

1. Introducción a C++

Ejercicio 1. Crear un archivo: “labo00.cpp” (con cualquier editor de texto) y escribir lo siguiente:

```
#include <iostream>

int f(int x){
    return x+1;
}

int main() {
    std::cout << "El resultado es: " << f(10) << std::endl;
    return 0;
}
```

Luego, compilar y ejecutar el código en la terminal:

```
g++ labo00.cpp -o labo00_ejecutable
./labo00_ejecutable
```

Ejercicio 2. Modificar el programa anterior para que f tome dos parámetros de tipo `int` y los sume.

Ejercicio 3. Modificar el programa anterior para que f tome dos parámetros x e y de tipo `int` y los sume sólo si $x > y$, en caso contrario el resultado será el producto.

Ejercicio 4. Crear un proyecto nuevo de C++ en **CLion** con el nombre labo00. Escribir el programa del ejercicio anterior y ejecutarlo.

Ejercicio 5. Escribir la función que dado $n \in \mathbb{N}$ devuelve si es primo. Recuerden que un número es primo si los únicos divisores que tiene son 1 y el mismo.

Iteración vs Recursión

Los siguientes ejercicios deben ser implementados primero en su versión **recursiva**, luego iterativa utilizando **while** y por último iterativa utilizando **for**.

Ejercicio 6. Escribir la función de Fibonacci que dado un entero n devuelve el n -ésimo número de Fibonacci. Los números de Fibonacci empiezan con $F_0 = 0$ y $F_1 = 1$. $F_n = F_{n-1} + F_{n-2}$

Ejercicio 7. Escribir la función que dado $n \in \mathbb{N}$ devuelve la suma de todos los números impares menores que n .

Ejercicio 8. Escribir la función `sumaDivisores` que dado $n \in \mathbb{N}$, devuelve la suma de todos sus divisores entre $[1, n]$.

■ **Hint:** Recordar que para la versión recursiva es necesario implementar `divisoresHasta`

Ejercicio 9. Escribir una función que dados $n, k \in \mathbb{N}$ compute el combinatorio: $\binom{n}{k}$. Hacerlo usando la igualdad $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$
¿Qué pasa si tuvieran que escribir la versión iterativa?

Ejercicio 10. ¿Es mejor programar utilizando algoritmos recursivos ó iterativos? ¿Es mejor usar **while** o **for**?

2. Entrada/Salida + Pasaje de parámetros

Ejercicio 11. 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 12. Escribir una función `escribirArchivo` que escriba en un archivo `salida.txt` dos enteros `a` y `b` y luego dos reales `f` y `g` separados con coma en una única línea.

Ejercicio 13. 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 `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 14. `numeros.txt` contiene una lista de números separados por espacios. Leerlos e imprimirlos por pantalla.

Ejercicio 15. ¿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 16. 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 17. Implementar la función `division` que cumpla con la siguiente especificación:

```
proc division (in dividendo Z, in divisor Z, out cociente:Z, out resto:Z) {
    Pre {dividendo ≥ 0 ∧ divisor > 0}
    Post {dividendo = divisor * cociente + resto ∧ 0 ≤ resto < divisor}
}
```

Resolver este ejercicio en versiones iterativa y recursiva.

Ejercicio 18. `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 y que muestre en la salida estándar 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

Resolver este ejercicio en versiones iterativa y recursiva.

Ejercicio 19. 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 20. `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)

3. Vectores

Importar el proyecto de la clase. En CLion seleccionar File → Import Project y seleccionar la carpeta con el nombre Labo02-Vectores que se encuentra en el archivo Laboratorio 02.zip

Ejercicio 21. Especificar y luego implementar en su versión **recursiva** e **iterativa**:

1. `bool divide(vector<int> v, int n)`
Dados un vector v y un entero n , decide si n divide a todos los números de v .
2. `int maximo(vector<int> v)`
Dado un vector, devuelve el valor máximo.
3. `bool pertenece(int elem, vector<int> v)`
Dado un entero, indica si pertenece o no al vector.

Ejercicio 22. Implementar en su versión **iterativa**:

1. `void mostrarVector(vector<int> v)`
Dado un vector de enteros muestra por la salida estándar (cout), el vector
Ejemplo: si el vector es $\langle 1, 2, 5, 65 \rangle$ se debe mostrar en pantalla $[1, 2, 5, 65]$
2. `vector<int> limpiarDuplicados(vector<int> v)`
Dado un vector de enteros, devuelve un vector de enteros con los elementos del vector sin duplicados. Ejemplo $v = \langle 1, 1, 2, 1, 1, 2, 3, 2, 3, 3 \rangle$ el resultado es $\langle 1, 2, 3 \rangle$
3. `vector<int> rotar(vector<int> v, int k)`
Dado un vector v y un entero k , rotar k posiciones los elementos de v .
 $[1, 2, 3, 4, 5, 6]$ rotado 2, debería dar $[3, 4, 5, 6, 1, 2]$.
4. `vector<int> reverso(vector<int> v)`
Dado un vector v , devuelve el reverso. Implementar también la versión recursiva de este problema.
5. `vector<int> factoresPrimos(int n)`
Dado un entero devuelve un vector con los factores primos del mismo. Los factores primos de un número entero son los números primos divisores exactos de ese número entero. Ejemplos: los factores primos de 6 son 2 y 3. Factores primos de 7 es 7
6. `bool estaOrdenado(vector<int> v)`
Dado un vector v de `int`, dice si es monótonamente creciente o monótonamente decreciente.
7. `void negadorDeBooleanos(vector<bool>& booleanos)`
Modifica un vector de booleanos negando todos sus elementos.
8. `void sinImpares(vector<int>& v)`
Dado un vector de enteros, devuelve el mismo vector colocando un 0 en las posiciones en las que haya un número impar.
9. `vector<pair<int, int> > cantidadCaracteres(vector<int> v)`
Dado un vector de enteros, devuelve una tupla que contine por cada entero la cantidad de apariciones del mismo.
Ejemplo $v = \langle 1, 1, 2, 1, 1, 2, 3, 2, 3, 3 \rangle$ el resultado es $\langle (1, 4), (2, 3), (3, 3) \rangle$
Tip: para aprender a usar las tuplas entrar a <http://www.cplusplus.com/> y buscar (en donde dice search) `pair`.

Ejercicio 23. *Integradores.* Implementar las siguientes funciones

1. `void palindromos(string rutaArchivoIn, string rutaArchivoOut)`
Este procedimiento debe leer un archivo que contiene una lista de strings y crear uno nuevo dejando sólo los palíndromos. Además, debe transformar las palabras a mayúscula. **Ayuda:** Buscar la función `toupper` definida en `cctype`. Utilizar como ejemplo el archivo `palindromos.txt`.
2. `void promedios(string rutaArchivoIn1, string rutaArchivoIn2, string rutaArchivoOut)`
Dados dos archivos en los que cada uno contiene una secuencia de enteros (ambas con la misma longitud), guardar el promedio de cada par de números que se encuentran en la misma posición en el archivo de salida. Por ejemplo: si tenemos dos secuencias "1 2 3 4" y "1 25 3 12" el resultado debe ser "1 13.5 3 8"

3. `void promediosHasta(string rutaArchivoIn, string rutaArchivoOut)`

Dado un archivo `rutaArchivoIn`, que contiene una lista de números separados por espacios, calcular una lista de promedios de la misma longitud donde cada elemento corresponde al promedio de todos los anteriores de la lista original (incluyendo dicho elemento). La lista de promedios debe ser guardada en el archivo indicado por `rutaArchivoOut`. Por ejemplo, dada la secuencia “1 2 3 2 1”, el resultado debe ser “1 1.5 2 2 1.8”.

4. `void cantidadApariciones(string rutaArchivoIn, string rutaArchivoOut)`

Dado un archivo `rutaArchivoIn`, que contiene una lista de números separados por espacios, contar la cantidad de apariciones de cada uno y escribe `rutaArchivoOut` con una línea por cada número encontrado, un espacio y la cantidad de apariciones. Por ejemplo: si el “1” aparece 44 veces y el “2” 20 veces, la salida debería contener dos líneas: “1 44” y “2 20”.

5. `int cantidadAparicionesDePalabra(string rutaArchivo, string palabra)`

Dada una palabra y un archivo de texto devuelve la cantidad de apariciones de la palabra en el archivo.

6. `void estadisticas(string rutaArchivo)`

Dado un archivo de texto, mostrar por pantalla las estadísticas de cantidad de palabras con longitud 1, 2, 3... hasta el máximo. Por ejemplo:

```
Palabras de longitud 1: 100
Palabras de longitud 2: 12
Palabras de longitud 3: 6
...
```

7. `void interseccion()`

Procedimiento que pide al usuario que se ingresen dos nombres de archivos que contengan sólo números enteros separados por espacios, luego calcula la intersección (números que se encuentran en ambos archivos) e imprime por pantalla el resultado.