



### 1. Entrada/Salida + Pasaje de parámetros

**Ejercicio 1.** Escribir un programa en el que se ingrese un número por teclado (entrada estándar), calcule si es primo y muestre por pantalla (salida estándar) “El número ingresado es primo” si es primo. En caso contrario: “El número ingresado no es primo”

**Ejercicio 2.** Escribir una función `escribirArchivo` que escriba en un archivo llamado `salida.txt` dos enteros `a` y `b` y luego dos reales `f` y `g` separados con coma en una única línea.

**Ejercicio 3.** Leer del archivo `entrada.txt` un valor entero y almacenarlo en una variable llamada `a` y luego leer un valor real y almacenarlo en una variable llamada `f`. Mostrar los valores leídos en la salida estándar. Ambos valores están separados por un espacio y hay una única línea en el archivo (por ejemplo: “-234 1.7”)

**Ejercicio 4.** `numeros.txt` contiene una lista de números separados por espacios. Leerlos e imprimirlos por pantalla.

**Ejercicio 5.** ¿Cuál es el valor de `a` luego de la invocación `prueba(a,a)`?

```
int a = 10;
void prueba(int& x, int& y) {
    x = x + y;
    y = x - y;
    x = 1/y;
}
prueba(a, a);
```

En los siguientes ejercicios, ingresar los valores por entrada estándar, mostrar en la salida estándar los valores ingresados y los resultados de las funciones.

**Ejercicio 6.** Implementar la función `swap`: `void swap(int& a, int& b)`, que cumpla con la siguiente especificación:

```
proc swap (inout a:Z, inout b:Z) {
    Pre {a = a0 ∧ b = b0}
    Post {a = b0 ∧ b = a0}
}
```

**Ejercicio 7.** `void collatz(int n, int& cantPasos)`

La conjetura de *Collatz* dice que dado un número natural  $n$  y el proceso que describimos a continuación, sin importar cuál sea el número original, provocará que la serie siempre termine en 1. El proceso:

- Si  $n$  es par lo dividimos por 2
- Si  $n$  es impar lo multiplicamos por 3 y le sumamos 1 al resultado

En este ejercicio, supondremos que la conjetura es cierta y se pide implementar una función que devuelva la cantidad de pasos que se realizan desde el número original hasta llegar a 1. Además debe guardar en un archivo la sucesión de números por la que pasa. Ejemplo: si calculamos `collatz` de 11, la cantidad de pasos es 15 y la sucesión es 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

**Ejercicio 8.** Dados dos archivos que contienen números separados por espacios (ambos archivos tienen la misma cantidad de números), se pide que se sumen los valores de los archivos y se genere uno nuevo con la suma de los mismos. Ejemplo: “`numeros.txt`” contiene 1 25 6 y “`numeros1.txt`” contiene 45 5 4 debe crear el archivo “`suma.txt`” que contenga 46 30 10.

**Ejercicio 9.** `void primosGemelos(int n, int& res1, int& res2)` Decimos que  $a$  y  $b$  son primos gemelos, si ambos son primos y además  $a=b-2$ . Queremos obtener los  $i$ -ésimos primos gemelos. Por ejemplo, son primos gemelos 3 y 5, 5 y 7, 11 y 13, 17 y 19, 29 y 31, 41 y 43 ... , los 4-ésimos primos gemelos son 17 y 19. Además se debe escribir en un archivo la secuencia de primos gemelos hasta llegar al  $i$ -ésimo. Para el ejemplo el archivo debe contener: (3,5) (5,7) (11,13) (17,19)