Universidad Nacional de Río Cuarto
Facultad de Cs. Exactas, Fco-Qcas y Naturales
Departamento de Computación
Asignatura: **INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN** Año: 2022

Práctico Nº 11

Tema: Recursividad **Duración**: 3 clases

Esta práctica tiene como objetivos:
- Conocer sobre algunos aspectos básicos del paradigma funcional
- Incorporar el concepto de recursividad, utilizando el paradigma imperativo
- Emplear recursividad de cola y en aumento en la construcción de funciones y acciones
- Realizar pruebas de escritorio sencillas
- Implementar en C algoritmos que hagan uso de la recursividad

Ej. 1.a) Defina la función recursiva **potencia** de un número natural y exponente natural que está en la teoría.
1.b) Defina la función recursiva **fibonacci** que está en la teoría.
1.c) Defina una función recursiva que reciba como parámetro un número natural y devuelva la **sumatoria** de los números desde el uno hasta el número dado incluido. Haga una prueba (como en la diapositiva 21), invocando a la función recursiva con parámetro actual de valor 4.

1.d) Defina una función recursiva que reciba como parámetro un número natural y devuelva la **productoria** de los números pares desde el uno hasta el número dado incluido.

Ej. 2.a) Defina una acción recursiva que reciba como parámetro un arreglo de hasta 30 números enteros, y la cantidad de valores cargados; y retorne en una variable, a llamar ***suma***, la sumatoria de todos los números contenidos en el arreglo.

2.b) Defina una acción recursiva que reciba como parámetro un arreglo de hasta 30 números enteros y la cantidad de valores cargados; y retorne en una variable a llamar ***producto***, el producto de todos números contenidos en el arreglo, entre la posición 1 y n, siendo n un parámetro que se pasa y que puede tomar el valor 1 hasta 30 como máximo.

Ej. 3) Defina una acción recursiva que reciba como parámetro un arreglo de 30 números enteros y la cantidad de valores cargados; y retorne en una variable a llamar ***pares*** el valor Verdadero si todos los números contenidos en el arreglo son pares sino que retorne Falso.

Ej. 4) Defina una acción recursiva que reciba como parámetro un arreglo de 30 caracteres y la cantidad de valores cargados; y retorne en una variable a llamar ***contA*** la cantidad de letras “a” que hay en el arreglo.

Ej. 5) Dada una lista simplemente encadenada del siguiente tipo

tipo TElemento = <nro ∈ Z, sig ∈ puntero a TElemento>

escribir las siguientes funciones o acciones recursivas:

5.a) ***LongElem***, función que dado un puntero al primer elemento de la lista retorne la cantidad de elementos, es decir la longitud de la misma.

5.b) ***Suma,*** función que dado un puntero al primer elemento de la lista retorne la suma de los elementos de la misma.

Ej. 6.a) Defina una función recursiva que reciba como parámetro una LSE de números enteros y retorne como resultado la cantidad de números pares que contiene.

6.b) Idem pero con una acción recursiva (agregando un parámetro para el resultado).

Ej. 7) Defina una acción recursiva que reciba como parámetro un número natural *n*, si *n* es par la acción debe dar por salida todos los números pares comprendidos entre *n* y 0, y si *n* es impar debe dar por salida todos los números impares entre *n* y 0.

Ej. 8) Defina una función que dado un puntero al primer elemento de una LSE de números enteros retorne el mayor valor de la lista.

Ej. 9) Dada las siguientes funciones, resueltas como **recursión en aumento**, hallar para cada caso una solución de **recursión de cola**.

9.a)
```
Función factorial (dato n ∈ N) → N
 {Def: (n0=0 ∧ fact(n0)=1) ∨ (n0>0 ∧ fact(n0)=1*2*..*n0)}
Inicio
 según
   n=0: ← 1
   n>0: ← n * factorial(n-1)
 fsegún
Ffunción
```

9.b)
```
TElem = <info ∈ m, next ∈ puntero a TElem>
 TLista = puntero a TElem

Función long (dato l ∈ TLista) → N
Inicio
 según
   l=nil: ← 0
   l≠nil: ← 1 + long((^l).next)
 fsegún
Ffunción
```

9.c)
```
Función contOcu (dato a ∈ arreglo [1..254] de Carácter, u ∈ (0..254), c ∈ carácter) → (0..254)
    {Def: ((contOcu(a,u,c,res)=0 ∧ el arreglo a esta vacío o no existe ningún carácter igual a c) ∨
    (contOcu(a,u,c,res)<>0 ∧ existe uno o más caracteres en a que son iguales a c)) ∧ c=Co }
    Inicio
     según
       u=0: ← 0
       a[u]=c: ← 1+ contOcu(a, u-1,c)
      a[u]<>c: ← contOcu(a, u-1,c)
     fsegún
    Ffunción
```

10) Analice si las soluciones obtenidas en el ejercicio 5.a), 5.b) y 5. c) son de recursión en aumento o de recursión en cola.

11) Resuelva el ejercicio 1.a) y 1.b) de esta práctica pero dando una solución con recursión en cola.

Preguntas Teóricas:

I) PREGUNTA: ¿Qué es el caso base y que es el caso inductivo? ¿hay diferencias entre ambos?

II) PREGUNTA: ¿Cómo nos damos cuenta que estamos usando recursión de cola?

III) PREGUNTA: ¿Cómo nos damos cuenta que estamos usando recursión en aumento?

IV) PREGUNTA: ¿Qué diferencia hay en el uso de la memoria entre recursión de cola y en aumento?

Plan de clases

Clase 1 ejercicios 1 a 4

Clase 2 ejercicios 5 a 8

Clase 3 ejercicios 9 a 11