



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 `writeToFile` que escriba en un archivo `salida.txt` 2 enteros `a` y `b` y luego 2 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. 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 “`salida.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) (11,17) (17,19)

3. Vectores

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.

4. L^AT_EX

1. Utilizando L^AT_EX, crear un documento que sea reproduzca el contenido del documento `intro_latex.pdf`, respetando la estructura de secciones y sub-secciones.
2. Completar la subsección Especificación reemplazando el `TODO` con el siguiente contenido:
Nota: Utilizar las macros de algo I. Las macros deben incluirlas despues de `\documentclass` y antes de `\begin{document}` de la siguiente manera:
`\input{pathCarpetaMacros/Algo1Macros}`


```
proc factorial (in n:  $\mathbb{Z}$ , out result:  $\mathbb{Z}$ ) {  
    Pre  $\{n \geq 0\}$   
    Post  $\{(n = 0 \rightarrow result = 1) \wedge (n > 0 \rightarrow result = \prod_{k=1}^n k)\}$   
}
```
3. Ahora agreguen la caratula y el índice. Para esto es necesario agregar el paquete `\usepackage{caratula}`
Tomar en cuenta que es necesario tener el archivo `caratula.sty` y las imagenes `logo_dc.jpg` y `logo_uba.jpg`.
Ademas es necesario agregar después de `\begin{document}` el contenido del archivo `header_para_caratula.tex`.
Es necesario cambiar los siguientes datos: titulo, subtítulo, fecha, materia, el nombre y datos de los integrantes del grupo. Notar que pueden agregarse tantos integrantes como sea necesario copiando la línea integrante.

5. Control de Versiones

1. Crear un proyecto nuevo de C++ en CLion.
2. Agregar el archivo `.gitignore` (`gitignore.file` en el zip) al repositorio (`git add .gitignore`)
3. Agregar todos los archivos al repositorio, ignorando aquellos comprendidos por el `.gitignore` (`git add .`)
4. Subir cambios al repositorio
5. Obtener una copia nueva del repositorio. Verificar que CLion pueda abrir y ejecutar el proyecto.
6. Usando `https://git.exactas.uba.ar/rcastano/labo-git-algo1/network/master`, inspeccionar las versiones del archivo `main.cpp` en las 3 branches que aparecen (`master`, `despedida_mejorada` y `que_tal`).
7. Obtener una copia completa del repositorio `https://git.exactas.uba.ar/rcastano/labo-git-algo1` y recordar el directorio en el que está ubicada. (`git clone --mirror https://git.exactas.uba.ar/rcastano/labo-git-algo1`)
Llamaremos `path_repo` al path completo hacia este directorio.
8. Realizar dos copias adicionales del repositorio usando los comandos `git clone path_repo path_copia1` y `git clone path_repo path_copia2`.
9. En el directorio `path_copia1`, integrar en el branch principal (`master`) los cambios del branch `que_tal`.
10. La integración todavía no se verá reflejada en la copia ubicada en `path_repo`. ¿Por qué no? Propagar los cambios hacia `path_repo` usando `git push` desde el directorio `path_copia1`.
11. En el directorio `path_copia2`, integrar en el branch principal (`master`) los cambios del branch `despedida_mejorada`.
12. En el directorio `path_copia2`, el comando `git push` fallará. Explicar por qué falla.
13. Traer los cambios de `path_repo` a la copia de `path_copia2`. (`git fetch`)
14. Identificar el problema usando `git status`. Resolver el problema usando `git pull`. Verificar que el proceso de merge haya sido el adecuado usando `git diff hash_commit`, donde `hash_commit` deberá ser el hash que aparece en la primera línea producida por `git log`.
15. Crear un branch nuevo y, en el mismo, renombrar el archivo `RIDMI.txt` usando el comando `git mv`, que mantiene la historia de cambios del archivo.
16. Crear un branch nuevo y borrar el archivo usando el comando `git rm`.
17. Integrar alguno de los dos branches a `master`.

6. Ciclos a partir de invariantes

A continuación se presentan una serie de ejercicios. Cada uno contiene un invariante asociado. Resolver cada problema respetando, para el ciclo principal del programa, el invariante dado.

Ejercicio 23. *Mínimo de una subsecuencia*

Devolver el índice del mínimo valor de una subsecuencia.

```
proc indice_min_subsec (in s:seq⟨ℤ⟩, in i,j:ℤ, out res:ℤ) {
  Pre { |s| > 0 ∧ 0 ≤ i, j < |s| ∧ i ≤ j }
  Post { i ≤ res ≤ j ∧ (∀k : ℤ) i ≤ k ≤ j ⟶L s[k] ≥ s[res] }
}
```

$$I \equiv i - 1 \leq l \leq j \wedge i \leq res \leq j \wedge (\forall k : \mathbb{Z}) l < k < j \longrightarrow_L s[k] \geq s[res]$$

Ejercicio 24. *Sumatoria de los elementos de una secuencia*

Calcular la suma de todos los elementos de una secuencia s .

$$I \equiv 1 \leq i \leq (|s| \text{ div } 2) + 1 \wedge_L suma = s[(|s| \text{ div } 2)] + \sum_{k=1}^{i-1} s[(|s| \text{ div } 2) - k] + (\text{if } (|s| \text{ div } 2) + k \geq |s| \text{ then } 0 \text{ else } s[(|s| \text{ div } 2) + k] \text{ fi})$$

Ejercicio 25. *Máximo Común Divisor*

Encontrar el máximo común divisor entre dos enteros positivos m y n . La post condición del ciclo es

$$Q_c \equiv a = b = \text{mcd}(a, b).^1$$

$$I \equiv 0 \leq a \leq m \wedge 0 \leq b \leq n \wedge \text{mcd}(a, b) = \text{mcd}(m, n)$$

Ejercicio 26. *División* Dados dos enteros positivos n y d calcular el cociente q y el resto r . El resultado debe ser de tipo $\text{pair} < \text{int}, \text{int} >$, donde el primer elemento de la tupla es q y el segundo es r y cumple

$$Q_c \equiv 0 \leq r < d \wedge 0 \leq q \leq n \wedge n = q \times d + r.$$

$$I \equiv (n \bmod d) \leq r \leq n \wedge 0 \leq q \leq n \wedge n = q \times d + r$$

Ejercicio 27. *Existe Pico*

Una secuencia tiene picos si en alguna posición el elemento es mayor tanto del anterior como del siguiente. Decidir si una secuencia dada (de al menos tres elementos) tiene picos.

$$I \equiv 1 \leq i < |s| \wedge_L res = (\exists k : \mathbb{Z}) 1 \leq k < i \wedge_L s[k] > s[k-1] \wedge s[k] > s[k+1]$$

Supongamos ahora que el invariante cambia de la siguiente manera:

$$I \equiv 1 \leq i < |s| \wedge_L res = \neg(\exists k : \mathbb{Z}) 1 \leq k < i \wedge_L s[k] > s[k-1] \wedge s[k] > s[k+1]$$

¿Le hace falta modificar su código para cumplir con dicho invariante?

Ejercicio 28. *Ordenar 1* Ordenar ascendentemente una secuencia (no vacía) de enteros.

$$I \equiv 0 \leq i \leq |s| \wedge_L |s| = |s_0| \wedge \text{mismos}(s, s_0) \wedge_L \text{ordenada}(\text{subseq}(s, 0, i)) \wedge (\forall k : \mathbb{Z}) 0 \leq k < |s| \wedge i > 0 \longrightarrow_L ((k < i \wedge s[k] \leq s[i-1]) \vee (k \geq i \wedge s[k] \geq s[i-1]))$$

fun *mismos*($s, s_0 : \text{seq}\langle \mathbb{Z} \rangle$) : **Bool** = $(\forall i : \mathbb{Z}) \#apariciones(s, i) = \#apariciones(s_0, i)$

fun *ordenada*($s : \text{seq}\langle \mathbb{Z} \rangle$) : **Bool** = $(\forall i : \mathbb{Z}) 0 \leq i < |s| - 1 \longrightarrow_L s[i] \leq s[i+1]$

Hint: Utilizar la función *indice_min_subsec* resuelta en el ejercicio 1.

¹Recordar que $\text{mcd}(a, b) = \text{mcd}(a-b, b) = \text{mcd}(a, b-a)$