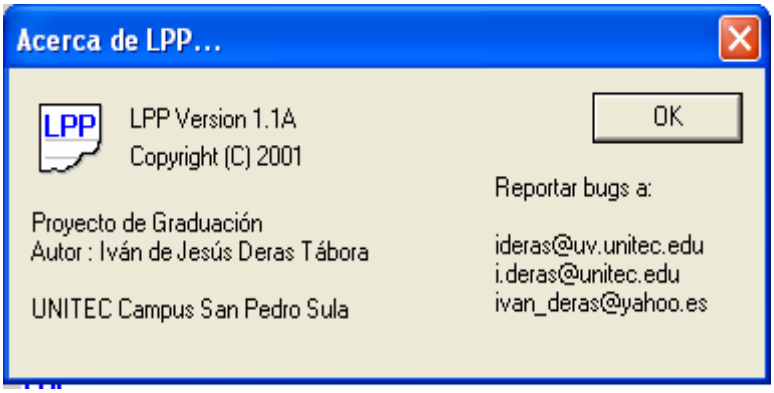
Programación en :

LPP

Por Lic. Rommel Castillo Suazo



Índice

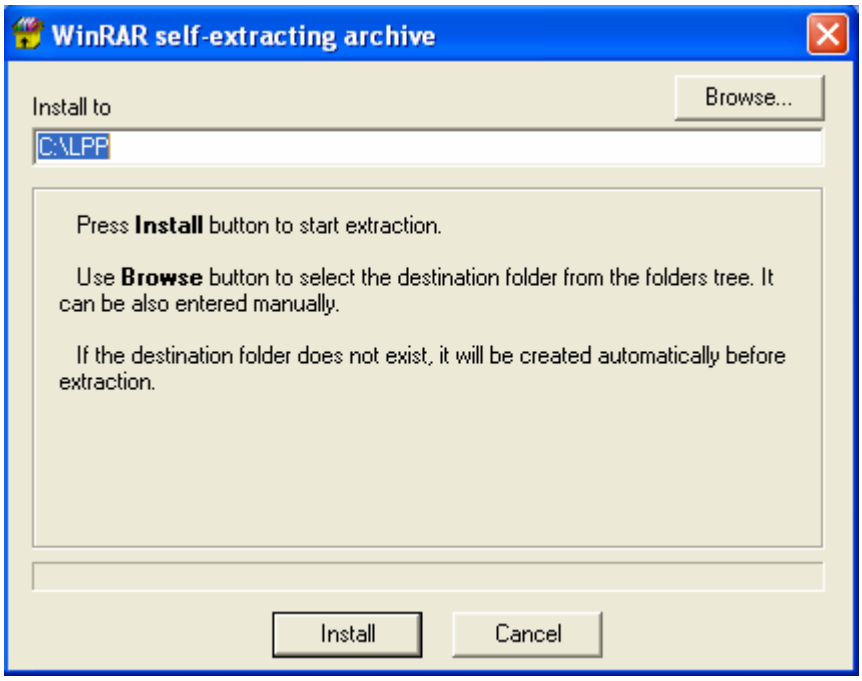
Qué es LPP?

Este lenguaje de programación fue creado como proyecto de graduación del Ingeniero Iván Deras.

Lpp es un lenguaje de programación para principiantes, el cual fue diseñado con la idea de facilitar el proceso de enseñanza-aprendizaje de un leguaje de programación en nuestro idioma, este contiene la mayoría de instrucciones que tienen los lenguajes de programación .

Instalación

Abrir el archivo " LPP_Instalador " el cual será proporcionado por la Universidad , luego nos presenta una pantalla que nos pide en que unidad queremos instalar el lpp



Luego presionamos instalar .

Abrir lpp

Para abrir lpp damos u clic en star , luego programs ,luego lpp y seleccionamos lpp y nos abre el programa.

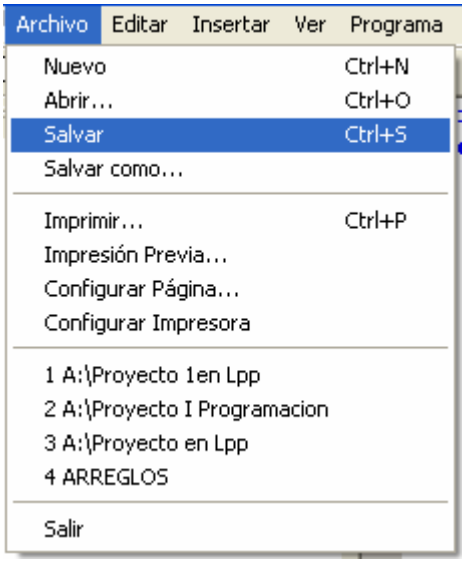
Escribir mi primer programa

Una vez que hemos abierto Lpp , este nos presenta una página en blanco , como lo hace Word , en ella escribiremos nuestro primer programa :

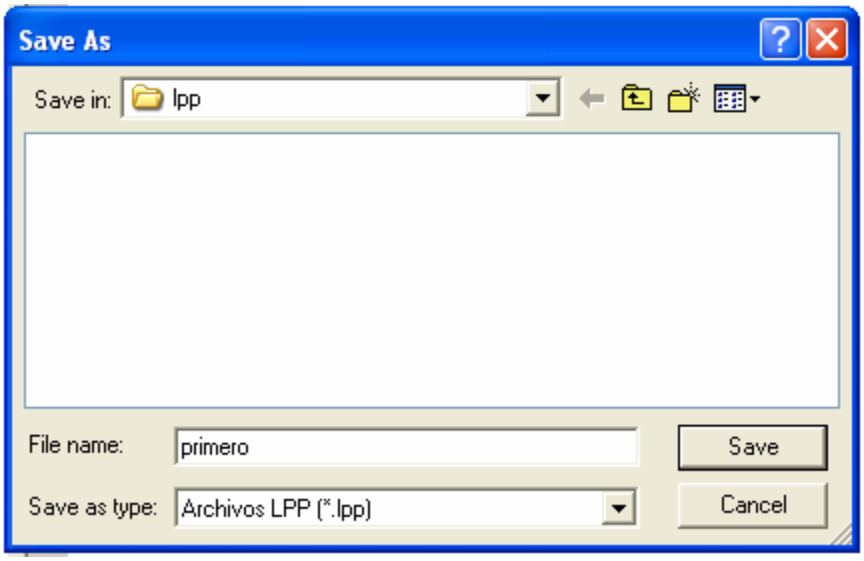
```
inicio
    escriba "Unitec"

Fin
```

Luego lo guardamos



Escribimos el nombre del programa en la ventana que nos aparece y luego presionamos salvar.

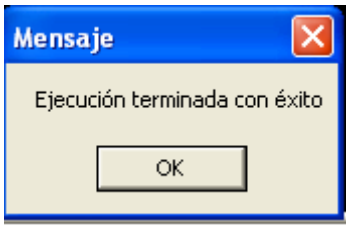


Ahora que los hemos salvado necesitamos , que nuestro programa funcione y escriba en la pantalla Unitec,

Primero lo compilamos, esto con el fin de encontrar errores, seleccionamos del menú la opción de programa, luego compilar , si tuviéramos errores el nos seleccionaría la frase donde se encuentre el error, luego lo corregimos y lo volvemos a compilar , hasta que no aparezca nada seleccionado.

Luego que el programa no tiene errores, seleccionamos programa, luego ejecutar, y en la pantalla aparecerá la palabra Unitec que es la salida del programa, también para ejecutar el programa puede usar el icono de ejecutar :

luego debemos de cerrar la pantalla de la salida del programa dando un click en ok de la ventana mensaje , si esta no aparece la puede buscar en el taksbar de Windows o el grupo del programa lpp si usa Windows xp.



Siempre que queremos escribir un programa en lpp iniciamos con la palabra **Inicio**

Luego escribimos el cuerpo del programa

Fin

Y terminamos con la palabra fin que indica el final del programa.

La palabra reservada **Escriba** *escribe en la pantalla lo que esta encerrado entre comillas .*

Saltar una línea

```
Inicio
      Escriba “Mi primer programa “
      Escriba “ en lpp “
Fin
```

La salida del programa seria

Mi primer programa en lpp

Esto porque el escriba deja en cursor en la misma línea, ahora si quisiéramos escribir :

Mi primer programa

En lpp

El programa seria de esta forma

ejemplo

```
Inicio
      Escriba “Mi primer programa “
      Llamar nueva_linea
      Escriba “ en lpp “
Fin
```

Con esto deducimos que la instrucción llamar Nueva_linea hace saltar una línea y el contenido del próximo escriba se escribe en la siguiente línea.

Declarar variables

Siempre que necesitemos hacer un programa, tendremos que declarar variables para poder guardar la información que introduzcamos al programa.

Los tipos de datos básico soportados son los siguientes :

1. Entero : solo números enteros.
2. Real : números con cifras decimales.
3. Caracter : cuando queremos guardar un solo carácter.
4. Booleano : cuando necesitamos guardar una expresión lógica (verdadero o falso)
5. Cadena : cuando queremos guardar letras .

Ejemplos

Si queremos declarar una variable de tipo entero seria así :

Entero numero

Numero se convierte en una variable de tipo entero

Si queremos declarar una variable para guardar el nombre seria así :

Cadena [25] nombre

Nombre seria una variable que guardaría solo 25 caracteres aunque usted puede escribir mas de 25 letras el en la memoria solo guardara los primeros 25 caracteres..

Operadores

LPP proporciona los siguientes operadores:

Operador	Función
()	Agrupar expresiones
^	Operador para exponenciación
*	Operador de multiplicación
/	Operador de división
mod	Operador de cáculo de residuo
div	Operador de división entera
y	Operador lógica y
+	Operador de suma
-	Operador de Resta
o	Operador lógico o

Leer valores y almacenarlos en las variables

Cuando nosotros queremos leer un valor y almacenarlo en una variables usaremos la palabra ***lea < variable>*** . y cuando queremos asignar un valor o una operación matemática usaremos ← que es el símbolo de < mas - .

Ejemplo sobre lectura de datos

```
Cadena [25] nombre
Inicio
      Escriba “Ingrese su nombre “
      Lea nombre
      Escriba  “Bienvenido “
      Escriba nombre
Fin
```

El programa declara una variable para el nombre , que guarda 25 caracteres máximo , ingresa el nombre y luego escribe en la pantalla Bienvenido el nombre

que se ingreso. Algo importante es que cuando se quiere presentar el valor de la variable esta no se escribe entre comillas.

Ela tabla se nos muestra como se pudo sustituir un bloque del programa que nos daría el mismo resultado

Caso 1	Caso 2
Escriba "Bienvenido " Escriba nombre	Escriba "bienvenido " , nombre

Asignaciones y Operaciones matemáticas en un programa.

El simbolp ← lo usaremos para asignar valores a las variables ejemplo

Sueldo←500 Con esta instrucción estamos asignando el valor de 500 a la variables sueldo que pudo declararse como entero o real

Nombre←"juan" con esta instrucción asignamos la cadena "Juan " a la variable nombre que es una variable de tipo cadena

Ejemplo sobre asignaciones de valores a las variables

```
Entero sueldo, aumento
Cadena[25] nombre
Inicio
      Escriba "Ingresar el nombre del empleado"
      Lea nombre
      Escriba "Ingresar el sueldo del empleado"
      Lea sueldo
      Aumento ← sueldo * 1.25
      Escriba "Nuevo sueldo con el 25% de aumento"
      Escriba aumento

Fin
```

El programa pide el nombre y el sueldo del empleado luego calcula el 25% de aumento de sueldo y lo guarda el la variable aumento y luego presenta el nuevo sueldo.

Ejemplo sobre suma de cadenas

```
Cadena[25] nombre,apellido,completo
Inicio
      Escriba " Su Nombre"
      Lea nombre
      Escriba " Apellido "
      Lea apellido
      Completo ← nombre + " " + apellido
      Escriba "Nombre completo" , completo
Fin
```

La variable completo toma el valor del nombre mas un espacio en blanco mas el apellido y lo guardamos en una variable donde ahora tenemos el nombre y el apellido.

Instrucciones condicionales

Anteriormente hemos estado haciendo programas que solo hacen cálculos, pero la programación es mas interesando cuando nuestros programas toman sus propias decisiones en LPP existen intrucciones condicionales que se describen a continuación :

Instrucción si:

Sintaxis

si condición entonces
    instrucciones
fin si

ó

si condición entonces
    instrucciones
sino
    instrucciones
fin si

Ejemplo sobre decisiones

Ingresar un numero y si el numero es mayor a 100 , escribir en la pantalla el numero es mayor a 100.

```
Entero num
Inicio
      Escriba “Ingresar un numero”
      Lea num
      Si num > 100 entonces
                  Escriba “El numero es mayor a 100”
      Fin Si
Fin
```

En programa solo imprimirá que el número fue mayor a 100 cuando cumpla con la condición ***num > 100*** sino cumple con la condición no hace nada .

Ejemplo sobre decisiones

Ingresar el nombre del empleado, las horas trabajadas, luego Calcular pago bruto (50 lps la hora) IHSS y total a pagar , presentar los resultado del programa

Nota : *el seguro social es 84 si el sueldo es mayor 2400 sino es el 3.5% del sueldo del empleado*

```
Entero horas
Real  Pbruto,ihss,tp
Cadena [25] nombre
Inicio
      Escriba “Ingresar el nombre”
      Lea nombre
      Escriba “Ingresar las horas trabajadas”
      Lea horas
      Pbruto←horas*50
      Si pbruto > 2400 entonces
                  Ihss← 84
            Sino
                  Ihss←0.035*pbruto
      Fin si
      Tp←pbruto-ihss
      Escriba “Pago bruto     ” , pbruto
      Llamar Nueva_linea
      Escriba “Seguro Social ” , ihss
      Llamar Nueva_linea
      Escriba “Total a pagar  ” , tp
      Llamar Nueva_linea
Fin
```

En este programa se uso en el calculo del ihss una decisión que tiene dos salidas una cuando se cumple la condición que es el entonces y la otra cuando no se cumple la condición que es el sino , ahora esto nos ayuda a que nuestros programas puedan tomar una decisión cuando la condición se cumple y otra cuando no se cumple.

Ahora en el siguiente ejercicio que se presenta , ya no hay dos soluciones a la condición hay tres soluciones , cuando sucede esto se usan condiciones anidadas.

Sintaxis de una condición anidada :

```
Si condición 1 entonces
                Instrucciones
          Sino si condición 2  entonces
                          Instrucciones
                    Sino si condición 2  entonces
                                          Instrucciones
                                    Sino
                                          Instrucciones
                    Fin si
          Fin si
  Fin si
```

Ejemplo sobre decisiones anidadas

Ingresar el nombre del empleado, la zona de trabajo , las ventas del empleado , luego calcular su comisión en base a un porcentaje basado en la zona de trabajo, luego determinar el IHSS y el total a pagar , presentar los datos.

Tabla para el caculo de la comisión

Zona	Porcentaje de Comisión
A	6%
B	8%
C	9%

```
caracter zona
cadena[25] nombre
real ventas , comis , ihss, tp

inicio
      escriba "Ingresar el nombre del empleado   "
      lea nombre
      escriba "Ingresar las ventas  del empleado "
      lea ventas
      escriba "Ingresar la zona de trabajo       "
      lea zona
      si zona ='A' entonces
                              comis← 0.06 * ventas
                        sino si zona='B' entonces
                                    comis← 0.08 * ventas
                              sino si zona='C' entonces
                                          comis← 0.09 * ventas
                                    sino
                                          comis← 0
                              fin si
                        fin si
      fin si
      si comis > 2400 entonces
                              ihss ← 84
                        sino
                              ihss←0.035*comis
```

```
      fin si
      tp←comis - ihss

      Escriba " Comsión ganada " , comis
      llamar nueva_linea
      Escriba " Segudo Social " , ihss
      llamar nueva_linea
      Escriba "Total a pagar " , tp
      llamar nueva_linea
fin
```

En este programa usamos decisiones anidadas para el calculo de la comisión del empleado , esto porque se tenían varias opciones de la cuales elegir .
El ultimo sino donde la comisión es 0 se hace porque no estamos seguros de que la persona que opera el programa introduzca correctamente la zona , si se ingreso otra zona de las permitidas la comisión es cero.

Estructura Caso

Esta se usa como sustituto en algunos casos del si anidado , por ser mas practico al aplicarlo en la evaluación de algunas condiciones.

Sintaxis

```
caso variable
        valor1, valor2, valor3, … :
                instrucciones
        valor1, valor2, valor3, … :
                instrucciones
                .
                .
        [ sino :
                instrucciones]
        fin caso
```

Los valores a evaluar , se separan por comas si hay varios, tal como aparece en la sintaxis valor1,valor2 etc, también se puede usar el sino que nos indica, que en caso de no seleccionar ninguna de las instrucciones anteriores se ejecutan.

Ejemplo sobre la aplicación de la estructura caso

En el ejercicio anterior usamos decisiones anidadas para determinar la comisión , ahora usaremos una estructura caso.

```
caracter zona
cadena[25] nombre
real ventas , comis , ihss, tp
inicio
      escriba "Ingresar el nombre del empleado   "
      lea nombre
      escriba "Ingresar las ventas  del empleado "
      lea ventas
      escriba "Ingresar la zona de trabajo       "
      lea zona
      caso Zona
            ‘a’,’A’ :   comis← 0.06 * ventas
            ‘b’,’B’ :   comis← 0.08 * ventas
            ‘c’,’C’ :   comis← 0.09 * ventas
            sino :
                 comis← 0
       fin caso
      si comis > 2400 entonces
                              ihss ← 84
                        sino
                              ihss←0.035*comis
      fin si
      tp←comis - ihss
      Escriba " Comsión ganada " , comis
       llamar nueva_linea
      Escriba " Segudo Social " , ihss
      llamar nueva_linea
      Escriba "Total a pagar " , tp
      llamar nueva_linea
fin
```

Ahora nuestro programa reconoce las mayúsculas y minúsculas en la evaluación de la zona.

Uso del operador O

El operador O se utiliza cuando estamos evaluando dos o mas condiciones y queremos que la condición se cumpla cuando una de las condiciones que estamos evaluando se hacen verdadera. Ejemplo

Cuando se introduce la zona en el ejercicio con la estructura Si solo evaluavamos una opción que la zona sea igual a la letra A y si el usuario escribía una a minúscula no se tomaba en cuenta pero esto se puede corregir de esta manera :

```
si (zona ='A')  o  (zona ='a')  entonces
                          comis← 0.06 * ventas
                sino si (zona='B') o (zona='b')  entonces
                                    comis← 0.08 * ventas
                     sino si (zona='C') o (zona='c') entonces
                                          comis← 0.09 * ventas
                                    sino
                                          comis← 0
                               fin si
                     fin si
     fin si
```

Ahora la condición dice, ***si zona es igual a la letra A o es igual a la letra a*** , cualquiera que sea la zona a o A en ambos casos la condición es verdadera , ahora el usuario puede usar mayúsculas y minúsculas y el resultado será el mismo.

Ejemplo sobre el operador O

Ingresar el nombre del cliente , luego la cantidad del producto, precio y tipo de cliente , calcular el subtotal , descuento , impuesto s/v, total a pagar, presentar los datos.

El descuento es del 10% si el cliente es de tipo A o la cantidad de cualquier producto es mayor a 100 sino es de 5%.

```
Real precio,st,des,tp,isv
Cadena[25] nombre
Caracter tipoM
Entero cant
Inicio
      Escriba "Nombre del cliente"
      Lea nombre
      Escriba "Ingresar el Tipo de cliente"
      Lea tipoM
      Escriba "Ingresar el precio del producto"
      Lea precio
      Escriba "Ingresar la cantidad "
      Lea cant
      St← precio*cant
      Si (tipoM ='a') o (tipoM='A' ) o  (cant>100)  entonces
                                              Des←st*0.10
                                        Sino
                                              Des←st*0.05
      Fin si
      Isv←(st-des)     *0.12
      Tp←(st-des)+isv
      Escriba  "Subtotal ", st
      Llamar nueva_linea
      Escriba "Descuento ", des
      Llamar nueva_linea
      Escriba "Impuesto ", isv
      Llamar nueva_linea
      Escriba "Total a pagar" ,tp
fin
```

Uso del operador Y

El operador Y se utiliza cuando estamos evaluando dos o mas condiciones y queremos que la condición se cumpla cuando las dos condiciones que estamos evaluando se hacen verdadera. Ejemplo

Ejemplo sobre el operador O

Se ingresa un numero y se desea saber si el numero esta entre 50 y 100.

```
entero num
inicio
      escriba "Numero a evaluar"
      lea num
      si (num >=50 ) y (num<=100) entonces
                  Escriba " el numero esta entre 50 y 100"
            sino
                  Escriba " Fuera del rango 50 y 100"
      fin si
fin
```

Instrucciones de ciclo

Hemos hecho programas que solo se repiten una vez , pero en la programación necesitamos que los programas corran varias veces y que nos presenten información al final de correr varias veces, en estos casos usaremos ciclos, que son estructuras de repetición, que se repiten hasta cumplir con una condición o simplemente indicamos cuantas veces se van a repetir.

Ciclo mientras:

Sintaxis

mientras condición haga
    instruciones
fin mientras

El ciclo mientras se utiliza cuando se quiere ejecutar repetidamente un bloque instrucciones basado en una condición, el ciclo se repite mientras la condición se cumple.

Ejemplo sobre el ciclo Mientras usando un contador

Ingresar 10 nombres

```
Entero contador
Cadena[25] nombre
Inicio
      Contador←0
      Mientras contador<10 haga
            Escriba “Ingresar el nombre”
            Lea nombre
            Contador← contador + 1
      Fin mientras
Fin
```

En este programa introducimos el concepto de contador , que es una variable que se incrementa su valor en 1 y de esta manera contamos cuantos nombres

se van ingresando para parar cuando ingresemos 10 , esto nos dice que la condición ya no se cumple porque cuando el contador vale 10 la condición de contador < 10 ya no se cumple porque es igual y el ciclo termina.

Ejemplo sobre el ciclo Mientras usando acumuladores

Ingresar 10 números y al final presentar la suma de los números.

```
Entero contador,suma,num
Inicio
      Contador←0
      Suma←0
      Mientras  contador<10 haga
            Escriba “Ingresar un numero”
            Lea num
            contador← contador +1
            Suma←num+suma
      Fin mientras
      Escriba “Suma de los 10 números  “, suma
Fin
```

El ciclo recorre 10 veces y pide los 10 números, pero la línea *suma*←*suma* + *num* ,hace que la variable suma, incremente su valor con el numero que se introduce en ese momento , a diferencia del contador, un acumulador se incrementa con una variable , acumulando su valor hasta que el ciclo termine , al final se presenta la suma, solo en ese momento se debe de presentar un acumulador, porque antes no reflejaría la suma de todos los números.

Siempre que usemos un contador o acumulador debemos darle un valor inicial de generalmente será 0.

Ejemplo sobre el ciclo mientras usando una respuesta para contralor la salida del ciclo.

Ingresar el nombre del cliente , el precio del producto, cantidad y luego calcular el subtotal , isv y total a pagar , presentar los datos luego preguntar si desea continuar , al final presentar el monto global de la factura.

```
Caracter Resp
Cadena[25] nombre
Real Precio, cantidad,  totalglobal, st, isv, tp

Inicio
      Totalglobal←0
      Resp←’S’
      Mientras resp<>’N’ haga
            Escriba “Nombre del cliente”
            Lea nombre
            Escriba “Ingresar  la cantidad del producto ”
            Lea cantidad
            Escriba “Ingresar el precio de producto      “
            Lea precio
            St← precio * cantidad
            Isv←st * 0.012
            Tp←st-isv
            Totalglobal←totalglobal+st
            escriba "Subtotal " , st
            llamar Nueva_linea
            escriba "Impuesto sobre venta " , isv
            llamar Nueva_linea
            escriba "Total a pagar " , tp
            llamar Nueva_linea
            Escriba "Desea continuar S/N"
            Lea Resp
      Fin mientras
      Escriba “Total de la venta” , totalglobal
fin
```

En este ejercicio , observamos que el ciclo lo controla una respuesta que se pide al final S para seguir o N para terminar , pero daría el mismo resultado si escribe

cualquier letra distinta a S , arenque no sea N siempre seguiría funcionando el programa , la validación de los datos de entrada lo estudiaremos mas adelante.

Ejemplo sobre estructuras de condición dentro del ciclo Mientras.

Ingresar el nombre del alumno, la nota examen y nota acumulada, luego calcular la nota final, y presentar la nota final y la observación del alumno.
Preguntar si desea continuar, al final presentar el numero de aprobados y reprobados.

```
Caracter Resp
Cadena[25] nombre
Real na,ne,nf
entero cr,ca

Inicio
      cr<-0
      ca<-0
      Resp<-'S'
      Mientras resp<>'N' haga
            Escriba "Nombre del alumno"
            Lea nombre
            Escriba "Nota acumulada "
            Lea na
            Escriba "nota examen "
            Lea ne
            nf<- na+ne

            si nf >= 60 entonces
                        escriba "Usted esta Aprobado"
                        ca<-ca+1
                  sino
                        escriba "Usted esta Reprobado"
                        cr<-cr+1
            fin si
```

```
            llamar Nueva_linea
            escriba "Nota final " , nf
            llamar Nueva_linea
            Escriba "Desea continuar S/N"
            Lea Resp
      Fin mientras
      llamar Nueva_linea
      Escriba "Total de reprobados" , cr
      llamar Nueva_linea
      Escriba "Total de aprobados" , ca

Fin
```

Como podemos observar en las líneas del programa, usamos dentro del ciclo mientras, decisiones para poder contar los reprobados y aprobados que resulten del ingreso de los alumnos, si la nota es mayor a 60 escribe aprobado e incrementa el contador y sino hace lo contrario, escribir reprobado e incrementar el contador.

Ciclo para

Sintaxis

para variable <- valor_inicial hasta valor_final haga

instrucciones

fin para

Descripción

El ciclo Para se utiliza generalmente para ejecutar un conjunto de instrucciones que se repiten un número de veces, establecido antes de ejecutar el ciclo.

Variable : es de tipo entero

Valor_inicial : este puede se un numero entero o una variable entera.

Valor_final : este puede se un numero entero o una variable entera.

Ejemplo : presentar los números del 1 al 10 en la pantalla.

```
Entero I
Inicio
      Para I←1 hasta 10 haga
            Escriba I
            Llamar nueva_linea
      Fin para
Fin
```

El programa el ciclo para establece el numero de veces que se repetirá el ciclo indicando ***1 hasta 10*** luego la variable I toma el valor 1 a 10 según el ciclo se va ejecutando, es por eso que al escribir la I la primera vez escribe 1 la segunda vez 2 y así hasta llegar al final que es 10.

Ejemplo : sobre el uso de variables en el rango del ciclo para.

```
Entero I, final
Inicio
      Escriba “Ingresar el numero de veces a repetir el ciclo “
      Lea final
      Para I←1 hasta final  haga
            Escriba I
            Llamar nueva_linea
      Fin para
Fin
```

Ahora el programa se vuelve mas dinámico, nosotros podemos indicar el numero de veces que se repetirá el ciclo, usando una variable entera para indicar el final del ciclo.

Ejemplo uso del ciclo Para , en el calculo del factorial de un numero.

```
Entero I, numero, factorial
Inicio
      Factorial←1
      Escriba “Ingresar el numero  para determinar su factorial “
      Lea numero
      Para I←1 hasta numero  haga
            Factorial← factorial * I
      Fin para
      Escriba “ factorial de “ , numero , “ es “, factorial
Fin
```

En este ejercicio se inicia el factorial en 1 porque acumulara una multiplicación y si lo iniciamos en cero nos daría el resultado cero, si nosotros ingresar 3, el ciclo se ejecutara 3 veces , el factorial tomaría el valor de 1x2x3.

Ciclos anidados

Cuando un ciclo se encuentra dentro de otro ciclo se le llama ciclo anidado.

Ejemplo de un ciclo anidado

Producir la siguiente salida en la pantalla

11111

22222

33333

44444

```
entero I,k
Inicio
      Para I ← 1 hasta  4 haga
            Para K ←1 hasta 5 haga
                        Escriba I
            Fin para
            Llamar nueva_linea
      Fin para
Fin
```

Cuando usamos ciclos anidados, las variables para manejar los ciclos para deben de ser diferentes pues cada una de ellas toma un valor diferente, en este ejercicio necesitamos que se haga 5 veces el ciclo que esta dentro , que es el que presenta 4 veces el valor de la I , luego salta una línea , para que aparezcan los grupos de números en cada línea.

Ejemplo de un ciclo anidado

Ingresar 5 números y calcular el factorial para c/u de los números.

En este ejercicio necesitamos ingresar 5 números pero cada vez que ingresemos un numero debemos de calcular el factorial , entonces necesitaremos una variable para el caculo del factorial, que forzosamente tiene que inicializarse en 1 cada vez que el ciclo que calcula el factorial inicie, des esta manera estaremos seguro que la variable no ha acumulado el valor del factorial anterior.

Ahora con lo anterior deducimos que necesitamos un ciclo para pedir los datos y otro para calcular el factorial .

```
entero I,k,fac,num
Inicio
      Para I ← 1 hasta  5 haga
            escriba " ingresar un numero "
            lea Num
            fac←1
            Para K ←1 hasta num haga
                        fac←fac*K
            Fin para
            escriba "factorial de ", num , " es ",fac
            llamar nueva_linea
      Fin para
Fin
```

Ciclo Repita

Sintaxis:

Repita

Instrucciones

Hasta condición

Descripción

El ciclo repita es lo contrario al ciclo mientras, en éste la ejecución se lleva a cabo hasta que se cumple la condición impuesta.

La condición no se verifica al inicio como el en ciclo mientras se verifica al final .

Ejemplo del repita

Ingresar el nombre del alumno, la nota , luego preguntar si desea continuar , al final presentar el numero de aprobados y reprobados.

```
caracter resp
real nota
entero ca,cr
cadena[25] nombre

inicio
      ca← 0
      cr←0
      repita
            Escriba "ingresar el nombre del alumno "
            lea nombre
            Escriba  "ingresar la nota del alumno  "
            lea nota

            si nota >= 60 entonces
                        ca← ca+1
                  sino
                        cr←cr+1
            fin si
```

```
            escriba " Desea continuar S/N"
            lea resp
      hasta (resp='n') o (resp='N')

      escriba " Aprobados ",ca
      llamar nueva_linea
      escriba " Reprobados ",cr
fin
```

si comparamos este programa con los hechos con el ciclo mientras notaremos que la variable Resp le damos un valor inicial de ‘S’ , para que sea distinta de N , ya que la condición se verifica al inicio del ciclo , pero ahora con el ciclo repita ya no es necesario pues el primer valor de resp lo toma dentro del ciclo , que es la pregunta que hacemos si desea continuar ,y luego verificamos la condición.

Algo importante del ciclo repita es que el ciclo se ejecuta por lo menos una vez , antes de validar la condición de salida del ciclo , es por esto , que siempre que escribamos un programa que verifique la condición antes de entrar ciclo se debe de usar el ciclo Mientras.

El programa anterior no es la versión final, puesto que debemos hacer que el usuario solo ingrese S o N cuando responda si desea continuar , esto nos lleva a escribir un ciclo repita dentro del ciclo repita , para pedir la respuesta y hacer que se salga del ciclo solo cuando responda S o N , de esta manera estaremos seguros de que la repuesta es correcta.

```
caracter resp
real nota
entero ca,cr
cadena[25] nombre

inicio
      ca←0
      cr←0
      repitA
            Escriba "ingresar el nombre del alumno "
            lea nombre
            Escriba  "ingresar la nota del alumno  "
            lea nota

            si nota >= 60 entonces
                              ca←ca+1
                        sino
                              cr←cr+1
            fin si
            repita
                  escriba " Desea continuar S/N"
                  lea resp
            hasta (resp='N') o (resp='S')
      hasta (resp='N')
      escriba " Aprobados ",ca
      llamar nueva_linea
      escriba " Reprobados ",cr
fin
```

Procedimientos

Un procedimiento es un subprograma que realiza una tarea específica y que puede se definido mediante 0 , 1 o mas parámetros . Tanto en entrada de información al procedimiento como la devolución de resultados desde el subprograma se realizad mediante parámetros, el cual nos sirve para introducir o modificar información del programa principal.

Sintaxis

```
Procedimiento  nombre_del_procedimiento [( parámetros ) ]
      [variables locales]
inicio
      instrucciones
fin
```

como se puede observar la sintaxis de los procedimientos es bastante parecida a la de las funciones solo que estos se declaran con la palabra procedimiento y no tienen tipo de retorno.

Siempre que usemos parámetros estos deben de ser del mismo tipo datos, esto nos dice que la variable del programa , debe de del mismo tipo del parámetro del procedimiento y pasados en el mismo orden en que estan colocados en el procedimiento .

Ejemplo : elaborar un procedimiento que presente 5 asteriscos en una línea horizontal.

```
cadena[25] nombre

procedimiento asteriscos
        entero I
inicio
      para i <- 1 hasta 5 haga
            escriba "*"
      fin para
fin

inicio
      escriba "Ingresar el nombre ..:"
      lea nombre
      llamar asteriscos
      llamar nueva_linea
      escriba nombre
      llamar nueva_linea
      llamar asteriscos
fin
```

En este programa usamos un subprograma (procedimiento) para imprimir 5 asteriscos , si no lo hubiéramos hecho de esta manera donde se encuentra la instrucción Llamar asteriscos tendríamos que escribir el ciclo , y lo haríamos dos veces , de la forma en que lo escribimos es mas estructurado, pues se divide ese proceso en un subprograma, que cuando necesitamos una línea de 5 asteriscos solo llamamos el procedimiento .

Nota : los procedimientos se llaman con la instrucción **Llamar.**

Ahora en el programa anterior usa un procedimiento estático, siempre escribirá 5 asteriscos, ahora lo podemos hacer dinámico usando parámetros para indicar cuantos asteriscos queremos presentar en la línea.

Parámetros de valor

Este tipo de parámetro se le conoce con el nombre de ***parámetro e valor*** , que esta por omisión , este tipo de parámetros aunque durante el procedimiento su valor cambie el valor no será asignado a la variable del programa principal , por ejemplo si la variable numero del programa que presentamos abajo se le asigna otro valor diferente al 10 , este cambio no se reflejaría en la variable num , y por esto en el programa principal , es este tipo de parámetros que se le conoce como parámetros de valor.

Ejemplo procedimientos con parámetros de entrada o valor

```
cadena[25] nombre
entero num
procedimiento asteriscos(Entero numero)
entero I
inicio

      para i <- 1 hasta numero haga
            escriba "*"
      fin para

fin

inicio
      num←10
      escriba "Ingresar el nombre ..:"
      lea nombre
      llamar asteriscos(10)
      llamar nueva_linea
      escriba nombre
      llamar nueva_linea
      llamar asteriscos(num)
fin
```

En la línea ***llamar asteriscos(10)*** estamos asignando al parámetro numero de asteriscos el valor de 10 , esto hace que el ciclo recorra 10 veces, luego mas abajo del programa en la instrucción ***llamar asteriscos(num)*** se paso una variable como parámetro asignando el valor de num a numero , el cual numero en el programa principal tiene un valor de 10 el cual se le asigna a numero en el paso del valor de parámetro .

Parámetros de variable

El siguiente programa , nos enseña el uso de los parámetros de variable o referencia, los cuales se les antepone la palabra reservada VAR para indicar que esa variable será un parámetro de referencia o variable, esto nos indica que cualquier cambio que sufra la variable del procedimiento , la variable del programa principal también lo sufrirá, de esta manera podemos enviar información modificarla y envirar resultados al programa principal.

Ejemplo parámetros de variable o referencia.

Elaborar un programa donde se ingrese el nombre y el apellido usando un procedimiento y luego presentar los datos.

```
cadena[30] nombre,apellido
procedimiento pedir_datos(var cadena[30] nombre,cadena[30] apellido)
inicio
      escriba "Ingresar el nombre "
      lea nombre
      Escriba "Ingresar el apellido"
      lea apellido
fin
inicio
      nombre←"No hay cambios en nombre"
      apellido←"No hay cambios en apellido"
      llamar pedir_datos(nombre,apellido)
      escriba "Nombre completo ",nombre," ",apellido
fin
```

En el programa anterior, se inician las variables de apellido y nombre, luego se pasan como parámetros al procedimiento, el nombre como variable y el apellido como valor luego escribimos los valores y solo el nombre presentara el cambio que sufrió en el procedimiento y el apellido seguirá escribiendo el mismo valor que tenia al empezar el programa esto porque no se paso como parámetro de variable (VAR) sino como de valor y no se le permitió sufrir alguna modificación.

Para mejorar el programa anterior el procedimiento tendría que escribirse así :

```
procedimiento pedir_datos (var cadena[30] nombre, var cadena[30]
apellido)
inicio
      escriba "Ingresar el nombre "
      lea nombre
      Escriba "Ingresar el apellido"
      lea apellido
fin
```

Ejemplo

Ingresar la base y el exponente y luego calcular la potencia.

En este programa usaremos un procedimiento para el ingreso de los datos y otro para calcular la potencia.

```
entero base,expo,pot

procedimiento ingreso(var entero base,var entero expo)
inicio
      escriba "Ingresar la base "
      lea base
      Escriba " Ingresar el exponente "
      lea expo
fin

procedimiento Potencia(entero base , entero expo, var entero potencia)
 entero i
inicio
      potencia←1
      para I ← 1 hasta expo haga
            potencia <- potencia * base
      fin para
fin

inicio
      llamar ingreso(base,expo)
      llamar potencia(base,expo,pot)
      Escriba "Potencia es ", pot
fin
```

En el procedimiento de ingreso los dos datos , exponente y base son de tipo entero y parámetros de variable , esto porque necesitamos que el procedimiento nos devuelva los valores para luego introducirlos en el procedimiento potencia pero aquí, base, expo son de tipo valor , esto porque no necesitamos modificar o leer su valor como anteriormente los hicimos en el procedimiento de ingreso , luego la variable pot si se pasa como parámetro de variable debido a que necesitamos modificar su valor y presentarlo en el programa principal.

Funciones

Las funciones son subprogramas que hacen una o más instrucciones pero que siempre nos devuelven un solo valor .

```
Sintaxis
funcion nombre_funcion [( parámetros  ) ]: tipo_de_retorno
      [variables locales]

inicio

      instrucciones

      Retorne valor

fin
```

Si notamos en la sintaxis de la función observamos que esta la palabra retorno la cual devuelve un valor que tiene que ser del mismo tipo que fue declarado el Tipo_de_retorno.

Ejemplo : calculo de la potencia

Usaremos el mismo ejercicio que usamos para los procedimientos, para hacer una demostración de cómo cambiaria el programa usando una función para el cálculo de la potencia.

```
entero base,expo,pot

funcion Potencia(entero base , entero expo): entero
  entero i, resp
inicio
      resp<-1
      para I <- 1 hasta expo haga
            resp <- resp * base
      fin para
      retorne resp
fin
procedimiento ingreso(var entero base,var entero expo)
inicio
      escriba "Ingresar la base "
      lea base
      Escriba " Ingresar el exponente "
      lea expo
fin

inicio
      llamar ingreso(base,expo)
      pot<-potencia(base,expo)
      Escriba "Potencia es ", pot
fin
```

Si notamos en la función Potencia se declaran una variable ***I*** que es para el ciclo y la otra ***resp*** que es para el caculo de la potencia, la cual usaremos como acumulador de la multiplicación de la potencia, y al final usando ***retorne resp*** , que es lo que nos devuelve el valor , y lo asigna en a la variable pot en el programa principal, cuando usamos la instrucción **pot<-potencia(base,expo).**

En conclusión las funciones siempre nos retornaran un valor que es productos de varios o un solo cálculos, y se devuelve usando la instrucción retorno.

Ejemplo de planilla (Procedimientos y funciones)

Se ingresar el nombre, las ventas y la zona del empleado usando un procedimiento, luego se calcula el la comisión en base a la zona de trabajo , ihss y total a pagar, luego se presentan los datos.

Nota:

- se deberá de usar un procedimiento para los cálculos y la presentación de los datos.
- Usar una función para el calculo del ihss.
- Usar una función para el calculo del la comisión.

Procedimiento de ingreso

En este procedimiento se ingresan los datos , validando que la zona solo sea A,B,C

Procedimiento de cálculo

Se calcula la comisión y ihss usando las funciones declaradas anteriormente , luego el total a pagar, algo que debemos de notar es que las ventas y la zona se pasan como parámetros de valor y las demás ihss, comis y tp como parámetros de variable porque necesitamos modificar su valor

Procedimiento presentar

Presentamos los cálculos y pasamos las variable como parámetros de valor , porque solo los necesitamos presentar.

```
real ventas,comis,ihss,tp
cadena[30] nombre
caracter zona

funcion  seguro(real comis ): real
 real Vihss
inicio
      si comis >2400 entonces
            vihss<- 84
      sino
            vihss<-0.035 * comis
      fin si
      retorne vihss
fin

funcion comision(caracter zona,real ventas) : real
   real vcomis
inicio

      caso zona
            'A' : vcomis<-0.05*ventas
            'B' : vcomis<-0.06*ventas
            'C' : vcomis<-0.09*ventas
      Fin Caso
      retorne vcomis
fin

procedimiento  Ingreso(cadena[30]  nombre,var  caracter  zona  ,  var  real
ventas)
inicio
      escriba "Ingresar el nombre "
      lea nombre
      escriba "Ventas mensuales   "
```

```
      lea ventas
      repita
            escriba "Zona A,B,C "
            lea zona
      hasta (zona ='B') o (zona ='C')     o (zona='A')
fin

procedimiento calculos(caracter zona,real ventas, var real comis, var
real ihss ,var real tp)
inicio
      comis<-comision(zona,ventas)
      ihss<-seguro(comis)
      tp<-comis-ihss
fin

procedimiento presentar(real comis , real  ihss , real tp)
inicio
      escriba " Comisión      ",comis
      llamar Nueva_linea
      escriba " Seguro Social ", ihss
      llamar Nueva_linea
      escriba " Total a pagar ", tp
      llamar Nueva_linea
fin

Inicio
      llamar Ingreso(nombre,zona,ventas)
      llamar calculos(zona,ventas,comis,ihss,tp)
      llamar presentar(comis,ihss,tp)
fin
```

Las funciones las declaremos antes de los procedimientos porque estas se usaran en el procedimiento cálculos, y es mas legible al momento de leer un programa.

Registros

Un registro es un dato estructurado, formado por elementos lógicamente relacionados , que pueden ser del mismo o de distintos tipos , a los que se les denomina campos . Los campos de un registro podrían ser de los tipos previamente definidos por lpp (entero , real etc) o bien por un registro definido anteriormente

Ejemplo: demostración de registros

En este programa usaremos un registro para guardar la información del alumno usando un registro que se llama reg_alumno.

Luego tendremos que declarar una variable que sea del tipo registro, se llama alum, después usaremos esa variable para pedir los datos , siempre que queremos acceder a un registro se hace

Registro.Variable

Entonces si queremos acceder a nombre escribiríamos

Alum.nombre

Alum porque así se llama la variable que es de tipo registro re_alumno .

```
/* Inicio del programa */
registro reg_alumno
      cadena[30] nombre
      entero Cuenta
      cadena[30] carrera
fin registro


reg_alumno Alum
/* declaración de la variable alum */
```

```
inicio
      Escriba "el nombre del Alumno "
      lea Alum.nombre
      Escriba "Cuenta del Alumno    "
      lea Alum.cuenta
      Escriba "carrera que estudia "
      lea alum.carrera
      Escriba " El alumno ", alum.nombre
      Escriba " Con cuenta ",alum.cuenta, " Estudia ", alum.carrera
fin
```

Ahora lo más importante es que podamos usar registros como parámetros en los procedimientos o funciones para hacer mas fácil el pasar información como parámetro .

Ejemplo registros con procedimientos

Se desea elaborar un programa donde se ingrese el nombre del alumno , la nota acumulada , examen , nota final y observación, luego que se determine la nota final y observación.

Usaremos un registro para guardar la información, un procedimiento para el ingreso de datos , otro para calcular la nota final y la observación (se usara una función para el calculo de la observación)

Siempre debemos de tomar en cuanta cuales son los parámetros de variable y de valor , en este programa usa en los procedimientos ingreso y calculo de variable y en presentar de valor porque no se modifica ninguna variable .

```
/* delcaración del regiistro*/

registro reg_alumno
      cadena[30] nombre
      real na,ne,nf
      cadena[10] obs
fin registro

/* delcaración de variables*/
reg_alumno Alum
entero I

funcion observacion (real nota): cadena[10]
      cadena[10] vobs
inicio
      si nota>= 60 entonces
            vobs<-"aprobado"
      sino
            vobs<-"reprobado"
      fin si
      retorne vobs
fin

procedimiento ingreso( var reg_alumno alum )
inicio
      escriba " Ingresar el nombre          "
      lea alum.nombre
      escriba "Ingresar la nota examen      "
      lea alum.ne
      escriba "Ingresar la nota acumulada   "
      lea alum.na
fin

procedimiento calculo(var reg_alumno alum)
inicio
      alum.nf<-alum.na + alum.ne
      alum.obs<-observacion(alum.nf)
fin
```

```
procedimiento presentar(reg_alumno alum)
inicio
      Escriba "Nota Final  ",alum.nf
      llamar nueva_linea
      escriba "Observación ",alum.obs
      llamar nueva_linea
fin

inicio
      para I<- 1 hasta 5 haga
            llamar ingreso(alum)
            llamar calculo(alum)
            llamar presentar(alum)
      fin para
fin
```

Arreglos

Es una Colección de datos del mismo tipo , que se almacenan en posiciones consecutivas de memoria y reciben un nombre común.

Y para referirse a un determinado elemento tendremos de acceder usando un índice para especificar la posición que queremos extraer o modificar su valor.

Los arreglos pueden ser :

1-Unidimensionales : solo tiene una solo dimensión una fila y una columna

2-Bidimensionales : tablas o matrices.

3-Multidimensionales : de 3 o mas dimensiones.

Arreglos de I Dimensión

Declaración :

arreglo[*<Dimensión>*] de *<Tipo de datos>* *<Nombre de la variable>*

Dimensión : es el tamaño del arreglo , es un numero entero con el cual indicamos el numero de elementos que queremos guardar con el mismo tipo.

Tipo de datos : es el tipo de datos que queremos que sea la colección , puede ser entero , real , cadena , carácter o un registro.

Nombre de la variable : es el nombre con el cual vamos a ser referencia en el programa principal

Ejemplo :

Arreglo[10] de entero numero

Con esta declaración estamos creando una colección de 10 números enteros

3	5	7	8	3	6	9	2	45	67
1	2	3	4	5	6	7	8	9	10

Siempre que nosotros queremos hacer referencia a uno de los elementos de l arreglos , tendremos que indicar la posición , con un numero entero que este dentro del rango.

Si que queremos escribir el valor de posición 7 tendremos que escribir :

```
Escriba numero[7]  /* de esta indicamos escribir la posición 7 */
```

O

```
I← 7   //asignamos un valor a una variable de tipo enteró
Escriba numero[ i ]  /* luego usamos la variable I para indicar la posición que queremos presentar.*/
```

Si deseamos asignar valores a un elemento del arreglo lo podremos hacer :

```
Lea numero[2]  /* indicamos directamente la posición que queremos leer */
```

```
I←6            /* Asignamos un valor a una variable entero  y luego la usamos  */
Lea numero[ i ]  /* para indicar la lectura de elemento que queremos leer */
```

Ejemplo

Ingresar 10 números a un arreglo de 10 elementos y luego presentar los números.

En este programa tendremos que usar un ciclo que la variable I tome un valor de 1..10 , para leer los valores o asignar valores al arreglo, luego usaremos otro ciclo para presentar los datos .

Cuando guardamos los datos en un arreglo , sus valores son almacenados en la memoria y no se borrar después al leer el siguiente numero , como en los programas anteriores , cuando usábamos una variable para ingresar 10 números , pero la variable al final del ingreso solo guardaba el ultimo numero que se introdujo, ahora con los arreglos se guardan los 10 números en la memoria .

```
/* programa de ingreso de 10 números a un arreglo */
arreglo[10] de entero numero
entero I

inicio
      para I <- 1 hasta 10 haga
          Escriba "Ingrese el numero de la pos# ", I ,"....:"
          lea numero[i]
      fin para

      para I <- 1 hasta 10 haga
            Escriba  numero[i]
            llamar nueva_linea
      fin para

fin
```

Ejemplo

Ingresar el nombre del empleado en un arreglo y el sueldo en otro arreglo , luego de ingresar los datos determinar el ihss , el total a pagar para cada uno de los empleados.

En este programa se almacena el nombre del empleado y el sueldo en dos arreglos diferentes el nombre en un arreglos de cadena y el sueldo en un arreglo de tipo real, primero se ingresa los datos en al arreglo luego se calculan los datos en otro ciclo con el fin de enfatizar que los arreglos guardan los datos en la memoria durante el programa funciona y los podemos usar después de ingresados los datos , lo que antes no podíamos hacer pues al ingresar el elemento 10 en la variable solo ese podíamos guardar , es por ello que los cálculos se hacían en el mismo ciclo.

```
Arreglo[5] de cadena[20] nombre
Arreglo[5] de real sueldo
real ihss,tp
entero I
Inicio
      para I <- 1 hasta 5 haga
                  Escriba "Nombre del empleado [",i,"]..:"
                  lea nombre[i]
                  Escriba "Sueldo del empleado ...:"
                  lea sueldo[i]
      fin para

      para I <- 1 hasta 5 haga
                  si sueldo[i]>2400 Entonces
                        ihss<-84
            sino
                      ihss<-0.035*sueldo[i]
                  fin si
                  tp<-sueldo[i]-ihss
            escriba "Nombre ...:",nombre[i]
            llamar nueva_linea
```

```
            Escriba "Sueldo ...:",sueldo[i]
            llamar nueva_linea
            Escriba "Ihss ...:",ihss
            llamar nueva_linea
            Escriba "Total pagar..:",tp
      fin para
fin
```

Uso de arreglos como parámetros en los procedimientos y funciones

En el ejemplo que, se presenta se demuestra el uso de los arreglos en las funciones y parámetros, el ejemplo muestra un procedimiento para el ingreso de datos a un arreglo de 5 números enteros, luego se usa una función de mayor que nos devuelve el numero del arreglo .

```
arreglo[5] de entero num
entero max
funcion mayor(arreglo[5] de entero num) : entero
      entero nummayor,i
inicio
      nummayor<-0
      para i <-1 hasta 5 haga
            si num[i]>nummayor entonces
                        nummayor<-num[i]
            fin si
      fin para
      retorne nummayor
fin
procedimiento ingreso(var arreglo[5] de entero num)
      entero i
inicio

      para i <-1 hasta 5 haga
             escriba "Ingresar un numero "
             lea num[i]
      fin para

fin
```

```
inicio
      llamar ingreso(num)
      max<-mayor(num)
      escriba "Mayor ", max
fin
```

Función mayor

En esta función se determina el numero mayor comparando los números que se ingresan, cuando se inicia la función nummayor vale cero pero supongamos que ingresamos en el arreglos 3-5-4-2-8

Cuando el elemento uno del arreglo se compara con 3 , hay una nueva asignación para nummayor que es 3 , cuando se compara con 5 el 3 es menor al 5 hay una nueva asignación a nummayor es 5 , cuando se compara con 4 el 5 no es menor al cuatro, asi que nummarot no se asigna ningún valor y se queda con el 5 ahora cuando se compara con 8 nummayor se le asigna el 8 porque el 5 es menor a 8 .

Num	Nummayor
cuando num[1] es 3	Entonces vale 3
cuando num[1] es 5	Entonces vale 5
cuando num[1] es 4	No hay cambio y sigue valiendo 5
cuando num[1] es 2	No hay cambio y sigue valiendo 5
cuando num[1] es 8	Entonces vale 8

Arreglos de II Dimensión (Bidimensional)

Declaración :

arreglo[*<Lineas>,<Columnas>*] de *<Tipo de datos>* *<Nombre de la variable>*

También se les denomina matrices o tablas. Un arreglo bidimensionales una tabla que ahora tiene líneas y columnas , donde las líneas indican la primera dimensión y las columnas la segunda dimensión.

	1	2	3	4
1				
2				
3				
4				
5				

La tabla que se muestra nos representa un arreglo de 2 dimensiones con 5 líneas y 4 columnas, el código para declarar este arreglo seria :

```
arreglo[5,4] de entero numero
```

La referencia a un determinado elemento de la matriz, requiere el empleo de un primero subíndice que indica la fila y el segundo que indica la columna. Ambos subíndices deberán de ser de tipo entero.

Por ejemplo si quisiéramos guardar el valor de 30 el la línea 4 columna 3 el código en LPP seria:

```
Numero[4,3]←30
```

El Siguiente ejemplo nos muestra como ingresar datos a un arreglo de 5 líneas y 4 columnas para luego presentar los datos en la pantalla:

```
arreglo[5,4] de entero numero
entero L,C

inicio
      para L <- 1 hasta 5 haga
            para c <- 1 hasta 4 haga
               Escriba "Numero[",  L , ",", C, "]"
               lea numero[L,C]
          Fin Para
      Fin Para

      llamar Limpiar_pantalla
      para L <- 1 hasta 5 haga
            para c <- 1 hasta 4 haga
                  Escriba  numero[L,C]," "
          Fin Para
          llamar nueva_linea
      Fin Para
fin
```

En este programa usamos dos variables enteras L que se usa para las líneas y C que se usa para las columnas, usamos ciclos anidados porque necesitas recorrer por cada línea, todos las columnas, esto sucede así :

Cuando la L tiene el valor de 1 la C toma el valor de 1 a 4 esto hace que se puede leer el elemento Numero [1,1], Numero [1,2], Numero [1,3], Numero [1,4] , luego cuando la L tiene el valor de 2 entonces la l vuelve a iniciar de 1 a 4 haciendo lo mimo 5 veces que es el numero de las líneas.

Suma de líneas y columnas de un arreglo Bidimensional

El programa que se presenta , ingresa los datos y los presenta usando un procedimiento

```
arreglo[5,4] de entero numero
Entero linea,col,sumaC,sumaL


funcion SumaLinea(arreglo[5,4] de entero numero, entero linea ): entero
entero sum,c
inicio
  sum<-0
   si (linea>=1) o (linea<=5) entonces
    para c<-1 hasta 4 haga
            sum<-sum + numero [linea,c]
      fin para
  fin si
  retorne sum
fin


funcion SumaColumna(arreglo[5,4] de entero numero, entero col ): entero
entero sum,L
inicio
  sum<-0
   si (col>=1) o (col<=4) entonces
    para L<-1 hasta 5 haga
            sum<-sum + numero [L,col]
      fin para
  fin si
  retorne sum
fin

procedimiento ingreso(var arreglo[5,4] de entero numero)
entero l,c
inicio
      para L <- 1 hasta 5 haga
            para c <- 1 hasta 4 haga
```

```
                Escriba "Ingresar un numero ..:"
                lea numero[L,C]
          Fin Para
          llamar nueva_linea
      Fin Para
fin

Procedimiento presentar(arreglo[5,4] de entero numero)
entero L,C
inicio
      llamar Limpiar_pantalla
      para L <- 1 hasta 5 haga
            para c <- 1 hasta 4 haga
                Escriba  numero[L,C]," "
          Fin Para
          llamar nueva_linea
      Fin Para
fin


inicio
      llamar ingreso(numero)
      llamar presentar(numero)
      Escriba "Linea a sumar"
      lea linea
      Escriba "Columna a suma"
      lea col
      sumaL<-sumaLinea(numero,linea)
      sumaC<-sumaColumna(numero,col)
      Escriba "suma de la columna ", col, " es ", sumaC
      llamar nueva_linea
      Escriba "suma de la Linea   ", Linea, " es ", sumaL
fin
```

Arreglos con registros

Hasta ahora nuestros arreglos solo han guardado un solo datos ya sea real , entero cadena o carácter , cuando se quiere guardar mas de un dato en una casilla del arreglo se declara un registro y el arreglo se declara que es del tipo registro que declaramos .

Ejemplo

```
registro Empleado
      entero codigo
      cadena[30] nombre
fin registro

arreglo[5] de empleado emple
```

Código	Código	Código	Código	Código
Nombre	Nombre	Nombre	Nombre	Nombre
1	2	3	4	5

Declaramos el registro empleado y luego declaramos el arreglo que será de tipo empleado ahora para acceder al arreglo :

Lectura de datos

```
Escriba "ingresar Nombre del Empleado "
lea emple[3].nombre
Escriba "Ingresar el codigo de registro "
lea emple[3].codigo
```

Al momento de leer , se tiene que especificar la posición del arreglo que deseo leer emple(3).nombre nos indica que se leerá de posición 3 el nombre

Escribir datos

```
Escriba "Nombre del Empleado ",emple[3].nombre
Llamar Nueva_linea
Escriba "Código de registro ", emple[3].codigo
```

Igual que al leer los datos para escribir especificamos el elemento del arreglo , del cual queremos presentar los datos del registro

Ejemplo arreglos con registro .

En este ejemplo declaramos el registro luego, se declara el arreglo de tipo registro, se elabora un procedimiento para el ingreso de los datos del arreglo y otro para presentar los registros del arreglo.

Cuando declaramos `var arreglo[5] de empleado emple` en el procedimiento de ingreso nos refeerimos a que temeos un arreglo de 5 elementos que es de tipo empleado(el registro) y que la variable se llama emple.

En ambos procedimientos se recorre el arreglo y luego por cada una de las posiciones del arreglo se ele el nombre y el código.

```
registro Empleado
      entero codigo
      cadena[30] nombre
fin registro
arreglo[5] de empleado emple

procedimiento Ingreso( var arreglo[5] de empleado emple)
entero i
inicio
        para i <- 1 hasta 5 haga
                  Escriba "ingresar Nombre del Empleado "
                  lea emple[i].nombre
                  Escriba "Ingresar el codigo de registro "
                  lea emple[i].codigo
        Fin Para
fin
```

```
Procedimiento Presentar(var arreglo[5] de empleado emple)
entero i
inicio
        llamar limpiar_Pantalla
        para i <- 1 hasta 5 haga
                  Escriba "Nombre del Empleado ",emple[i].nombre
                  Llamar Nueva_linea
                  Escriba "Código de registro ", emple[i].codigo
                  Llamar Nueva_linea
                  Llamar Nueva_linea
        Fin Para
fin

inicio
      llamar ingreso(emple)
      llamar presentar(emple)
fin
```

Ejemplo arreglos con registro.

En este ejemplo declaramos el registro luego, se declara el arreglo de tipo de tipo registro alumno , luego usamos una función para determinar la observación , no se introduce todo el registro porque solo se ocupa un dato , para determinar la observación , luego en el procedimiento de calculo al momento de enviaar la nota para usar la observación indicamos el elemento del arreglo y la parte del registro que queremos enviar que es la nota :

```
alum[i].obs<-observacion(alum[i].nf)
```

```
/* delcaración del regiistro*/
registro reg_alumno
      cadena[30] nombre
      real na,ne,nf
      cadena[10] obs
fin registro

/* delcaración del arreglo de tipo registro*/
arreglo[5] de reg_alumno Alum

funcion observacion (real nota): cadena[10]
      cadena[10] vobs
inicio
      si nota>= 60 entonces
            vobs<-"aprobado"
      sino
            vobs<-"reprobado"
      fin si
      retorne vobs
fin
```

```
procedimiento ingreso( var arreglo[5] de reg_alumno alum )
      entero I
inicio
      para i <-1 hasta 5 haga
            escriba " Ingresar el nombre          "
            lea alum[i].nombre
            escriba "Ingresar la nota examen      "
            lea alum[i].ne
            escriba "Ingresar la nota acumulada   "
            lea alum[i].na
            llamar Nueva_Linea
      fin para

fin

procedimiento calculo(var arreglo[5] de reg_alumno alum)
  entero I
inicio
      para i <- 1 hasta 5 haga
            alum[i].nf<-alum[i].na + alum[i].ne
            alum[i].obs<-observacion(alum[i].nf)
      fin para
fin

procedimiento presentar(arreglo[5] de reg_alumno alum)
entero I
inicio
      para i <- 1 hasta 5 haga
            Escriba "Nombre del alumno ",alum[i].nombre
            llamar Nueva_linea
            Escriba "Nota Final  ",alum[i].nf
            llamar nueva_linea
            escriba "Observación ",alum[i].obs
            llamar nueva_linea
            llamar Nueva_Linea
      fin para
fin
```

```
inicio

            llamar ingreso(alum)
            llamar calculo(alum)
            llamar presentar(alum)

fin
```

Ejemplo arreglos con registro.

Se declara un registro con las variables de nombre ventas, comisión ihss y total a pagar, se laboran una función para el seguro social, luego se elabora un procedimiento de ingreso de datos donde se el nombre y las ventas, después el procedimiento de calculo, donde se determina la comisión que es el 5% de las ventas, el seguro usando la función del Seguro y el total a pagar , luego se presentan los datos usando un procedimiento.

```
registro Empleado
      cadena[30] nombre
      real ventas,comis,ihss,tp
fin registro

arreglo[5] de empleado emple


funcion seguro(real sueldo) : real
inicio
      si sueldo >2400 entonces
            retorne 84
        sino
            retorne 0.035*sueldo
      fin si
fin
```

```
procedimiento Ingreso(var arreglo[5] de empleado emple)
entero i
inicio
        para i <- 1 hasta 2 haga
                  Escriba "ingresar Nombre del Empleado "
                  lea emple[i].nombre
                  Escriba "Ingresar las ventas "
                  lea emple[i].ventas
        Fin Para
fin

procedimiento Calculo(var arreglo[5] de empleado emple)
entero I
inicio
        para i <- 1 hasta 2 haga
                  emple[i].comis<-emple[i].ventas*0.05
                  emple[i].ihss<-seguro(emple[i].comis)
                  emple[i].tp<-emple[i].comis-emple[i].ihss
        Fin Para
fin


Procedimiento Presentar(arreglo[5] de empleado emple)
entero i
inicio
        para i <- 1 hasta 2 haga
                  Escriba "Empleado ",emple[i].nombre
                  Llamar Nueva_linea
                  Escriba "Comisión ..:", emple[i].comis
                  Llamar Nueva_linea
                  Escriba "Seguro Social..:", emple[i].ihss
                  Llamar Nueva_linea
                  Escriba "Total a Pagar ..:", emple[i].tp
                  Llamar Nueva_linea
                  Llamar Nueva_linea
        Fin Para
Fin
```

```
inicio
      llamar ingreso(emple)
      llamar calculo(emple)
      llamar presentar(emple)
fin
```

Manejo De Archivos De Texto

Hasta esta parte , todos los resultados de los programas se borran de la memoria al terminar el programa, en este capitulo aprenderamos como guaradar la información en un archivo de texto para su posterior utilización.

Sintaxis

Declarar un tipo archivo

Declarar un tipo archivo secuencial es necesario para , declarar variable de este tipo ejemplo :

```
tipo Arch es archivo secuencial
```

Abrir un archivo

Sintaxis

Abrir nombre_archivo como variable [para lectura, escritura]

ejemplo :

```
abrir "empleados.txt" como archemple para lectura
```

Descripción

Esta instrucción sirve para abrir el archivo. Las operaciones permitidas para el archivo son lectura, escritura o ambas. En la sintaxis variable se refiere a variable de tipo archivo que se usará para referenciar el archivo.

Cerrar un archivo

Sintaxis

Cerrar variable de tipo archivo

Ejemplo :

```
Cerrar archemple
```

Descripción

Esta instrucción sirve para cerrar un archivo. Variable

Leer de un archivo

Sintaxis

Leer variable_archivo, variable_datos

ejemplo :

```
leer archemple,emple.nombre
```

Descripción

Esta instrucción lea una variable desde un archivo. La primera variable de la instrucción debe ser de tipo archivo, la segunda puede ser de cualquier tipo, eso dependerá del tipo de archivo.

Escribir en un archivo

Sintaxis

Escribir variable_archivo, variable_datos

ejemplo :

```
escribir archemple, emple.nombre
```

Descripción

Esta instrucción escribe una variable en un archivo. La primera variable de la instrucción debe ser de tipo archivo, la segunda puede ser de cualquier tipo, eso dependerá del tipo de archivo.

Ejemplo Ingreso de datos a un archivo secuencial (texto).

Lo primero que tenemos que hacer es crear con windows un archivo de texto , con el notepad, y lo salvamos con el nombre de empleados , en el mismo directorio donde salvaremos el programa de ingreso de datos.

Declaramos el tipo de archivo secuencial

```
tipo Arch es archivo secuencial
```

luego el registro que usaremos para ingresar los datos

```
registro Empleado
     cadena[50] nombre
     real sueldo
     caracter sexo
fin registro
```

luegodeclaramos la variable para manejar el archivo de texto, que de tipo arch y la variaboe de tipo registro

```
Empleado emple
Arch ArchEmple
caracter resp
```

Luego en el programa lo primero que se hace es abrir el archivo para escritura, luego se piden los datos y se salvar en el archivo , al final se cierra el archivo de texto, ahora si nosatros queremos saber si guardo los datos , pordemos abrir empleados con el notepad y veremos los datos que se salvaron en el archivo.

```
tipo Arch es archivo secuencial

registro Empleado
      cadena[50] nombre
      real sueldo
      caracter sexo
fin registro

Empleado emple
Arch ArchEmple
caracter resp
inicio
      abrir "empleados.txt" como archemple para escritura
      repita
                  Escriba "Nombre del emnpleado..:"
                  lea emple.nombre
                  Escriba "Sueldo del empleado...:"
                  lea emple.sueldo
                  Escriba "Sexo ..:"
                  lea emple.sexo

                  escribir archemple, emple.nombre
                  escribir archemple, emple.sueldo
                  escribir archemple, emple.sexo
                  Escribir "Desea Continuar ..:"
                  lea resp
            hasta (resp="S") o (resp="N")
      hasta resp='N'
      cerrar archemple
fin
```

Ejemplo Listar el contenido de un archivo secuencial (texto).

Se declara el tipo del archivo , el registro y las variables para usar el registro y el archivo de texto , luego se abre el archivo para lectura y se hace un ciclo mientras no sea fin de archivo , esto se logra con la función FDA que nos devuelve verdatero cuando se encuentra al final del archivo y falso cuando no lo esta .

Se usa la instruccióon leer , para recuperar los valores se se guardaron el el archivo de texto, luego usando un procedimiento se escriben los valores del registro en la pantalla.

```
tipo Arch es archivo secuencial

registro Empleado
      cadena[50] nombre
      real sueldo
      caracter sexo
fin registro

Empleado emple
Arch ArchEmple
caracter detener


Procedimiento presentar(empleado emple)
inicio
      escriba "Nombre del empleado ...:",emple.nombre
      llamar nueva_linea
      escriba "Sueldo....:",emple.sueldo
      llamar nueva_linea
      escriba "Sexo......:",Emple.sexo
      llamar nueva_linea
      lea detener

fin
```

```
inicio
      abrir "empleados.txt" como archemple para lectura

      mientras no fda(archemple) haga
            leer archemple,emple.nombre
            leer archemple,emple.sueldo
            leer archemple,emple.sexo
            llamar presentar(emple)
      fin mientras
      cerrar archemple
fin
```