Informática: Es la **ciencia** que estudia el análisis y **resolución de problemas(objetivo principal de la informática, del mundo real->depende del programa (software) que carguemos en la máquina.)** utilizando **computadoras**.

PROGRAMA = ALGORITMO(instrucciones) + DATOS

Las instrucciones (acciones) representan las operaciones que ejecutará la computadora al interpretar el programa. Un conjunto de instrucciones forma un algoritmo para alcanzar un resultado en un tiempo finito.

Los datos son los valores(variables) de información necesaria para ejecutar el programa. Es una representación de un objeto del mundo real mediante la cual podemos modelizar aspectos del problema que se quiere resolver con un programa sobre una computadora. Puede ser constante o variable. Datos-> conjunto-> valores o estados posibles, operaciones permitidas, representación interna.

Un programa debe tener:

Desarrollador:

* **Operatividad**: El programa debe realizar la función para la cual fue creado.
* **Legibilidad** : El código fuente de un programa debe ser fácil de leer y entender. Esto obliga a acompañar a las instrucciones con comentarios adecuados.
* **Organización**: El código de un programa debe estar descompuesto en módulos que cumplan las subfunciones del sistema.
* **Documentados:** Todo el proceso de análisis y diseño del problema y su solución debe estar documentado mediante texto y/o gráficos para favorecer la comprensión, la modificación y la adaptación a nuevas funciones.

Computadora: máquina capaz de aceptar datos de entrada, ejecutar con ellos cálculos aritméticos y lógicos y dar información de salida (resultados), bajo control de un programa previamente almacenado en su memoria.

* Debe contener instrucciones válidas.
* Deben terminar.
* No deben utilizar recursos inexistentes.

**Tipo de Datos: es una clase de objetos de datos ligados a un conjunto de operaciones (permitidas) para crearlos y manipularlos. Un rango de valores posibles.** **Una representación interna dentro de la computadora. Se clasifican en:**

**SIMPLES: aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.**

**Definido por el lenguaje: provistos por el lenguaje y tanto la representación como sus operaciones y valores son reservadas al mismo.**

* **Numero: es el conjunto de números que se pueden necesitar. Estos números pueden ser enteros (integer: Son los más simples. Existe un valor mínimo y uno máximo. Suma, Resta, Multiplicación, División, Operaciones relacionales (< <= > => = <>), Cociente división entera (div), Resto división entera (mod)) o reales (Representan números que pueden requerir decimales. Tienen una precisión fija-> notación científica. Suma, Resta, Multiplicación, División, Operaciones relacioneales (< <= > => = <>))**
* **Lógico: permite representar datos que pueden tomar dos valores verdadero o falso(boolean). Operaciones: Disyunción (or), Conjunción (and), Negación (not).**
* **Carácter: representa un conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato de tipo caracter contiene solo un caracter. (Tabla ASCII). Valores: ‘a’, … ‘z’; ‘A’, … ‘Z’; ‘@’, … ‘!’; ‘0’, … ‘9’. Operaciones relacionales: =, <, >, <=,=>, <>.**
* **Los diferentes tipos de datos deben estar declarados y luego podemos asociar al mismo *variables*, es decir *nombres simbólicos* que pueden tomar los valores.**

**Variable: 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.**

**Constante:** **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. NO puede cambiar su valor durante el programa.**

***Pre-condición* es la información que se conoce como verdadera antes de iniciar el programa (ó módulo).**

***Post-condición*  es la información que debería ser verdadera al concluir el programa (ó módulo), si se cumplen adecuadamente los pasos especificados.**

**Definido por el programador/usuario: Permiten definir nuevos tipos de datos a partir de los tipos simples. es aquel que no existe en la definición del lenguaje, y el programador es el encargado de su especificación.**

**Cómo definimos un tipo de datos ? Type (Es el nombre con el que se conoce el nuevo tipo)-> identificador = tipo;-> (Puede ser un tipo standart o uno definido por el lenguaje) No se necesita especificar ni el conjunto de valores posibles para identificador , ni el conjunto de operaciones posibles, ya que se basan en un tipo predefinido.**

**VENTAJAS: Flexibilidad: en el caso de ser necesario modificar la forma en que se representa el dato, sólo se debe modificar una declaración en lugar de un conjunto de declaraciones de variables. Documentación: se pueden usar como identificador de los tipos, nombres autoexplicativos, facilitando de esta manera el entendimiento y lectura del programa. Seguridad: se reducen los errores por uso de operaciones inadecuadas del dato a manejar, y se pueden obtener programas más confiables.**

* **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. Es simple y ordinal. Operaciones Permitidas: Asignación,** Comparación**,** Todas las operaciones permitidas para el tipo base. No permitidas: depende del tipo base. Ventajas: Facilita el chequeo de posibles errores, pues permite que el lenguaje verifique si los valores asignados se encuentran dentro del rango establecido. Ayuda al mantenimiento del programa.

**COMPUESTOS: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.**

* **STRING: es una sucesión de caracteres de un largo determinado (max 255), que se almacenan en un área contigua de la memoria. Es compuesto y Ocupa una cantidad memoria fija. Se tienen k caracteres tratados como una sola variable (donde k es la longitud del string). Operaciones permitidas: Asignación, Comparación, Operadores relacionales. No permitidas: Copiar, concatenar, validar. Asignación: para asignar valor a una variable de tipo de dato string se hace igual que si fuera una variable de tipo carácter, :=. Si se le asigna mayor cantidad de caracteres que lo declarado como longitud máxima, los últimos a partir de esa longitud se pierden y se dice que la hilera de caracteres “se trunca”. Operadores Relacionales: los strings pueden compararse por =, <>, <=, =>.**
* **Estructuras de datos: es un conjunto de variables (que podrían ser de distinto tipo) relacionadas entre sí y que se puede operar como un todo, bajo un nombre único. Nos permite representar los elementos del mundo real. Permite al programador definir un tipo al que se asocian diferentes datos que tienen valores lógicamente relacionados y asociados bajo un nombre único. Ej: Representar los datos de un auto-> marca, modelo, año de fabricación, precio, color, etc.**

**Clasificación:**

**Tipos de datos que se almacenan (elementos): Homogenea: datos mismo tipo. Heterogenea: datos distinto tipo.**

**Ocupación de memoria (tamaño): Estática: la cantidad de memoria que ocupa no varía durante la ejecución del programa. Dinámica: la ocupación en memoria varía durante la ejecución del programa.**

**Acceso a sus elementos: Secuencial: para acceder a un elemento particular se debe respetar un orden predeterminado.** **Directo: se puede acceder a un elemento particular, directamente, sin necesidad de pasar por los anteriores a él.**

**Linealidad: 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: si para un elemento dado pueden existir 0, 1 ó mas elementos que le suceden y 0, 1 ó mas elementos que le preceden.**

**Registro: es uno de los tipos de datos estructurados, que permiten agrupar diferentes clases de datos en una estructura única. Heterogenea y estatica. Campos: son los componentes del registro. Cada uno tiene un nombre y tipo. Las operaciones a las que puede llegar un campo son todas a las que puede llegar su tipo. Se nombra y se asigna un tipo(estáticos) a cada campo.**

* Con la variable registro: Se puede asignar un registro a otro, l1:= l2 ; No se puede leer, read (l1); No se puede imprimir, write (l1). Con cada campo del registro: se puede realizar cualquier operación que permita el tipo al cual pertenece el campo; NombreVariableRegistro.nombreCampo; l1.nombre, permite todas las operaciones para strings.
* Sentencia **WITH**: permite que un registro sea nombrado una vez, y luego sea accedido directamente. Procedimiento leer utilizando WITH:

Procedure leer (var l: lugar)

Begin

WITH alu do

begin

read(nombre);

read(provincia);

read(cantV);

read(descripcion);

end;

End;

**Operaciones de entrada / salida**

**Operaciones de Entrada – 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: El valor se guarda en la variable asociada a la operación de lectura.**

**Operaciones de Salida – write: Se usa para mostrar el contenido de una variable, por defecto en pantalla. Informa un texto, el contenido de una variable, la combinación de ambos, o el resultado de alguna cuenta.**

**Concepto de Estructura de Control: permiten modificar el flujo de ejecución de las instrucciones de un programa. Todos los lenguajes de programación tienen un conjunto mínimo de instrucciones que permiten especificar el control del algoritmo que se quiere implementar. Como mínimo deben contener: secuencia, decisión e iteración.**

**Secuencia: La más simple, s*ucesión de operaciones* (por ej. asignaciones), en la que el orden de ejecución coincide con el orden físico de aparición de las instrucciones.**

**Decisión: Es necesario tomar *decisiones* en función de los datos del problema. La estructura básica de decisión entre *dos* alternativas acerca de una condición. If/else. La evaluación de la condición debe retornar verdadero o falso. Si no existe el else se continúa con el programa. Cuando se debe realizar mas de una acción se debe agregar el begin end.**

**Selección: Más de 2 alternativas. Case… Of: La variable de decisión debe ser de tipo ordinal. Puede haber más de un valor en cada una de las entradas. Un valor debe aparecer en una sola de las entradas. Deben incluirse todas las posibilidades.**

**Como leo un carácter? Pongo una variable de tipo carácter. Leo esa variable. Pregunto que carácter es y si no es ese es otro y uso el else. Como determino que posibilidad es? Case variable of-> posibilidad 1: accion1 posibilidad2: begin-> acción2 accion3-> end;-> else acción-> end;. Como informo un carácter? Begin-> read (car); case car of-> ‘a’ .. ‘ z’: write (“El carácter es una minúscula”);-> else write (“El carácter es una mayuscula”)-> end .**

**Repetición: *repite N veces un bloque de acciones. For (índice):= A to B do .* Este número de veces que se deben ejecutar las acciones es fijo y conocido de antemano. Consideraciones acerca de la variable índice: La variable de control debe ser de tipo ordinal (entero, boolean, char). NO debe modificarse dentro del lazo. Los incrementos ó decrementos y testeos son implícitos. Al terminar el ciclo, la variable índice no tiene un valor definido (su uso se limita a la repetición).**

**Iteración: para ejecutar acciones sin saber la cantidad de veces.**

**Pre condicionales: Evalúan la condición y si es verdadera ( se conoce antes de iniciar el programa) se ejecuta el bloque de acciones. Dicho bloque se pueda ejecutar 0, 1 ó más veces. Se repite mientras la condición es verdadera<- While (condición) do .**

**Post condicionales: Ejecutan las acciones (bloque) luego evalúan la condición y ejecutan las acciones mientras la condición es falsa. Dicho bloque se pueda ejecutar 1 ó más veces. Se Repeat ( bloque acciones) until (condición). Debería ser verdadera la información al concluir el programa.**

**Modularizar: A partir del modelo es necesario encontrar la forma de descomponer en partes (módulos) para obtener una solución. La descomposición funcional de todas las acciones que propone el modelo nos ayudará a reducir la complejidad, a distribuir el trabajo y en el futuro a reutilizar los módulos. Modularizar significa dividir un problema en partes funcionalmente independientes, que encapsulen operaciones y datos. Separar en funciones lógicas con datos propios y datos de comunicación. Objetivo: cada modulo (Tarea específica bien definida comunicada por parámetros. encapsula, acciones tareas o funciones ) puede resolverse independientemente y la solución de los mismos combinados resuelven el problema original.**

**Cuando se descompone un problema en subproblemas, deben ser de forma tal que: Cada subproblema está en un mismo nivel de detalle; puede resolverse independientemente. Las soluciones de estos puede combinarse para resolver el problema original.**

Genera: Mayor Productividad: al dividir un sistema de software en módulos independientes un equipo puede trabajar simultáneamente en varios modulos, aumeta la producción.

Reusabilidad: posibilidad de utilizar repetidamente el producto de software desarrollado.

Facilidad de Mantenimiento: al dividir un sistema en módulos permite aislar los errores que se producen con mayor facilidad (Corregir los errores en menor tiempo y disminuir los costos de mantenimiento de los sistemas).

Facilidad de Crecimiento: con el tiempo aparecen nuevos requerimientos del usuario(software). Esto permite disminuir los riesgos y costos de incorporar nuevas prestaciones a un sistema en funcionamiento.

Legibilidad: mayor claridad para leer y comprender el código fuente.

COMUNICACIÓN: La comunicación externa de un módulo con el resto del sistema (puede no existir) se produce a través de datos de entrada y datos de salida.

Variables globales: Problemas: Demasiados identificadores. No se especifica la comunicación deseada entre los módulos. Conflictos de nombres de identificadores utilizados por diferentes programadores. Posibilidad de perder integridad de los datos, al modificar involuntariamente en un módulo datos de alguna variable que luego deberá utilizar otro módulo.

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

Mecanismos:

Parámetros 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). Los cambios que realice, no se verán reflejados fuera del módulo. Ej: **procedure uno (nombre1: tipo; nombre2: tipo);. Debería ser tratado como una variable de la cuál el Procedimiento hace una copia y la utiliza localmente.**

**Parámetro por referencia: La comunicación por referencia (OUT, INOUT) significa que el módulo recibe el nombre de una variable (referencia a una dirección). 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. Ej: procedure uno (var nombre1: tipo; var nombre2: tipo);. Operan directamente sobre la dirección de la variable original, en el contexto del módulo que llama. (No requiere memoria local).**

Modulos:

Procedure: Conjunto de instrucciones que realizan una tarea especifica y retorna 0, 1 ó más valores. Se invocan por su nombre. Devuelven cualquier tipo de valor.

Función: Conjunto de instrucciones que realizan una tarea especifica y retorna un único valor de tipo simple. Se invocan: por su nombre: asignando a una variable el valor; dentro de un if/ while (en los procedimientos no existe); dentro de un write (en los procedimientos no existe). Devuelven valores de tipos simples. Solo parámetros por valor.

Alcance de las variables

Program alcance;

Const

…

Type

…

**Var Variables globales: usadas en todo el programa (incluyendo módulos)**

a,b: integer;

**procedure prueba;**

|  |
| --- |
| Var **Variable local del proceso: usadas sólo en el proceso que están declaradas**  c: integer; Se busca si es variable local. Se busca si es un parámetro Se busca si es variable global al programa. |

Begin

End.

|  |
| --- |
| **Var variable local del programa: usadas sólo en el cuerpo del programa**  d:integer; Se busca si es variable local al programa. Se busca si es variable global al prog. |

Begin

End.