24 ★

Escribir un método **static boolean esSinRepetidos(String s)** que dada una cadena, devuelve **true** si no hay letras repetidas en la cadena. En caso contrario, devuelve **false**. No utilizar el método del ejercicio [25](#).

Ejercicio 25

Escribir un método **static String sinRepetidos(String s)** que dada una cadena, devuelve una nueva cadena donde cada uno de los caracteres que aparecían en **s**, aparecen sólo una vez. Se debe mantener la posición relativa de los caracteres: para aquellos que se encuentren repetidos puede conservarse cualquiera de sus apariciones. Por ejemplo, para la palabra “**casos**” puede devolver “**caso**” o “**caos**”, conservando la primera o la segunda letra **s** respectivamente.

Ejercicio 26 ★

La sucesión de Fibonacci es una sucesión de números naturales que describe, por ejemplo, el número de individuos en una población de conejos tras varias generaciones. Esta sucesión tiene la particularidad de estar presente en muchos elementos de la naturaleza, y que a medida que

se aproxima al infinito, el cociente entre dos elementos consecutivos, se aproxima a la proporción áurea. Los números de la sucesión se obtienen de la siguiente manera:

$$\begin{aligned}f_0 &= 0 \\f_1 &= 1 \\f_n &= f_{n-2} + f_{n-1}\end{aligned}$$

Los primeros números de la sucesión serían entonces: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Implementar el método que devuelve el n -ésimo elemento de la sucesión:

- a) usando recursividad, con la siguiente signature: **static int fibrec(int n)**
- b) usando un ciclo, con la siguiente signature: **static int fibiter(int n)**

Responder:

- a) Comparar los tiempos entre una implementación y otra. ¿Cuál es el término más grande que puede calcular cada una de las implementaciones en un tiempo menor a 10 segundos? ¿Hay diferencia? ¿Te imaginás por qué?
- b) ¿Cuánto vale f_{47} ? Interpretá este resultado.

Ejercicio 27 ★

La sucesión de Collatz se define de la siguiente manera. Se comienza del número n y se prosigue así:

- Si n es par, entonces el siguiente número es $n/2$
- Si n es impar, entonces el siguiente número es $3 * n + 1$
- Cuando n vale 1, no hay siguiente número.

Escribir un método **static void collatz(int n)** que toma un natural n e imprime, en líneas separadas, los números de la sucesión.