Algoritmos y Estructuras de Datos I
Primer Cuatrimestre 2018

Guía Práctica
EJERCICIOS DE TALLER

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

1. Introducción a C++

Ejercicio 1. Crear un programa (en cualquier editor de texto) y ejecutarlo como se muestra a continuación.

Archivo: labo00.cpp

```
#include <iostream>

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

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

Para 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**. Para todos ellos, utilizar el siguiente esqueleto de archivo y modificarlo con los procedimientos que implementen.

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 resultado:

- si es primo "El número ingresado es primo"
- y si no lo es "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 un variable `f`. Mostrar los valores leídos en la salida estaándar
Ambos valores están separados por una coma y hay una única línea en el archivo.

- archivo.txt:
- `-234,1.7`

Ejercicio 14. El archivo `numeros.txt` contiene una lista de números separados por espacios. Leer los números del archivo 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:$\mathbb{Z}$, inout b:$\mathbb{Z}$) {

`Pre` $\{a = a_0 \wedge b = b_0\}$
`Post` $\{a = b_0 \wedge b = a_0\}$
}

Ejercicio 17. Implementar la función division que cumpla con la siguiente especificación:

`proc division` (in dividendo $\mathbb{Z}$, in divisor $\mathbb{Z}$, inout cociente:$\mathbb{Z}$, inout resto:$\mathbb{Z}$) {
`Pre` $\{dividendo \geq 0 \wedge divisor > 0\}$
`Post` $\{dividendo = divisor * cociente + resto \wedge 0 \leq resto < divisor\}$
}
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 ademas a=b-2. Queremos obtener los iesimos 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-esimos primos gemelos son 17 y 19. Además se debe escribir en un archivo la secuencia de primos gemelos hasta llegar al i-esimo.
Para el ejemplo el archivo debe contener: (3,5) (5,7) (11,17) (17,19)