Trabajo Práctico 8
División de Problemas en Subproblemas
Uso de Procedimientos y Funciones

Ejercicio 1: Considere definida una función.

function *Invertir*(Num: integer):integer;
{ Objetivo: Invierte el orden de los dígitos del número entero
Num: número que se desea invertir.
Salida: Devolverá un entero con los dígitos de Num en orden inverso }

Realice un programa que, utilizando la función dada, determine si un número natural Num ingresado por el usuario es o no capicúa. Por ejemplo,
si Num = 12321, el programa deberá mostrar por pantalla "El número 12321 ES CAPICUA".
si Num = 2343, el programa deberá mostrar por pantalla "El número 2343 NO ES CAPICUA"

Ejercicio 2: Implemente las siguientes funciones

function *factorial*(N: integer): integer;
{ Objetivo: esta función calcula el factorial de un número N.
N: número al cual se le desea calcular el factorial.
Salida: la función devolverá el valor correspondiente a N! }

function *exponente*(Base, Exponente: integer): integer;
{ Objetivo: esta función calcula $Base^{Exponente}$.
Salida: la función devolverá el valor correspondiente a $Base^{Exponente}$ }

Ejercicio 3: Escriba un programa en Pascal dividiendo el problema en subproblemas, que lea un número natural *n* y muestre por pantalla todos los números primos comprendidos entre *1* y *n.*

Por ejemplo: para *n* = 20, el programa deberá mostrar por pantalla:
Los números primos entre 1 y 20 son: 2 3 5 7 11 13 17

Ejercicio 4: Escriba un programa en Pascal dividiendo el problema en subproblemas, que lea tres números enteros a, b y c y muestre el máximo de los valores absolutos, sin utilizar la función predefinida abs.

Por ejemplo: para a = 26, b = -45, c= 3, el programa deberá mostrar por pantalla:
El máximo valor absoluto es 45

Ejercicio 5: Escriba un programa en Pascal dividiendo el problema en subproblemas, que elimine de un archivo de enteros todos las componentes que sean múltiplos de 2 o múltiplos de 5 o primos o terminen en 7.

Ejercicio 6: Se dice que *M* es el número maximal para *N*, si *M* es **el mayor número que puede formarse usando los dígitos de *N***. Ejemplos: Si *N*=125345, el número maximal *M* es 554321; si *N*=2756, *M* es 7652.

Escriba un programa en Pascal que lea dos números naturales *a* y *b* y muestre por pantalla todos los números *Num* comprendidos entre *a* y *b* que verifiquen que coinciden con su maximal:

Por ejemplo: para *a* = 320 y *b* = 332, el programa deberá mostrar por pantalla:
Los números entre 320 y 332 que coinciden con su maximal son: 320, 321, 322, 330,331, 332

Ejercicio 7: Para cada uno de los incisos a continuación calcule el resultado para n=0, n=2 y n=6 (suponiendo m=4).

a) $1+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots+\frac{1}{n!}$

b) $n+\frac{n-1}{2!}+\frac{n-2}{3!}+\frac{n-3}{4!}+\cdots+\frac{1}{n!}$

c) $\frac{1}{n!}+\frac{2^2}{(n-1)!}+\frac{3^4}{(n-2)!}+\frac{4^6}{(n-3)!}+\frac{5^8}{(n-4)!}+\cdots+\frac{n^{2(n-1)}}{1!}$

d) $1 - (n + 1) + (n^2 + n + 1) - (n^3 + n^2 + n + 1) + ... + (-1)^n (n^n + n^{(n-1)} + n^{(n-2)} + ... + 1)$

e) $1+\frac{m}{1}n+\frac{m*(m-1)}{2!}n^2+\frac{m*(m-1)*(m-2)}{3!}n^3+\cdots+n^m=\sum_{i=0}^{m}\left(\frac{\prod_{j=0}^{i-1}m-j}{i!}n^i\right)$

f) $\frac{4}{\pi}\sum_{k=1}^{m}\frac{\sin((2k-1)n)}{2k-1}$

Ejercicio 8: Para cada uno de los incisos del ejercicio 7 escriba un programa que calcule el resultado de dicha sumatoria. *Divida el problema en partes, por ejemplo, considerando que las partes de una sumatoria se denominan sumandos y que las partes de un producto se conocen como factores. Así podría definirse sumando(i) o termino(i) que retorne el valor del término de dicha posición, que a su vez puede calcularse en función de factor(i), etc.*

Ejercicio 9: Suponga que cuenta con un archivo A de números enteros ya ingresados. Para cada uno de los siguientes incisos se desea generar otro archivo nuevo con los elementos de A que cumplan lo indicado:

a) Sean capicúas y tengan una cantidad impar de dígitos.
b) Sean primos o tengan todos los dígitos impares.
c) Tengan una cantidad par de dígitos, no sean capicúas y tengan al menos un dígito par.

Para cada uno de los incisos anteriores (en forma individual) se solicita que:

i) Divida el problema en subproblemas y haga un gráfico o esquema de su propuesta de diseño para la solución.
ii) Describa las funciones y procedimientos necesarios identificando los parámetros de entrada y salida, agregando una breve descripción del objetivo de la primitiva.
iii) Realice un programa en PASCAL que resuelva el problema. No es necesario implementar las primitivas, simplemente deberá declarar los encabezados de cada una.

Ejercicio 10: Reescriba el programa generado en el **ejercicio 1** utilizando el siguiente procedimiento.

PROCEDURE *Invertir*(Num: integer; **var** NumInvertido:integer);
{ Objetivo: Invierte el orden de los dígitos del número entero
Num: número que se desea invertir
Salida: Devolverá en NumInvertido los dígitos de Num en orden inverso }

Ejercicio 11: Indique cuantos parámetros por valor y cuantos por referencia hay en los siguientes procedimientos y funciones:

1. **PROCEDURE** *Eje1*(**var** letra1,letra2:char; N1,N2:integer; **var** Error:boolean);
2. **PROCEDURE** *Eje2*(**var** A:char; **var** b:integer; **var** c:boolean);
3. **FUNCTION** *F1*(a,b:integer; es: boolean):real;
4. **FUNCTION** *LeeLetra*: CHAR;
5. **FUNCTION** *LeeNumero*(l:char; **var** error:boolean):integer;

Ejercicio 12: Implemente un procedimiento que dado un dígito d ∈ [1..9] muestre por pantalla el siguiente renglón:

1 2 3 .. d .. 3 2 1

Por ejemplo, si **d = 6** el procedimiento deberá imprimir

1 2 3 4 5 6 5 4 3 2 1

El encabezamiento del procedimiento sería:

PROCEDURE ImprimeRenglón(digito:integer);
{Objetivo: Imprime en pantalla un renglón de la forma 1 2 3 .. d .. 3 2 1 donde d es un dígito de 0 a 9}

Escriba un programa en Pascal utilizando dicho procedimiento, para que solicite un dígito **d** al usuario, y muestre por pantalla una figura como la siguiente

1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
...
...
1 2 3 .....d..... 3 2 1

Ejercicio 13: Analizar cuáles de las invocaciones a procedimientos o funciones detalladas a continuación son correctas en base a las siguientes declaraciones.

```
VAR w: Char;
        x: Integer;
        y: Real;
        z: Boolean;

PROCEDURE Proc1(a,b: Integer; var c: Char);
BEGIN ... END;

FUNCTION Funcion1(x: char):Real;
BEGIN ... END;

FUNCTION Funcion2(VAR a: Real; b: Boolean):Integer;
BEGIN ... END;
```

1. **Proc1(7, y, w);**
2. **Proc1(7, y, c);**
3. **Proc1(27, x, w, w);**
4. **Proc1(2.4, 5+8, w);**
5. **Proc1(7, 5, 'c');**
6. **x:= Funcion1(w);**
7. **y:= Funcion1(w);**
8. **y:= Funcion1('x');**
9. **Funcion1(w);**
10. **x := Funcion2(y, false);**
11. **y := Funcion2(y, true);**
12. **x := Funcion2(3+5, z);**
13. **x := Funcion2(3.5+y, z);**

Ejercicio 14: Suponga que cuenta con tres archivos A, B y C, y todos tienen ingresados números reales. Para cada uno de los siguientes incisos se desea generar otro archivo nuevo con aquellos elementos de A, B y C, que respeten lo indicado:

a) Copiar todos los elementos que se encuentre en A y en B pero no en C.
b) Copiar aquellos elementos que aparezcan en A una cantidad par de veces o aparezcan en B una cantidad impar de veces, y que si aparece en C entonces no debería aparecer en la misma cantidad que apareció en A o en B.
c) Copiar todo elemento que ocurra en A antes que en B, y en B antes que en C. Si el elemento no aparece en algún archivo entonces no se copia.

Para cada uno de los incisos anteriores (en forma individual) se solicita que:

i) Divida el problema en subproblemas y haga un gráfico o esquema de su propuesta de diseño para la solución.
ii) Describa las funciones y procedimientos necesarios identificando los parámetros de entrada y salida, agregando una breve descripción del objetivo de la primitiva.
iii) Realice un programa en PASCAL que resuelva el problema. No es necesario implementar las primitivas, simplemente deberá declarar los encabezados de cada una.

Ejercicio 15: Suponga que cuenta con un archivo "alumnosAM2.dat" de números enteros que contiene el número de registro de todo alumno inscripto para cursar la materia Análisis Matemático 2. También se dispone de 3 archivos adicionales con las notas que obtuvieron dichos alumnos en cada examen. Llamaremos “parcial1.dat” y “parcial2.dat” a los archivos que contienen las notas de los parciales y “recu.dat” al archivo que contiene la nota del recuperatorio. Cada archivo de notas almacenará la información según el siguiente formato “registro nota registro nota … <EOF>”. Las notas solo podrán ser valores enteros entre cero y diez, y todo registro que aparece en alguno de los archivos de notas deberá estar presente en el archivo de alumnos (*estas propiedades no necesitan ser verificadas*). Si un alumno no se presentara a rendir un parcial, su nro de registro no aparecería en el archivo correspondiente, por ejemplo, si el alumno 96546 no viene al 2do parcial, dicho registro no aparecerá en el archivo “parcial2.dat”. Se considera aprobado un examen si ha obtenido una nota mayor o igual a 6. Ninguno de los archivos se encuentra ordenado.

Se solicita generar tres archivos donde uno de ellos contenga los registros de todos los alumnos que cursaron la materia (es decir, sacaron más de 6 en ambos parciales o aprobaron el recu) y el promedio obtenido entre todas sus notas, otro archivo que contenga los registros de aquellos alumnos que perdieron la materia y otro archivo con los números de registro de los alumnos que estuvieron ausentes a todos los parciales.

i) Divida el problema en subproblemas y haga un gráfico o esquema de su propuesta de diseño para la solución.
ii) Describa las funciones y procedimientos necesarios identificando los parámetros de entrada y salida, agregando una breve descripción del objetivo de la primitiva.
iii) Realice un programa en PASCAL que resuelva el problema. No es necesario implementar las primitivas, simplemente deberá declarar los encabezados de cada una.

Ejercicio 16: Considere el siguiente programa en Pascal y muestre que salida se producirá en la pantalla como resultado de su ejecución.

```
PROGRAM p7;
procedure multiplicarXvalor(x,y,z: integer);  // Usando el metodo de sumas sucesivas
var i:integer;
begin
    z := 0;
    for i:= 1 to y do z := z + x;
end;
```

```
procedure multiplicarXref(x,y: integer; var z: integer);  // Usando el metodo Ruso
begin
   z := 0;
   while x >0 do begin
      if (x mod 2) <> 0 then z := z + y;
      x := x div 2;
      y := y + y;
   end;
end;
procedure multiplicarXref2(var x,y,z: integer);    // Usando sumas sucesivas con efectos colaterales
begin
   while y > 0 do begin
      z := z + x;
      y := y - 1;
   end;
end;
var  a,b,z: integer;
begin
    a:=12;   b:=4;   z:=1; writeln('1- A:',a,' B:',b,' Z:',z);
    multiplicarXvalor(a,b,z);
    writeln('2- A:',a,' B:',b,' Z:',z);  z := 1;
    multiplicarXref(a,b,z);
    writeln('3- A:',a,' B:',b,' Z:',z);  z := 1;
    multiplicarXref(a,b,z);
    writeln('4- A:',a,' B:',b,' Z:',z);  z := 1;
    multiplicarXref2(a,b,z);
    writeln('5- A:',a,' B:',b,' Z:',z);  z := 1;
    multiplicarXref2(a,b,z);
    writeln('6- A:',a,' B:',b,' Z:',z);
    readln;
end.
```

Ejercicio 17: El período de un caracter se define como la distancia entre dos apariciones del mismo. Por ejemplo, si tiene la siguiente secuencia "parte de la traza es trabajosa." El período de **t** es 8 que es la cantidad de caracteres que separan las t. Cuando se considera el caracter **a** se observa que existen varios ocurrencias con diferentes períodos 8, 3, 1, 6, 1 y 3 (obsérvese que sólo mide la distancia de un caracter al próximo inmediato idéntico, pero no se mide la distancia entre cualquier par, por ejemplo, la distancia entre la **a** de **P<u>a</u>rte** y la **a** de **tr<u>a</u>za** no se mide). Diremos que un **caracter es periódico** si ocurre siempre con la misma periodicidad.

Considere que se dispone de dos archivos "valores.in" y "registro.dat", donde "valores.in" registra que elementos aparecen en "registro.dat". Por ejemplo, para la secuencia de arriba el archivo "valores.in" contendría los siguientes valores "p a r t e d l z s b j o .".

a) Realice un programa que muestre todos los caracteres periódicos (junto con su periodicidad) a basado en los datos de "registro.dat" pudiendo ayudarse del archivo "valores.in".

b) Realice un procedimiento que reciba como entrada un archivo que contiene una secuencia de caracteres y genere otro archivo "valores.in" que contenga únicamente los caracteres que aparezcan en el archivo provisto, con la particularidad que cada carácter aparecerá solo una vez. Por ejemplo, el carácter **a** aparecerá solo una vez en "valores.in" a pesar que en la secuencia aparece reiteradas veces y el carácter **u** no aparecerá ya que no se encuentra en la secuencia ingresada

Ejercicio 18: Las personas de un pueblo chico tienen distintos lazos de amistad entre sí. Todo andaba bien en el pueblo, hasta que los vecinos X e Y se pelearon y empezaron a disputar el liderazgo. El pueblo se revolucionó y para evitar más peleas quisieron saber cuántos aliados tenía cada líder.

La relación de fuerza de amistad está catalogada con un número natural.

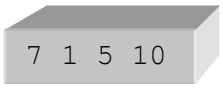
- Un vecino Z es aliado de X en lugar de Y si el lazo de amistad que une a Z con X es mayor estricto que el lazo de amistad de Z con Y.
- También, un vecino Z es aliado de X en lugar de Y si existe un lazo de amistad que lo une a X y no existe un lazo de amistad con Y.
- Por otra parte, no podemos definir si Z está aliado a X o a Y, si no están definidos lazos de amistad con ninguno de ellos, o bien Z posee el mismo lazo de amistad con ambos.

Realice un programa en Pascal para determinar cuántos aliados tiene cada vecino participante de la pelea (X e Y) descomponiendo el problema en subproblemas que permitan resolver y entender de forma más sencilla el problema. Considere que recibe dos archivos, DATOS.IN y ALIADOS.IN, donde:

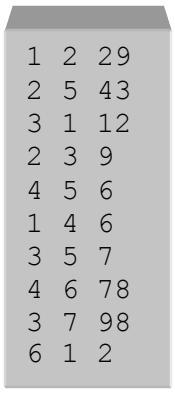
DATOS.IN consiste de los siguientes números naturales: - N: cantidad de vecinos, $2 \le N \le 100$, - X: el primer oponente, $1 \le X \le N$, y - Y: el segundo oponente, $1 \le Y \le N$. - M: cantidad total en ALIADOS.IN de lazos de amistad, $0 \le M \le 5000$.	ALIADOS.IN consiste de M ternas que representan las relaciones de amistad y cada terna consta de: - K que representa a un vecino, - R representa otro vecino, y - L representa la fuerza de amistad entre K y R. $1 \le L \le 100$

El programa debe generar un archivo ALIADOS.OUT con 1 sola línea conteniendo dos números, que representan la cantidad de aliados de X e Y respectivamente

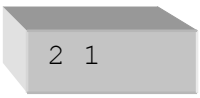
Ejemplo



DATOS.IN



ALIADOS.IN



ALIADOS.OUT

Suponga que X=1 y que Y=5, obsérvese que 1 tiene como amigos a 2, 3, 4 y 6; y que 5 tiene como amigos a 2, 4 y 3.
La fuerza de amistad entre 1 y 2 es 29, y entre 2 y 5 es 43, en consecuencia 2 es aliado de 5 porque es más fuerte el vinculo. Por otro lado 3 es aliado de 1 (12 > 7), 4 es imparcial (6=6) y 6 es sólo amigo de 1, entonces 1 tiene 2 aliados (3 y 6) y 5 tiene 1 aliado (2).

Ejercicio 19: Conteste las siguientes preguntas dando un ejemplo en el caso que la situación planteada sea posible, o fundamentando su respuesta con conceptos teóricos. Dentro de un programa en Pascal:

- ¿Pueden dos procedimientos tener el mismo nombre?
- ¿Pueden haber dos funciones con el mismo identificador?
- ¿Puede un identificador de constante ser igual a un identificador de variable?
- Indique cuando un procedimiento P puede llamar a una función F que está declarada dentro de otro procedimiento Q, y cuando no.
- ¿Puede una variable local tener como nombre V si está declarada dentro de un procedimiento cuyo nombre también es V?
- ¿Puede una variable local tener como nombre V si está declarada dentro de una función cuyo nombre es V? ¿Hay alguna diferencia con respecto a que V sea un procedimiento?