**CADP**

**Informática:** Es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.

**Memoria estática**: La memoria estática es aquella donde se almacenan las variables de las cuales su tamaño no será alterado. *Las variables solo cambian su contenido, no su tamaño.*

**Memoria dinámica:** La memoria dinámica es aquella donde las variables a almacenar no tienen un tamaño definido, y podrán aumentar o disminuir su tamaño tanto les sea permitido por la misma.

Pascal es un lenguaje de tipado estatico. Lo que significa que las variables deben declararse con un tipo especifico y no permite la conversión entre otros tipos. Esto ayuda a prevenir errores y mejora la seguridad del programa.

***String*** : Tipo de datos compuestos

***Tipos de datos*** ***Enteros:*** Es un tipo de dato simple.Son ordinales (tienen anterior y siguiente), son números enteros, por lo que no esta permitido usar números con coma, están permitidas las operaciones matematicas, lógicas y **DIV y MOD**

***Tipo de datos Reales:*** Es un tipo de dato simple, las operaciones permitidas son: Operaciones matematicas y lógicas. Con números reales no esta permitido usar **DIV** **y** **MOD**

***DIV*** : valor entero de la división , 5 div 2 = 2 (valor entero)

***MOD :*** Resto de la división , 5 div 2 = 1 (resto)

NO ES LO MISMO USAR LA OPERACIÓN **DIV** , QUE USAR **/**

Ya que el DIV se queda con el valor entero, mientras que el / se queda con el valor real de la división, es decir 5 / 2 = 2.5

**No podemos** asignarle a un **integer un numero real.**

Mientras que a **un numero real, se le puede asignar un entero**

***Variables:*** EL contenido de la variable será de algún tipo de dato posible. La dirección inicial se asocia con el nombre de la variable. Las variables pueden cambiar su valor durante el programa

***Constantes*:** El contenido de la constante será de algún tipo de dato posible. La dirección inicial se asocia con el nombre de la variable. Las constantes no pueden cambiar su valor durante el programa

**Tipo de dato:** El tipo de dato es un atributo, tanto para el ordenador, como para el programador, para saber qué valores manejará una variable declarada en memoria.

Existen varios tipos de datos, y estos se dividen en simples y compuestos.

● SIMPLES: Integer, real, boolean, char y puntero.

● COMPUESTO: String y Estructuras (registro, lista y vector).

● DEFINIDOS POR EL PROGRAMADOR: Permiten definir nuevos tipos de datos, a partir de datos simples

●DEFINIDOS POR EL LENGUAJE: Tanto las representaciones,como sus operaciones y valores son reservadas al mismo.

***El tipo de dato real no admite div y mod.***

***Estructuras pre y post condicionales***

|  |  |
| --- | --- |
| ***Pre-condicional*** | ***Post-condicional*** |
| ***EVALUA*** la condicion, y ***si es verdadera*** ***EJECUTA***. Se puede ejecutar 0,1, o mas veces. El ***while / IF*** son estructuras pre-condicionales | Primero ***EJECUTA*** y después ***EVALUA***  Este tipo de estructuras, se ejecutan AL MENOS una vez, ***repite mientras la condicion es FALSA*** , un ejemplo de esta estructura es el ***REPEAT UNTIL*** |

***Estructuras de Selección:***

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

***Cuando usar una lista y cuando usar un vector?***

Usamos un vector cuando sabemos exactamente la cantidad de datos que tenemos que guardar, es decir cuando se conoce una dimF.

Usamos una lista cuando no sabemos la cantidad de datos que vamos a guardar.

***Cuando usamos un vector contador?***

Generalmente, usamos un vector contador: Por ejemplo cuando tenemos 5 categorias de productos, y queremos saber la categoría (producto) que mas se vendió.

Siempre que usemos un vector contador debemos inicializarlo en 0. De esta manera debemos ir contando y almacenando la cantidad de productos de cada categoría. Si se quiere saber el producto con mas ventas, debemos recorrer el vector contador y sacar el máximo.

***Cuando usamos parámetros por valor y cuando por referencia?***

Por valor: cuando queremos comunicarle/ pasarle un valor al modulo, pero no queremos que ese valor de modifique.

Por referencia: Cuando queremos comunicarle / pasarle un valor al modulo pero queremos que se modifique el valor de la variable

***Cuando uso una función?***

Usamos una función cuando queremos que devuelva un tipo de dato simple, es decir: Integer,real,boolean,char,puntero,subrango. Las funciones devuelven un solo valor

***Cuando uso un corte de control?***

Generalmente, se usa corte de control cuando la información se encuentra ordenada por algún criterio.Deberíamos usar corte de control para aprovechar que la información se encuentra ordenada y poder procesar la información de manera organizada.

***Tipos de elementos en estructuras de datos***

|  |  |
| --- | --- |
| ***Homogéneas*** | ***Heterogéneas*** |
| Los elementos que la componen son del mismo tipo | Los elementos que la componen pueden ser de distinto tipo |

***Tipos de tamaño en estructura de datos***

|  |  |
| --- | --- |
| ***Estáticas*** | ***Dinámicas*** |
| El tamaño de la estructura no cambia durante la ejecución del programa | El tamaño de la estructura puede cambiar durante la ejecución del programa |

***Tipos de acceso a una estructura de datos***

|  |  |
| --- | --- |
| ***Secuencial*** | ***Directo*** |
| Para acceder a un elemento en particular, de debe pasar por todos los elementos anteriores | Se puede acceder a un elemento en particular directamente, sin tener que estar pasando por los anteriores |

***Linealidad***

|  |  |
| --- | --- |
| ***Lineal*** | ***No líneal*** |
| Esta formado por niguno, uno o varios elementos que guardan relación de adyacencia ordenada, donde a cada elemento le sigue uno y le precede uno solamente | Para un elemento pueden existir 0, 1, o varios elementos que le suceden y 0 , 1 o más elementos que le preceden |

***Estructura de datos***

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

***Vale la pena modularizar una suma?***

Sea una suma, o algunas líneas de código que se repitan varias veces a lo largo de nuestro programa, se debería modularizar.

***Que debo escribir cuando el enunciado dice “se dispone”?***

Cuando dice “Se dispone“ debemos declarar la estructura en el type, y luego cuando invoquemos a la estructura en el programa principal, deberíamos poner un comentario que diga se dispone

Ej: CargarLista(ListaNros) ;// se dispone

***Que cambia del enunciado si tenemos que eliminar un elemento de una lista pero ese elemento puede existir / no existir***

***Si el elemento seguro existe***: Debemos recorrer la lista mientras act<>nil y mientras no encontremos el elemento. Luego cuando salga del while va a ser porque encontró el elemento, entonces no hace falta preguntar si act^.elem.\_\_\_ = elemBuscado

***Si el elemento puede no existir:*** Debemos recorrer la lista mientras act <>nil y mientras no encontremos el elemento. Cuando salga del while, se debe preguntar, si act <>nil es porque se encontró el elemento, caso contrario, el elemento no se encuentra en la lista.

En resumen: El cambio que debemos hacer es al salir del while. Porque se puede encontrar el elemento, o no puede existir el elemento (Depende lo que nos diga el enunciado si seguro existe o puede no existir.)

***Puntero***

Un puntero es un tipo de dato simple, que trabaja con la memoria dinámica,

contiene la dirección de memoria, en memoria dinámica, donde se encuentra almacenado el dato real, este dato puede ser un entero, un real, un string, y los demás tipos de datos, inclusive otro puntero.

Cuando trabajamos con punteros, las únicas operaciones que modifican la memoria son new y dispose. NEW= reserva memoria, DISPOSE= libera memoria

***Tipo de dato definido por el programador***

Un tipo de dato definido por el programador es un

nuevo tipo de dato que define el programador, para su posterior uso en el programa. Este se puede obtener a partir de tipos de datos simples o compuestos.

***Estructuras de control***

***while*:** estructura de control que se ejecuta en base a una condición, puede ejecutarse 0 , 1 o mas veces. La condición deberá ser verdadera para que pueda repetir/ volver a iterar.

(no se conoce la cantidad de veces que debe iterar)

**For:** estructura de control que se ejecuta una cantidad de veces fija. Se necesita una variable índice, y para usar esta estructura se debe conocer cuantas veces debe iterar

Repeat Until: Primero ejecuta, después evalua. Se ejecuta al menos 1 vez, repite mientras la condicion sea falsa.

**Como definir un subrango?**

Para poder definir un subrango tenemos que usar tipos de datos ordinales.  
Un subrango es un tipo de dato simple y ordinal.

Es decir, puede ser un integer,boolean,char. Pero un numero real no puede ser un subrango porque no es ordinal.

La variable de tipo subrango no puede relacionarse con otra variable de otro tipo, es decir si tenemos una variable de tipo nota, solo podría relacionarse con otra variable de tipo nota.

***Ventajas de definir un subrango:***

***Flexibilidad:***

Si se modifica la forma en que se representa el dato, solo se debe modificar una declaración en vez de un conjunto de variables.

***Documentacion:***

Se pueden usar como identificación de los tipos, es decir,usando nombres autoexplicativos, facilitando el entendimiento del programa

***Seguridad:***

Se reducen los errores por uso de operaciones inadecuadas del dato, y se pueden obtener programas mas confiables

***Modularizacion***

● La modularización sirve para separar el código, de forma lógica, en diferentes

partes, encapsulando código relacionado.

● Ayuda a encontrar errores de forma más eficiente.

● Hace más legible el código.

● La reutilización es primordial cuando se habla de modularizar, evitamos programar el doble y reutilizamos el módulo en otra sección del código.

***Procedure y Function:***

En pascal existe el proceso y la función, que son dos formas de modularizar en un programa, estas están definidas por la siguiente sintaxis:

Procedure nombreProceso (parámetros de E o E/S);

**Características de procedure:**

● Podrá recibir datos, llamados parámetros por valor, y, eventualmente, podrá devolver datos, cuando el parámetro está definido por referencia.

● Puede recibir, pero no necesariamente retornar un valor.

● Puede no recibir ningún parámetro y funcionar igualmente.

Function nombreFuncion (parámetros de entrada): tipoDeFuncion;

**Características de function:**

● Podrá devolver un único valor, de tipo simple, definido en el encabezado de la

función, luego de los parámetros.

● Podrá recibir datos, pero no retornar los mismos.

● Sus parámetros no pueden ser por referencia.

● Para retornar un valor, se debe asignar un valor al nombre de la función, que es similar a una variable, guarda su valor en esa posición de memoria, de lo contrario,retornará basura.

**Caracteristicas de procedure y function:**

● Las variables definidas dentro de las mismas sólo serán exclusivas de estas, es

decir, su alcance es local.

● Se podrán llamar procesos y funciones dentro de las mismas.

● Cuando se pasa un parámetro por valor, se realiza una copia de este dato, y podrá ser utilizada dentro del módulo sin alterar su valor original.

● Los parámetros, al igual que las variables declaradas dentro del módulo, son locales al módulo.

● Ambas pueden manipular variables globales.

● Se pueden declarar types dentro de las mismas.

**Diferencias:**

● Procedure puede recibir parámetros de entrada, y también existe la opción de ser devueltos, function no tiene esta posibilidad, sólo podrá retornar un único valor.

● Los tipos de datos devueltos por estos son diferentes en cada uno, mientras el

procedure puede recibir y devolver cualquier tipo de dato, function solo puede

devolver datos de tipo simple.

● El tipo de dato que devolverá la función, al ser único, debe ser definido en el

encabezado de la misma, y solo podrá devolver ese único tipo de dato, mientras que el proceso podrá devolver todos los tipos de datos que estén por referencia en la sección de parámetros.

***Estructuras de datos:***

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

***Propiedades de las estructuras de datos:***

***Elementos: Homogenea y Heterogenea:***

● Homogénea: Los elementos que la componen son del mismo tipo.

● Heterogénea: Los elementos que la componen pueden ser de distinto tipo.

Acceso: Secuencial y Directo.

● Secuencial: Para ingresar a un valor, se debe tener un orden en particular.

● Directo: Para ingresar a un valor, solo se debe saber dónde está posicionado, o cuales su identificador.

***Linealidad: Lineal y no Lineal:***

● Lineal: Está formada por ninguno, uno o varios elementos que guardan una relación de adyacencia ordenada donde a cada elemento le sigue uno y le precede uno, solamente

● No lineal: Para un elemento dado pueden existir 0, 1 ó más elementos que le

suceden y 0, 1 ó más elementos que le preceden.

***Tamaño: Dinamico y estatico:***

● Dinámico: El tamaño puede aumentar durante la ejecución del programa.

● Estático: El tamaño no puede variar durante la ejecución del programa.

**Registros:** Un registro es una estructura de datos que contiene elementos en su interior, comúnmente llamados campos, donde cada uno tiene un identificador y un tipo de dato. Los registros son Estaticos.

Estos datos siempre están relacionados, ya que provienen de los atributos de algo en particular. El registro es directo, ya que para ingresar a cada posición se debe saber el identificador del campo, y heterogénea, ya que sus elementos pueden ser de diferentes tipos, como también, pueden ser todos de un mismo.

**Vector:** Un vector es una estructura de datos que contiene casillas, de las cuales, cada una contiene un dato, y cada casilla tiene un número que la identifica dentro del vector que va desde 1 hasta el ingresado por el programador, conocida como dimensión física.

Es directo,porque se puede ingresar a cualquier casilla sin necesidad de haber recorrido las anteriores.

Es homogénea, ya que, el vector solo admite en sus casillas elementos del mismo tipo.

Es de tamaño estático, ya que su tamaño será el declarado, y una vez compilado no podrá aumentarse.

**Listas:** Una lista, es una estructura de datos dinámica, que trabaja con punteros y nodos. Un puntero y nodo inicial comienza la lista, y contendrá, en alguno de sus campos, la iguiente posición de memoria donde se encuentre el próximo nodo para continuar la lista. Una lista es de acceso secuencial, ya que para llegar al último nodo, se deberá recorrer cada uno, hasta encontrar uno que, en el campo del siguiente nodo contenga “nil”. Es de tamaño dinámico, ya que su tamaño irá aumentando o disminuyendo a medida que se ejecute el programa, utilizando las instrucciones correspondientes (new o dispose).

***Algoritmos de búsqueda:*** Un algoritmo de búsqueda es un conjunto de instrucciones que están diseñadas para localizar un elemento con ciertas propiedades dentro de una estructura de datos.

**Búsqueda secuencial:** se basa en buscar un elemento dentro de un vector, de principio a fin, o hasta ser encontrado. En caso de que este desordenado, este será el más “eficiente”, en caso de que este ordenado, será ineficiente.

**Búsqueda binaria o dicotómica:** Este algoritmo, a diferencia de los anteriores, comienza su búsqueda por la mitad del vector, comparando el elemento buscado con esta posición, y en caso de no ser el elemento a buscar, el algoritmo decidirá ir a la primera mitad o a la segunda mitad. Este proceso se repetirá n cantidad de veces, hasta que se encuentre o no el elemento.