Práctica de Laboratorio
Estructuras de Control en Pascal

1. Requerimientos

- Tener EclipseGavab instalado, junto con la librería PFC señalada en el anterior laboratorio.
- En caso de usar dispositivos móviles, descargar e instalar aplicaciones donde se use lenguaje Pascal.

2. Marco teórico

2.1. Acerca de Pascal y Pascal-FC

El lenguaje de programación Pascal es un lenguaje de alto nivel para el aprendizaje de programación estructurada e iniciación de las estructuras de datos de todo tipo, desde programas secuenciales como ordenar arreglos y archivos secuenciales, hasta procesos basados en algoritmos recursivos, estructuras dinámicas de información y compiladores.

La estructura de un programa básico en Pascal consta de un encabezamiento, una sección de declaraciones y una sección de sentencias ejecutables para escribir código eficiente, optimizado, y menos complejo posible:

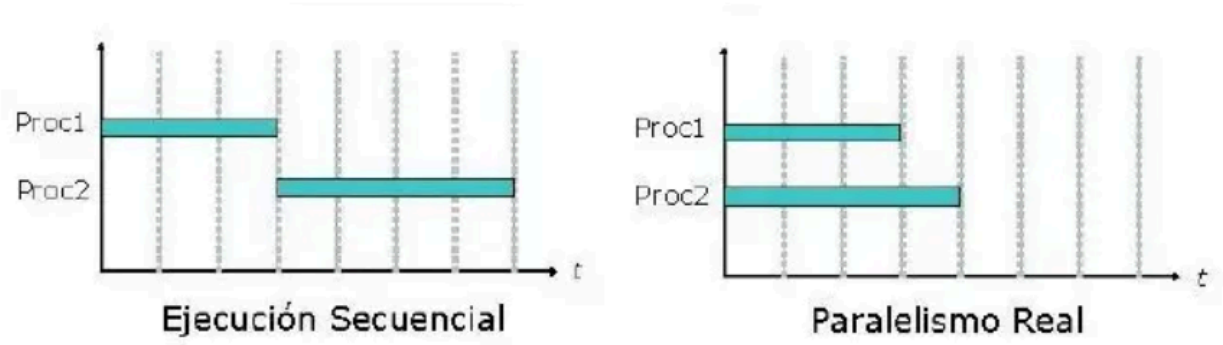
(1) Encabezamiento: Al principio la palabra "**program**", seguida del nombre del programa.
(2) Declaraciones y definiciones: Se declaran variables, procesos funciones, etc. Como declaraciones globales se encuentran:
- Const: definiciones de constantes
- Type: definición de tipos de datos definidos por el usuario
- Var: declaraciones de variables (BOOLEAN, CHAR, INTEGER, REAL)
- Procedure: definición de procedimientos y subprogramas
- Function: definición de funciones y subprogramas.

(3) Parte Ejecutable: Se empieza y termina el programa mediante begin... end, conteniendo dentro sentencias de ejecución.

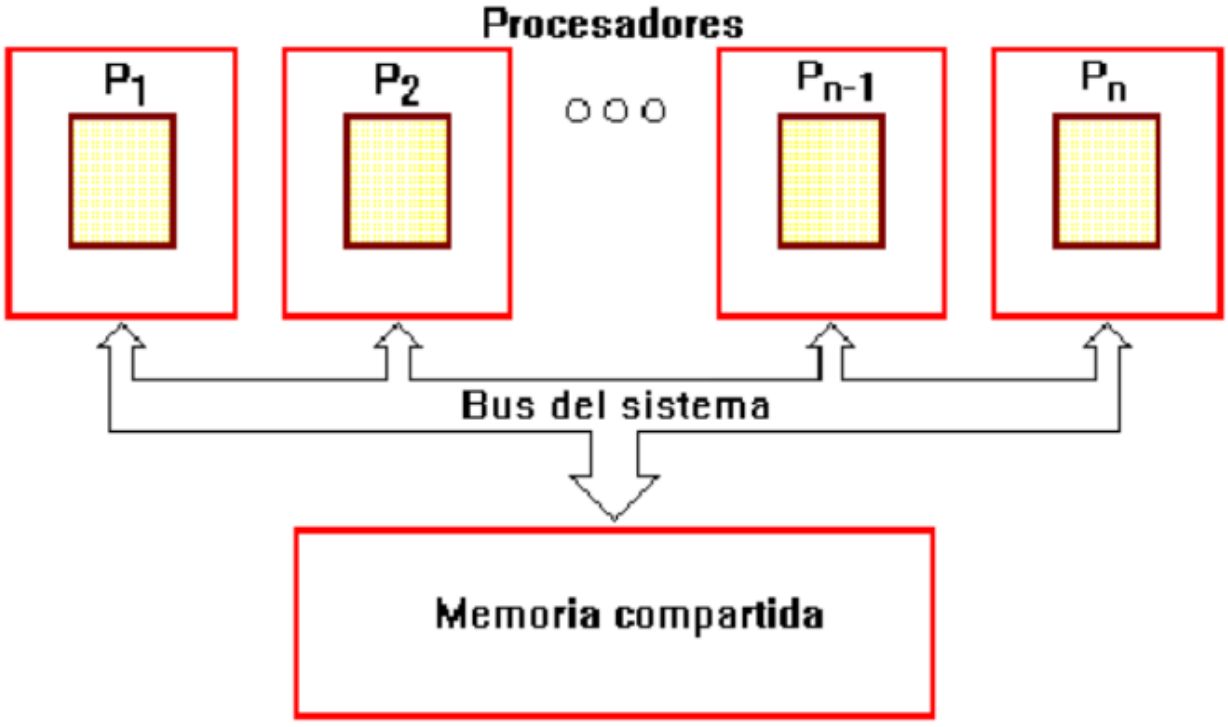
Por otra parte, el lenguaje Pascal-FC es una extensión de Pascal orientado al aprendizaje de programación de procesos concurrentes que internamente contempla programa secuencial y procesos hechos en lenguaje Pascal:

- Programa secuencial: Son un conjunto de declaraciones e instrucciones de datos que se ejecutan una a continuación de otra, declarada a partir de un algoritmo, usado generalmente en lenguajes Pascal, siendo:
  - Conjunto de sentencias y declaración de variables.
  - Procedimientos o funciones.
- Proceso: Es la ejecución de un programa secuencial, tomando en cuenta las siguientes acciones:
  - Ejecuta un procedimiento o función.
  - Internamente en Pascal hay un solo proceso

- Internamente en Pascal-FC pueden existir varios procesos de un mismo programa secuencial cuando se ejecuta varias veces de forma simultánea.
- Ejecuta sub procesos, también llamados hilos de ejecución (Thread).



Durante la ejecución entre dos o más procesos concurrentes, la primera instrucción de uno de ellos se ejecuta entre la primera y última instrucción del otro, donde cada instrucción son un conjunto de programas secuenciales de un sistema informático, similar a la arquitectura de procesamiento de bus de sistema de un ordenador de un solo procesador, o de un servidor de varios procesadores, donde sus procesos pueden acceder a una memoria en común.



2.2. Características de lenguaje Pascal

- No existe distinción entre mayúsculas y minúsculas.
- Existen variables compartidas que varios procesos pueden leer y escribir.
- Los procesos se intercambian mensajes entre sí.
- Un proceso envía un mensaje y otro lo recibe.

- Puede usar funciones establecidas como:
- Emplea los siguientes operadores:

Operadores Aritméticos	
Suma	+
Resta	-
Multiplicación	*
División	/
div	División de Enteros (21 div 10 = 2)
Mod	El resto de la división (21 Mod 10 = 1)
Operadores de Relación	
"Menor" | "Menor o igual a"	< | <=
"Mayor" | "Mayor o igual a"	> | >=
Distintos	<>
Iguales	=
Operadores Boleanos	
... y ...	and
... o ...	or
no es ...	not

2.3. Sentencias secuenciales

Son una serie de pasos de algoritmos impresos uno detrás del otro, donde el primer paso que se haya escrito será el primero que se ejecute.

Para el presente laboratorio se van a declarar primero variables, se emplea la expresión ":=" para asignar a la variable un valor.

```
program operaciones;
begin
  writeLn('5+2=', 5+2);
  writeLn('5-2=', 5-2);
  writeLn('5/2=', 5/2);
  writeLn('5*2=', 5*2);
  writeLn('7 div 2 = ', 7 div 2);
  writeLn('7 mod 2 = ', 7 mod 2);
  writeLn('La raiz de 5 es ', sqrt(5));
  writeLn('Numero al azar: ', random(10) );
end.
```

```
program ReadLn1;

var
  numero, doble: integer;

begin
  write('Introduce un numero: ');
  readLn(numero);
  doble := numero * 2;
  writeLn('Su doble es ', doble);
end.
```

```
program Enteros1;
var
  edad: integer;
  meses : integer;
begin
write('Dime tu edad: ');
  readLn(edad);
  meses := edad * 12;
  writeLn('Aproximadamente tienes ', meses, ' meses');
end.
```

Se usan números enteros

Para la entrada de datos, se van a utilizar los procedimientos read y readln, siendo este último igual al primero, con la diferencia de generar un salto de línea, siendo la sintaxis:

read(a, b, c, ...);

Donde a, b y c son las variables a leer.

Cuando se va leyendo, se ingresa cada valor en una línea separada por al menos un espacio, terminando con la presión de la tecla ENTER.

```
program coseno;
var
  pi, angulo, resultado: real;

begin
  pi := 3.1416;
  angulo := pi / 4; (* 45 grados = PI / 4 radianes *)
  resultado := cos(angulo);
  writeLn('El coseno de 45 grados es ', resultado:4:3);
end.


(*
El coseno de 45 grados es 0.707
*)
```

Como:

3e2 o 3E+2 equivaldria a $3x10^2$, es decir, 3x100 = 300

3e-2 o 3.0E-002 equivaldria a $3x10^{-2}$, es decir, 3x0.01 = 0.03

Entonces:

Para que el decimal sea más legible, se indica:

- El número decimal.
- La cantidad de digitos totales del número.
- La cantidad de digitos decimales del número

```
write(numero:cifrasTotales:cifrasDecimales);
```

```
(* Numeros reales y formato en pantalla *)

program Reales2;

var
  centimetros, metros: real;

begin
  write('Dime la longitud, en centimetros: ');
  readLn(centimetros);
  metros := centimetros / 100;
  writeLn('Equivalen a ', metros:5:4, ' metros');
end.

(* Ejemplo de ejecucion:
Dime la longitud, en centimetros: 32.8
Equivalen a 0.3280 metros
*)
```

Se usan números reales

Para la salida de datos se van a utilizar los procedimientos write y writeln, siendo este último igual al primero, con la diferencia de generar un salto de línea, siendo la sintaxis:

write(a, b, c, ...);

Todos los elementos que aparezcan en un solo write o writeln se escriben en una misma línea.

```
(* Caracteres, toma de contacto *)

program Caracteres;

var
  letra1, letra2: char;

begin
  write('Dime una letra: ');
  readLn(letra1);
  write('Dime otra letra: ');
  readLn(letra2);
  writeLn('Las letras eran ', letra1, ' y ', letra2);
end.

(* Ejemplo de ejecucion:
Dime una letra: a
Dime otra letra: s
Las letras eran a y s
*)
```

Para variables de tipo caracter, los valores se encierran en comillas simples.

```
(* Caracteres y valores prefijados *)

program Caracteres2;

var
  letra1, letra2: char;

begin
  write('Dime una letra: ');
  readLn(letra1);
  letra2 := 'Z';
  writeLn('Tu letra era ', letra1, ' y la mia ', letra2);
end.

(* Ejemplo de ejecucion:
Dime una letra: a
Tu letra era a y la mia Z
*)
```

Se usan valores caracteres

2.4. Sentencias selectivas

Esta estructura se utiliza para tomar decisiones, evaluando alguna condición para luego llevar a cabo alguna de las opciones. Estas se dividen en estructuras simples, estructuras dobles, estructuras compuestas y estructuras múltiples.

2.5. Sentencias Selectivas Simples

Son conocidas por evaluar una condición y si esta llega a cumplirse, realizará una serie de pasos. Su sintaxis es “If... then”

```
program par;
VAR
    t,num:INTEGER;
begin
write('Introduzca un numero entero: ');
readln(num);
if ((num mod 2 = 0)) THEN
    WRITE ('El numero introducido es par');
end.


(*
Introduzca un numero entero: 4
El numero introducido es par
*)
```

```
program if5;

var
    numero: integer;
    esPositivo: boolean;

begin
    writeLn('Escriba un numero');
    readLn(numero);
    esPositivo := numero>0;
    if esPositivo then writeLn('El numero es positivo');
end.
(* Ejemplo de ejecucion:
Escriba un numero
2
El numero es positivo
*)
```

2.6. Sentencias selectivas dobles

Similar a la sentencias selectivas simples, se evalúa una condición, si es verdad, se realizará una serie de pasos, caso contrario, se realizará otra serie de pasos.

Su sintaxis es “If... else... then”, donde nunca se escribe “;” antes del “else” porque eso indicaría el final de la sentencia “if”

```
program if2;

var
    numero: integer;

begin
    writeLn('Escriba un numero');
    readLn(numero);
    if numero<0 then
        writeLn('El numero es negativo.')
    else
        writeLn('El numero es positivo o cero.')
end.

(* Ejemplo de ejecucion:
Escriba un numero
26
El numero es positivo o cero.
*)
```

Se recomienda que si se escriben dos “if”, no generara el mismo comportamiento que escribir “If... else... then”, ya que al usar dos “if” analizara siempre la segunda condición, incluso en caso de que se haya cumplido la primera.

```
program if2b;

var
    numero: integer;

begin
    writeLn('Escriba un numero');
    readLn(numero);
    if numero < 0 then
        writeLn('El numero es negativo.')
    if numero >= 0 then
        writeLn('El numero es positivo o cero.')
end.

(* Ejemplo de ejecucion:
Escriba un numero
26
El numero es positivo o cero.
*)
```

Como se puede observar, las líneas que "dependen" de la anterior (como la orden que sigue a un "if" o a un "else") están un poco más a la derecha, se aconseja no usar el símbolo de tabulación delante del texto cuando se añade una “sangría” adicional, sino espacios en blanco, y la cantidad de espacios que se suele emplear es 4.

2.7. Sentencias Selectivas Compuestas

Se utiliza para realizar múltiples acciones cuando se cumple una determinada condición. Las sentencias compuestas deben estar rodeadas por "begin" y "end", además, no es necesario escribir ";" antes de llegar a la línea de "end".
La sintaxis es similar a las sentencias simples o compuestas.

Ejemplo:

```
program ssc;
var
   numero:integer;
begin
  writeLn('Escriba un numero: ');
  readLn(numero);
  if numero<0 then
   begin
     writeLn('El numero es negativo.');
     readLn
   end;
end.
```

Ejecución:

```
Escriba un numero:
-5
El numero es negativo.
```

Para hacerlo más entendible: los pasos que forman parte de cada "sentencia compuesta" se tabulan adicionalmente con una posición a la derecha, por lo que facilita ver dónde comienza y termina el bloque.
También se pueden enlazar varias condiciones usando los operadores “and”, “or” y el “not”.

```
IF ( opcion = 1 ) AND ( puntos > 500 ) THEN [...]

IF ( opcion = 3 ) OR ( enemigos = 0 ) THEN [...]

IF NOT ( vidas > 0 ) THEN [...]

IF ( opcion = 2 ) AND NOT ( nivelDeAcceso < 40 ) THEN [...]
```

2.8 Sentencias Selectivas Múltiples

Se utiliza cuando el programa tiene que realizar la comprobación de muchas opciones, para ello se tiene la orden “case”.

```
CASE expresion OF
     caso1: sentencia1;
     caso2: sentencia2;
     ...
     casoN: sentenciaN
ELSE
     sentencia
END;
```

Luego ingrese una sentencia para cada “caso” con la idea que cumplirá la primera de ellas, y agregue una declaración "else" si ninguna de las opciones coincide.

Ejemplo:

```
program ssm;
var
  grado: char;
begin
  write('Ingrese el grado: ');
  readln(grado);
  case(grado) of
      'A': writeln('Excelente');
      'B','C': writeln('Bien hecho');
      'D': writeln('Pasaste');
      'F': writeln('Intentalo de nuevo');
  else
      writeln('No es un grado');
  end;
  readln
end.
```

Ejecución:

```
Ingrese el grado: A
Excelente
Tu grado es: A
```

Se recomienda que la "expresión" sea un tipo de datos con un número finito de elementos como "integer" o "char" pero no "real".

2.9. Sentencias Repetitivas

La declaración de repetición ejecuta repetidamente una lista de instrucciones, donde la cantidad de veces que se ejecuta la lista de instrucciones está determinada por una condición lógica que determina cuándo la instrucción se ejecutará repetidamente y cuándo no se ejecutará.
Cada lista de instrucciones ejecutadas, llamada iteración o bucle, tiene tres tipos de sentencias repetitivas: ciclo “Desde... Hasta”, ciclo “Mientras” y ciclo “Repetir... Hasta”.

2.10. Sentencias Repetitivas de ciclo “Desde… Hasta”

Repite sobre una orden o grupo de órdenes comenzando desde una variable con un valor inicial hasta alcanzar otro valor final, con el número de veces especificado. Utilice la sintaxis "for".

```
FOR variable := ValorInicial TO ValorFinal DO
    Sentencia;
```

Ejemplo:

```
program src1;
var
  num,i:Integer;
begin
    write('Ingrese un numero: ');
    readln(num);
    for i:=1 to num do
        write(i,' ');
    readln
end.
```

Donde el valor inicial debe ser menor al valor final, debido a que su valor aumentará en una unidad.

Ejecución:

```
Ingrese un numero: 5
1 2 3 4 5
```

Existen casos donde los bucles “for” pueden enlazar uno dentro de otros.

Ejemplo:

```
program src2;
var
  tabla,num:Integer;
begin
     for tabla:=1 to 5 do
         for num :=1 to 10 do
           writeln(tabla,' por ', num, ' es ', tabla*num);
     readln
end.
```

Ejecución:

```
1 por 1 es 1
1 por 2 es 2
1 por 3 es 3
1 por 4 es 4
1 por 5 es 5
1 por 6 es 6
1 por 7 es 7
1 por 8 es 8
1 por 9 es 9
1 por 10 es 10
2 por 1 es 2
```

...

```
4 por 10 es 40
5 por 1 es 5
5 por 2 es 10
5 por 3 es 15
5 por 4 es 20
5 por 5 es 25
5 por 6 es 30
5 por 7 es 35
5 por 8 es 40
5 por 9 es 45
5 por 10 es 50
```

Si se requiere repetir más de una orden, también deberán de ser encerradas con las instrucciones "begin" y "end", convirtiendo las órdenes en una sentencia compuesta.

Ejemplo:

```
program src3;
var
  tabla,num:Integer;
begin
     for tabla:=1 to 5 do
     begin
          for num :=1 to 10 do
            writeln(tabla,' por ', num, ' es ', tabla*num);
          writeln; (*Linea en blanco*)
     end;
     readln
end.
```

Ejemplo de ejecución:

```
1 por 1 es 1
1 por 2 es 2
1 por 3 es 3
1 por 4 es 4
1 por 5 es 5
1 por 6 es 6
1 por 7 es 7
1 por 8 es 8
1 por 9 es 9
1 por 10 es 10

2 por 1 es 2
2 por 2 es 4
...
4 por 9 es 36
4 por 10 es 40

5 por 1 es 5
5 por 2 es 10
5 por 3 es 15
5 por 4 es 20
5 por 5 es 25
5 por 6 es 30
5 por 7 es 35
5 por 8 es 40
5 por 9 es 45
5 por 10 es 50
```

2.11. Sentencias Repetitivas de ciclo "Mientras"

Repite una sentencia mientras se cumple una condición, siendo útil cuando no se sabe la cantidad de veces que se debe de repetir un bloque. Usa la sintaxis "While… do".

```
WHILE condicion DO
      sentencia;
```

Requiere de un bloque "begin" y "end" que contenga la sentencia.

Ejemplo:

```
program buclewhile;
var
  numero: integer;
begin
  write('Ingrese un numero menor a 10: ');
  readln(numero);
  if (numero<10) then
  begin
       while (numero<=10) do
           begin
                write(numero,' ');
                numero := numero + 1;
           end;
  end;
  readln
end.
```

El ciclo while seguirá repitiendo mientras la condición sea verdadera, si llegara a ser falsa, sale del bucle.

Ejecución:

```
Ingrese un numero menor a 10: 5
5 6 7 8 9 10
```

2.12. Sentencias Repetitivas de ciclo "Repetir… Hasta"

Repite un conjunto de declaraciones hasta que se cumple una condición y garantiza que la lista de sentencias se ejecute al menos una vez debido a que la comprobación se ejecuta al final. Usa la sintaxis "Repeat… Until".

```
REPEAT
      sentencia 1;
      sentencia 2;
      (* ... *)
      sentencia N;
UNTIL condicion;
```

Como ejecuta un grupo de sentencias, no necesita de un "begin" y "end", siendo una condición opuesta al ciclo "While".

Codificación:

```
program buclerepeat;
var
  ClaveCorrecta, Intento: Integer;
begin
  ClaveCorrecta := 123;
  repeat
    writeln('Introduce la clave de acceso: ');
    readln(Intento);
  until Intento = ClaveCorrecta;
  writeln('Clave Correcta');
  (*Aquí iría el resto del programa*)
  readln
end.
```

Ejemplo de ejecución:

```
Introduce la clave de acceso:
654
Introduce la clave de acceso:
123
Clave Correcta
```

En pascal-FC, nos permite repetir indefinidamente una sentencia a través de la sintaxis:

```
program demoforeverepetir;
var
    i: integer;

begin
    i := 0;
    repeat
        i := i + 1;
        writeln (i);
    forever;

end.
```

3. Casos Prácticos

3.1. Caso 1:

Escribir un programa que detecte el nombre introducido por el usuario.

3.2. Caso 2:

Escribir un programa que determine si un número leído desde el teclado es múltiplo de 5.

3.3. Caso 3:

Escribir un programa que dado un número del 1 al 12 escriba el correspondiente nombre del mes del año.

3.4. Caso 4:

Escribir un programa que detecte si un número leído desde el teclado es mayor o menor que 100.

3.5. Caso 5:

Escribir un programa que dado un número del 1 al 7 escriba el correspondiente nombre del día de la semana.

3.6. Caso 6:

En una tienda se venden productos de primera necesidad a los cuales se les aplica un descuento del 10% de la compra total, si esta es igual o mayor a S/.150. Escriba un programa que, a partir del importe total de la compra muestre lo que debe pagar el cliente.

3.7. Caso 7:

Escribir un programa que una vez leída una hora en formato (horas, minutos, segundos) indique cuál será el tiempo en segundos.

3.8. Caso 8:

Escribir un programa que detecte si se han introducido en orden decreciente tres números introducidos por el usuario.

3.9. Caso 9:

Escribir un programa que visualice en pantalla los números múltiplos de 7 comprendidos entre 1 y 100

3.10. Caso 10:

Realizar un programa de cajero automático realizando las siguientes operaciones:

a. Debe iniciar con 1000 soles en la cuenta.
b. Debe solicitar y definir si la clave ingresada es correcta.
c. Debe desplegar un menú de opciones (extraer dinero, depositar dinero, transferir dinero, consultar saldo)
d. Dada la opción seleccionada por el usuario debe ejecutar la operación.
e. Debe preguntar al usuario si desea ver el resumen de la operación (el valor en soles que queda en la cuenta luego de ejecutar la operación).
f. Debe preguntar al usuario si desea realizar otra operación.

4. Referencias

Berlanga Llavori, R. (2000). Introducción a la programación con Pascal. España: Universitat Jaume I.

Gonzáles, T., Losada, I., Martínez, R., Nava, F., Pantrigo, J., Paredes, M., & Sanz, A. (2005). Introducción a la programación: problemas resueltos en Pascal.