DATOS

Es una representación de un objeto del mundo real mediante la cual podemos modelizar cosas del problema que se quiere resolver. Puede ser constante o variable.

Imagen que contiene Texto

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente

Un tipo de dato definido por el programador/usuario es aquel que no existe en la definición del lenguaje, y el programador es el encargado de su especificación

NO puede relacionarse con ninguna otra variable que no sea de su mismo tipo , ej declaro un cadena20 = string[20] no puedo hacer operaciones con otros strings que no sean de tipo cadena20, idem si son enteros etc, resto es todo lo mismo que para cualquier stirng o entero si seria de tipo entero.

**Diagrama, Escala de tiempo

Descripción generada automáticamente**

**DATOS NUMERICOS**

**Representa un conjunto de números. Estos números pueden ser enteros o reales.**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Operaciones con datos de tipo enteros**

Importante, el div y el mod no se puede utilizar con números reales

Imagen que contiene Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Imagen que contiene Aplicación

Descripción generada automáticamente

**DATO LOGICO**

**Permite representar datos que pueden tomar dos valores verdadero o falso.**

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**DATO CARACTER**

**Representa un conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato de tipo carácter contiene solo un carácter.**

Texto

Descripción generada automáticamente

Carácter puede ser letra, carácter especial, un numero de un digito etc.

Texto

Descripción generada automáticamente con confianza baja

En representación en la tabla ascii vienen primero las mayúsculas y después las minúsculas

**DATO STRING**

Representa un conjunto finito de caracteres. Como máximo representa 255 caracteres. En general se utilizan para representar nombres.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Comparación de datos string

x:= ‘ABC’ y:=‘aBC’

si pregunto

x > y ?

lo que hace es comparar primero la primer letra de cada cadena de string, en este caso como A es menor a ‘a’ minúscula da falso, pero si fueran las dos a minúscula pasaría a comparar el siguiente carácter de cada string y asi sucesivamente hasta que uno sea distinto y de el resultado en caso de que sean distintas las cadenas, si son iguales va a dar falso ya que x seria igual a y

**VARIABLES – CONSTANTES**

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**Variables**

Es una zona de memoria cuyo contenido va a ser alguno de los tipos mencionados anteriormente. La dirección inicial de esta zona se asocia con el nombre de la variable.

Puede cambiar su valor durante el programa.

**Constantes**

Idem a variables pero NO puede cambiar su valor durante el programa.

**PRE y POST CONDICIONES**

**PRE CONDICON**

Es la información que se conoce como verdadera antes de iniciar el programa (ó módulo). Ejemplo un club pide crear un programa en los cuales los socios con números pares no se les cobrara la cuota mensual.

**POST CONDICON**

es la información que debería ser verdadera al concluir el programa (ó módulo), si se cumplen adecuadamente los pasos especificados. Por ejemplo si un cliente pide un programa que solo sume números pares, el programa debería procesar los números que se ingresan y descartar aquellos que no son pares y si sumar aquellos que son pares. Siguiendo el ejemplo de la pre condición debería efectuar una factura de prueba en la cual verique que si pongo un dni par la factura sea de cero pesos cumpliendo el objetivo del programa

**LECTURA / ESCRITURA**

**READ**

Se usa para tomar datos desde un dispositivo de entrada (por defecto desde teclado) y asignarlos a las variables correspondientes

El usuario ingresa un valor, y ese valor se guarda en la variable asociada a la operación read.

**WRITE**

Se usa para mostrar el contenido de una variable, por defecto en pantalla

El valor almacenado en la variable asociada a la operación write, se muestra en pantalla.

**ESTRUCTURAS DE CONTROL**

Conjunto de instrucciones que permiten especificar el control del algoritmo que se quiere implementar.

Diagrama

Descripción generada automáticamente

**SECUENCIA**

La estructura de control más simple, está representada por una sucesión de operaciones (por ej. asignaciones), en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones

Ej

var

num:integer;

begin

read (num);

write (num);

end.

**DECISION (if)**

En un algoritmo representativo de un problema real es necesario tomar decisiones en función de los datos del problema. La estructura básica de decisión entre dos alternativas es la que se representa simbólicamente:

Diagrama

Descripción generada automáticamente

La condición de falsa es optativo si ejecuta o no alguna acción, no necesariamente siempre lo debe hacer.

**SELECCIÓN (case)**

Permite realizar distintas acciones dependiendo del valor de una variable de tipo ordinal

Diagrama

Descripción generada automáticamente

**La variable del case debe ser de tipo ordinal**

**Las opciones deben ser disjuntas**

**ITERACION**

Si utiliza para ejecutar un bloque de instrucciones desconociendo el número exacto de veces que se ejecutaran. Se denominan estructuras de control iterativas condicionales.

Las acciones se ejecutan dependiendo de la evaluación de la condición.

Estas estructuras se clasifican en pre-condicionales y post-condicionales.

**ITERACION – PRECONDICIONAL**

Evalúan la condición y si es verdadera se ejecuta el bloque de acciones. Dicho bloque se **pueda ejecutar 0, 1 ó más veces**.

**Evalúa la condición y en caso de ser verdadera, ejecuta las acciones.**

**Se repite mientras la condición es verdadera.**

**Puede ejecutarse 0, 1 o más veces.**

Importante: el valor inicial de la condición debe ser conocido antes de la evaluación de la condición. Es decir no puedo evaluar una condición que no fue inicializada, ejemplo no puedo evaluar que num <> 0 si a num no le asigne ningún dato.

Diagrama

Descripción generada automáticamente

Al igual que en el if, puede que si la condición de falsa no ejecute ninguna acción sino que termine el programa.

**ITERACION – POSTCONDICIONAL**

Ejecutan las acciones luego evalúan la condición y ejecutan las acciones mientras la condición es falsa. **Dicho bloque se pueda ejecutar 1 ó más veces**

**Ejecuta las acciones y luego evalúa la condición.**

**Se repite mientras la condición es falsa.**

**Puede ejecutarse 1 o más veces.**

Diagrama

Descripción generada automáticamente

Cuando la condición da verdadera luego puede o no haber mas acciones

Es el repeat until

**REPETICION**

Es una extensión natural de la secuencia. Consiste en repetir N veces un bloque de acciones.

Este número de veces que se deben ejecutar las acciones es fijo y conocido de antemano

Diagrama

Descripción generada automáticamente

La variable índice debe ser de tipo ordinal

La variable índice no puede modificarse dentro del lazo

Las instrucciones dentro del for se ejecutan al menos 1 o mas veces

La variable índice se incrementa y decrementa automáticamente

Cuando el for termina la variable índice no tiene valor definido.

**TIPOS DE DATOS DEFINIDOS POR EL USUARIO**

Un aspecto muy importante en los lenguajes de programación es la capacidad de especificar y manejar datos no estándar, indicando valores permitidos, operaciones válidas y su representación interna.

Diagrama

Descripción generada automáticamente

Un tipo de dato definido por el usuario es aquel que no existe en la definición del lenguaje, y el programador es el encargado de su especificación.

Diagrama

Descripción generada automáticamente

Imagen que contiene Gráfico

Descripción generada automáticamente

En este caso edad no va a poder operar con valor

Texto

Descripción generada automáticamente

**Definidos por le usuario: SUBRANGO**

Es un tipo ordinal que consiste de una sucesión de valores de un tipo ordinal (predefinido o definido por el usuario) tomado como base.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Como es de tipo ordinal no se puede hacer con los reales por ejemplo

Imagen que contiene Gráfico

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza baja

**MODULARIZACION**

Significa dividir un problema en partes funcionalmente independientes, que encapsulen operaciones y datos.

No se trata simplemente de subdividir el código de un sistema de software en bloques con un número de instrucciones dado. Sino de separar en funciones lógicas con datos propios y datos de comunicación perfectamente especificados. Es decir, no se trata solo de dividir el código y listo, sino en realidad de dividir el problema en varios sub problemas y que cada subproblema se pueda resolver lo mas independiente posible uno de otro

Diagrama

Descripción generada automáticamente con confianza media

También cada modulo debe poder probarse de manera independiente.

**MODULO**

**Tarea específica bien definida se comunican entre sí adecuadamente y cooperan para conseguir un objetivo común.**

En ellos se pueden representar los objetivos relevantes del problema a resolver.

Existen diferentes metodologías para usarlos en los programas en particular nosotros usaremos la METODOLOGIA TOP-DOWN

La metodología top down parte de un problema grande y se va achicando el problema en problemas mas pequeños para resolver el problema original.

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Ejemplo un grupo puede trabajar en el modulo agregar, otro en la baja etc

Texto

Descripción generada automáticamente

Ejemplo el alta baja y demás es necesario no solo en no s euna concesionaria sino en un banco en una fabrica etc

Pudiendo de esta forma reutilizarse

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

**PROCEDIMIENTO**

Conjunto de instrucciones que realizan una tarea especifica y retorna 0, 1 ó más valores.

**FUNCION**

Conjunto de instrucciones que realizan una tarea especifica y retorna 1 valor **de tipo simple**.

**ALCANCE DE LAS VARIABLES**

Escala de tiempo

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza media

**PARAMETROS**

La solución a estos problemas ocasionados por el uso de variables globales es una combinación de ocultamiento de datos (Data Hiding ) y uso de parámetros.

El ocultamiento de datos significa que los datos exclusivos de un módulo NO deben ser "visibles" o utilizables por los demás módulos.

El uso de parámetros significa que los datos compartidos se deben especificar como parámetros que se trasmiten entre módulos.

Desventajas de las variables globales frente a los parámetros:

Imagen de la pantalla de un celular de un mensaje en letras blancas

Descripción generada automáticamente con confianza baja

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**PARAMETRO POR VALOR**

Un dato de entrada por valor es llamado parámetro IN y significa que el módulo recibe (sobre una variable local) un valor proveniente de otro módulo (o del programa principal).

Con él puede realizar operaciones y/o cálculos, pero no producirá ningún cambio ni tampoco tendrá incidencia fuera del módulo.

Texto

Descripción generada automáticamente con confianza media

**PARAMETRO POR REFERENCIA**

La comunicación por referencia (OUT, INOUT) significa que el módulo recibe el nombre de una variable (referencia a una dirección) conocida en otros módulos del sistema.

Puede operar con ella y su valor original dentro del módulo, y las modificaciones que se produzcan se reflejan en los demás módulos que conocen la variable.

Texto

Descripción generada automáticamente

CONDICIONES PARA LA COMUNICION POR PARAMETROS

El número y tipo de los argumentos utilizados en la invocación a un módulo deben coincidir con el número y tipo de parámetros del encabezamiento del módulo.

Un parámetro por valor debiera ser tratado como una variable de la cuál el módulo hace una copia y la utiliza localmente. Algunos lenguajes permiten la modificación local de un parámetro por valor, pero toda modificación realizada queda en el módulo en el cual el parámetro es utilizado.

El número y tipo de los argumentos utilizados en la invocación a un módulo deben coincidir con el número y tipo de parámetros del encabezamiento del módulo.

**TIPOS DE DATOS ESTRUCTURADOS**

Permite al programador definir un tipo al que se asocian diferentes datos que tienen valores lógicamente relacionados y asociados bajo un nombre único.

Tabla

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

**REGISTRO**

Es un tipo de datos estructurado, que permite agrupar diferentes clases de datos en una estructura única bajo un sólo nombre

Forma

Descripción generada automáticamente

**La característica principal es que un registro permite representar la información en una única estructura.**

Imagen que contiene Mapa

Descripción generada automáticamente

Es decir la raza de ani2 va a ser igual a la de ani1, el nombre de ani2 va a ser el mismo de ani1 etc

La lectura no se hace read(ani1) sino campo a campo ej read(ani1.raza)

Los registros se comparan campo a campo

Las operaciones son la spermitidas para el tipo según el campo

No es necesario hacer un read de todos los campos del registro ni un write. El orden en que imprimo y cargo el registro (read) puede ser en cualquier orden

**ARREGLO**

Un arreglo (ARRAY) es una estructura de datos compuesta que permite acceder a cada componente por una variable índice, que da la posición de la componente dentro de la estructura de datos.

Gráfico de cajas y bigotes

Descripción generada automáticamente con confianza media

**VECTOR (arreglo de una dimensión)**

Es una colección de elementos que se guardan consecutivamente en la memoria y se pueden referenciar a través de un índice.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**VECTOR - OPERACIONES**

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

Recordar inicializar un vector si voy a sumar o similares-

RECORRIDOS

Pueden ser parcial o total, sino tengo un arreglo completo no debo recorrerlo del todo al igual que si encuentro el elemento antes no debo recorrerlo del todo

Recorrido total con un for, recorrido parcial con un while

**DIMENSION FISICA**

Se especifica en el momento de la declaración y determina su ocupación máxima de memoria.

La cantidad de memoria total reservada no variará durante la ejecución del programa.

**DIMENSION LOGICA**

Se determina cuando se cargan contenidos a los elementos del arreglo.

Indica la cantidad de posiciones de memoria ocupadas con contenido real. Nunca puede superar la dimensión física

Es la cantidad de elementos reales que se guardan en el arreglo.

Puede modificarse durante la ejecución del programa

Nunca puede ser mayor a la dimensión física (se debe controlar)

La dimensión lógica es una variable y toma valor cuando se carga el vector.

No puede haber posiciones libres ejemplo, en un vector de 10 posiciones y 3 elementos, no pueden haber elementos en la posición 1, 5, y 8, con posiciones vacías en el medio, tienen que estar una tras otra

**VECTORES - AGREGAR**

Significa agregar en el vector un elemento detrás del último elemento cargado en el vector. Puede pasar que esta operación no se pueda realizar si el vector está lleno

**VECTORES -INSERTAR**

Significa agregar en el vector un elemento en una posición determinada. Puede pasar que esta operación no se pueda realizar si el vector está lleno o si la posición no es válida

**VECTORES -ELIMINAR**

Significa borrar (lógicamente, es decir el elemento va a seguir estando, salvo que se pise con otro dato, pero ya no voy a poder acceder a el en caso de que este en la ultima posición de la dimensión lógica ya que voy a decrementar en uno la dimensión logica) en el vector un elemento en una posición determinada, o un valor determinado. Puede pasar que esta operación no se pueda realizar si la posición no es válida, o en el caso de eliminar un elemento si el mismo no está

**VECTORES - BUSQUEDAS**

Significa recorrer el vector buscando un valor que puede o no estar en el vector. Se debe tener en cuenta que no es lo mismo buscar en un vector ordenado que en uno que no lo este.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**MEMORIA ESTATICA**

Las variables declaradas permanecen en la memoria estática durante toda la ejecución del programa, mas allá de que sigan siendo utilizadas o no.

Obviamente al permanecer en la memoria siguen ocupando memoria

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

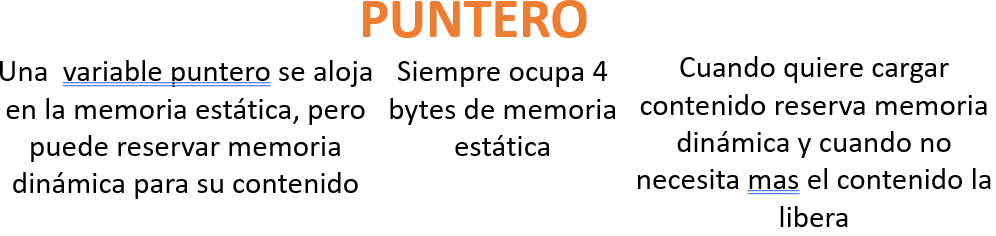
Al string su capacidad máxima es 255 caracteres, sino defino su capacidad se va a reservar 255 bytes + 1 byte

Subrango depende del tipo, si hago de enteros ocupara lo que ocupa un entero y asi

Vector ej de 10 elementos enteros ocupara 10\*6 bytes

**Memoria Dinámica**

Para solucionar los problemas mencionados anteriormente los lenguajes permiten la utilización de tipos de datos que permiten reservar y liberar memoria dinámica durante la ejecución del programa a medida que el programador lo requiera



Cuando la variable puntero reserve memoria ahí se ocupará la memoria dinámica (la cantidad de bytes de memoria dinámica dependerá del tipo de elementos que maneje el puntero)

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Una variable de tipo puntero contiene como dato una dirección de memoria dinámica. En esa dirección de memoria se encuentra el dato que realmente se quiere guardar.

**El valor puede ser de cualquiera de los tipos vistos (char, boolean, integer, real, string, registro, arreglo u otro puntero ).**

**Un puntero es un tipo de datos simple. Por lo que puede ser utilizado en una función.**

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**CREACION**

Implica reservar una dirección memoria dinámica libre para poder asignarle contenidos a la dirección que contiene la variable de tipo puntero. new(variable tipo puntero)

Begin

new (p); aca el lenguaje elije un pedazo de memoria disponible

No se puede asignar a un puntero una dirección específica adrede por nosotros

p:= ABCD

**ELIMINACION**

Implica liberar la memoria dinámica que contenía la variable de tipo puntero. dispose(variable tipo puntero)

dispose (p);

**LIBERACION**

Implica cortar el enlace que existe con la memoria dinámica. La misma queda ocupada pero ya no se puede acceder. Nil

Gráfico, Gráfico de embudo

Descripción generada automáticamente con confianza media

En el caso del dispose puede ser reutilizada la memoria liberada por otro new

En el caso del nil libera la conexión pero la memoria no puede ser utilizada por un nuevo new

**ASIGNACION entre punteros**

Implica asignar la dirección de un puntero a otra variable puntero del mismo tipo. :=

Imagen que contiene Texto

Descripción generada automáticamente

Gráfico, Gráfico de cajas y bigotes

Descripción generada automáticamente

Si hago dispose q p también queda apuntando a nada

Si hago nil p q sigue apuntando correctamente

**CONTENIDO de un puntero**

p^:=8

Captura de pantalla de un celular

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

**IMPORTANTE TODO LO QUE SE CALCULA EN MEMORIA ESTATICA ES LO DECLARADO EN VARIABLE**

Otero ejemplo

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Otro

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Otro

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Otro

Texto

Descripción generada automáticamente

Las variables declaradas para estos ejemplos son las mismas que la de la primer filmina es edecir p apunta a un real por eso vale 8 bytes

**TIPOS DE DATOS –** **LISTA**

Es una colección de nodos. Cada nodo contiene un elemento (valor que se quiere almacenar en la lista) y una dirección de memoria dinámica que indica donde se encuentra el siguiente nodo de la lista.

Toda lista tiene un nodo inicial.

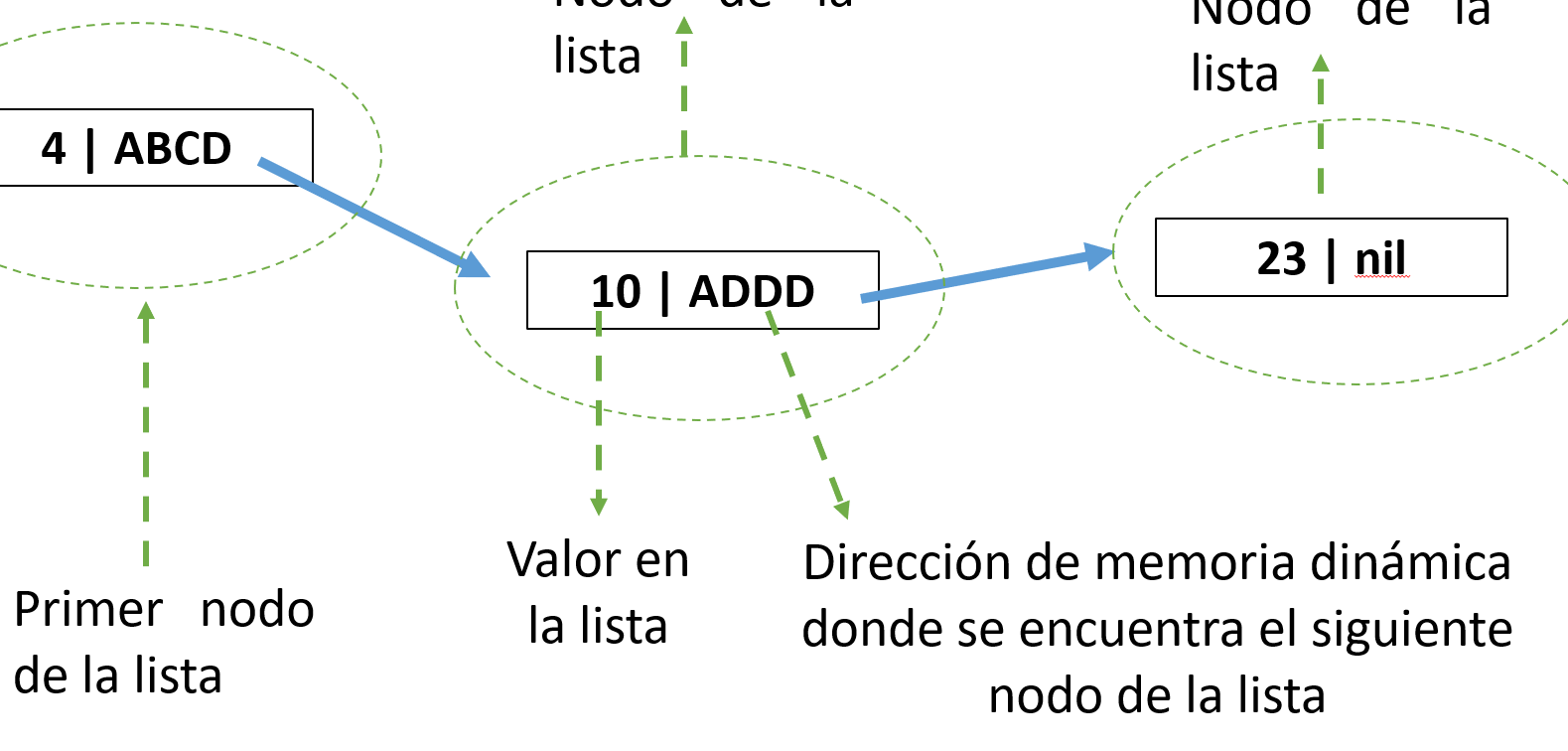
Los nodos que la componen pueden no ocupar posiciones contiguas de memoria. Es decir, pueden aparecer dispersos en la memoria, pero mantienen un orden lógico interno.

En memoria estática se declara una variable tipo PUNTERO (ya que son las única que pueden almacenar direcciones). La dirección almacenada en esa variable representa la dirección donde comienza la lista. Inicialmente ese puntero no contiene ninguna dirección.

Luego a medida que se quiere agregar elementos a la lista (nodo), se reserva una dirección de memoria dinámica y se carga el valor que se quiere guardar.

El último nodo de la lista indica que la dirección que le sigue es nil.

Un puntero ocupa 4 bytes en memoria dinámica porque aloja la dirección de memoria donde se encuentra el dato guardado, véase



En este caso es una lista de enteros, los cuales son 4,10 y 23

Texto

Descripción generada automáticamente

**DECLARACION**

Type

nombreTipo= ^nombreNodo;

nombreNodo = record

elemento: tipoElemento;

punteroSig: nombreTipo;

end;

tipoElemento es cualquiera de los tipos vistos (entero,char,boolean,registro,arreglo,real,subrangol).

Es una estructura recursiva.

Imagen que contiene Logotipo

Descripción generada automáticamente

Hacer eso no reserva espacio en memoria dinámica, solo indica que la lista esta vacia sin ninguna dirección de memoria dinámica asignada. Si haría un new ahí si reserva

**RECORRER UNA LISTA**

Implica posicionarse al comienzo de la lista y a partir de allí ir “pasando” por cada elemento de la misma hasta llegar al final.

**CORRECCION DE PROGRAMAS**

Diagrama

Descripción generada automáticamente con confianza media

El propósito del **Testing** es proveer evidencias convincentes que el programa hace el trabajo esperado.

Diagrama, Texto

Descripción generada automáticamente con confianza media

Una vez que el programa ha sido implementado y se tiene el plan de pruebas:

Texto

Descripción generada automáticamente

**DEBIGGING**

Es el proceso de descubrir y reparar la causa del error.

Para esto pueden agregarse sentencias adicionales en el programa que permiten monitorear el comportamiento más cercanamente. (como un write para ver que los resultados vayan siendo correctos durante la ejecución)

Imagen que contiene interior, pájaro, árbol, ave

Descripción generada automáticamente

**WALKTHROUGHS**

Es el proceso de recorrer un programa frente a una audiencia.

Texto

Descripción generada automáticamente

**VERIFICACION**

Es el proceso de controlar que se cumplan las pre y post condiciones del mismo.

Para determinar la corrección de un programa puedo utilizar una o varias técnicas de corrección la cantidad de veces necesarias hasta que le programa sea correcto

**EFICIENCIA DE PROGRAMAS**

Una vez que se obtiene un algoritmo y se verifica que es correcto, es importante determinar la eficiencia del mismo.

El análisis de la eficiencia de un algoritmo estudia el tiempo de ejecución de un algoritmo y la memoria que requiere para su ejecución.

Diagrama

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Imagen que contiene Diagrama

Descripción generada automáticamente

Escala de tiempo

Descripción generada automáticamente con confianza media

Imagen que contiene Diagrama

Descripción generada automáticamente

El if es aux > 45 = 1UT el and 1UT el aux < 300 1UT son 3 en total

Imagen que contiene Diagrama

Descripción generada automáticamente

Aca se calcula con el peor de los casos

Tenemos 1UT de la evaluación de la condición y después tengo que ver que tiene mas UT si lo que esta en el begin o lo que esta en el else

En el begin tengo 3UT y en el else tengo 4UT, entonces tomo el peor caso en este caso el else y el total será 5UT de la sentencia del if

Diagrama

Descripción generada automáticamente

Gráfico

Descripción generada automáticamente

En este la forma fácil de calcular N es hacer 9-4+1 =6

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza media

IMPORTANTE EL OPERADOR LOGICO SE CUENTA!!!!!