En informática el término **programación** se refiere a la acción de crear programas y programar es la serie de instrucciones, que le vamos a dar a nuestra máquina, para lograr lo que nuestro programa necesite para funcionar.

Las **partes** que componen a nuestro programa son el lenguaje de programación y los algoritmos.

Lenguaje de programación: es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y algoritmos para, de esa forma, resolver problemas.   
Las instrucciones que sigue la computadora para la creación de programas están escritas en un lenguaje de programación y luego son traducidas a un lenguaje de máquina que puede ser interpretado y ejecutado por el hardware del equipo.  
Hay distintos tipos de lenguajes de programación: • Lenguaje máquina: dígitos binarios o bits (0 y 1) • Lenguajes de alto nivel: Tienen como objetivo facilitar el trabajo del programador, ya que utilizan unas instrucciones más fáciles de entender.

Algoritmo: estas instrucciones que le vamos a dar a nuestro programa, se conocen como algoritmos. Un algoritmo es un método para darle instrucciones a nuestro programa y resolver un problema.  
Características: • En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el **algoritmo será siempre el mismo**.  
 • Un algoritmo debe ser **preciso** e indicar el **orden** de realización de cada paso.   
 • Un algoritmo debe estar específicamente **definido**. Es decir, si se ejecuta un mismo algoritmo dos veces, con los mismos datos de entrada, se debe obtener el mismo resultado cada vez.   
 • Un algoritmo debe ser **finito**. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos. Debe tener un inicio y un final.   
 • Un algoritmo debe ser **correcto**: el resultado del algoritmo debe ser el resultado esperado.   
 • Un algoritmo es **independiente** tanto del lenguaje de programación en el que se expresa como de la computadora que lo ejecuta.  
Tanto el lenguaje de programación como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

Un algoritmo básicamente consta de tres elementos: Datos de Entrada o Información de entrada, Procesos y la Información de Salida.

La mejor manera de representar un algoritmo es a través de un diagrama de flujo.

¿Dónde se van a ver reflejados estos dos conceptos? Estos se van a ver reflejados en nuestro programa.  
Un programa no es más que una serie de algoritmos escritos en algún lenguaje de programación de computadoras. Un programa es, por lo tanto, un conjunto de instrucciones — órdenes dadas a la computadora— que producirán la ejecución de una determinada tarea. En esencia, un programa es un medio para conseguir un fin. El fin será probablemente definido como la información necesaria para solucionar un problema.  
Un programa puede ser lineal (secuencial) o no lineal. Un programa es lineal si las instrucciones (acciones) se ejecutan secuencialmente como los ejercicios propuestos en esta guía, es decir, sin bifurcaciones, decisión ni comparaciones.

Estructura: Un algoritmo consta de dos componentes: una cabecera de programa y un bloque algoritmo. El bloque algoritmo consta de dos componentes o secciones: las acciones de declaración y las acciones ejecutables.

Elementos: Los elementos de un programa, son básicamente, los componentes que conforman las instrucciones previamente mencionadas, para crear nuestro programa y resolver sus problemas. Estos elementos siempre estarán dentro de un algoritmo. Los elementos de un programa son: **identificadores, variables, constantes, operadores, palabras reservadas.**

**Identificadores:** sirve para identificar las entidades del programa (nombre del programa, nombres de variables, constantes, subprogramas, etc.). En PseInt los identificadores deben constar sólo de letras, números y/o guión bajo \_, comenzando siempre con una letra y se suelen escribir siempre en minúsculas. Estos tampoco pueden contar de tildes, ni de la letra Ñ, ya que generaría errores. Si queremos poner un identificador que sea de más de una palabra, usamos algo llamado camelCase.

**Variables y constantes:** vendrían a ser como pequeñas cajas, que guardan algo en su interior, en este caso información.

Tipos de datos:

✓ Entero.  
✓ Real: para separar decimales se utiliza el punto.   
✓ Carácter: comillas simples, un carácter (unidimensional): ‘a’, 'A'.   
✓ Lógico: cuando necesitamos guardar una expresión lógica (verdadero o falso. ✓ Cadena: cadenas se encierran entre comillas dobles. una cadena (multidimensional): “esto es una cadena”, "hola mundo"

Notas: Cadena y Carácter son términos equivalentes, no genera error que las escribamos indistintamente.

Estructura de Control: determinan el orden en que deben ejecutarse las instrucciones de un algoritmo, es decir, si serán recorridas una después de la otra (estructuras secuenciales), si habrá que tomar decisiones sobre si ejecutar o no alguna acción (estructuras selectivas o de decisión) o si habrá que realizar repeticiones (estructuras repetitivas). Esto significa que una estructura de control permite que se realicen unas instrucciones y omitir otras, de acuerdo a la evaluación de una condición.

✓ Secuencial: Es la estructura en donde una acción (instrucción) sigue a otra de manera secuencial. Las tareas se dan de tal forma que la salida de una es la entrada de la que sigue y así en lo sucesivo hasta cumplir con todo el proceso.

✓ Selectiva: en estas estructuras, se realiza una evaluación de una condición y de acuerdo al resultado, el algoritmo realiza una determinada acción. Las condiciones son especificadas utilizando expresiones lógicas. En programación, una condición es toda sentencia de la cual se puede determinar su verdad (true) o falsedad (false). En su gran mayoría, son comparaciones. Las estructuras selectivas/alternativas pueden ser:   
• Simples: Si  
• Doble: Si - SiNo  
• Múltiples: Según – Si anidado

#### Funciones PseInt:

|  |  |
| --- | --- |
| Funciones | Significado |
| RC(número) | Devuelve la raíz cuadrada del número. |
| ABS(número) | Devuelve el valor absoluto del número |
| LN(número) | Devuelve el logaritmo natural del número |
| EXP(número) | Devuelve la función exponencial del número. |
| SEN(número) | Devuelve el seno de número. |
| COS(número) | Devuelve el coseno de número. |
| TAN(número) | Devuelve la tangente de número. |
| ASEN(número) | Devuelve el arcoseno de número. |
| ACOS(número) | Arcocoseno de x |
| ATAN(número) | Arcotangante de x |
| MOD | Devuelve el módulo (resto de la división entera). |
| TRUNC(número) | Trunca el valor x (parte entera de x) |
| REDOND(número) | Redondea al valor más cercano a x |
| AZAR(número) | Entero aleatorio entre 0 y x -1 |
| ALEATORIO(min,max) | Entero aleatorio entre valor mínimo y máximo |

Funciones cadena de texto:

|  |  |
| --- | --- |
| Funciones | Significado |
| Longitud(cadena) | Devuelve la cantidad de letras que compone la cadena. |
| Mayusculas(cadena) | Devuelve una copia de la cadena con todas sus letras en mayúsculas. |
| Minusculas(cadena) | Devuelve una copia de la cadena con todas sus letras en minúsculas. |
| Subcadena(cadena, posición\_inicial, posición\_final) | Devuelve una nueva cadena que consiste en la parte de la cadena que va desde la posición pos\_inicial hasta la posición pos\_final. |
| Concatenar(cadena, cadena2) | Devuelve una nueva cadena que resulta de unir las cadenas cadena1 y cadena2. |
| ConvertirANumero(cadena) | Recibe una cadena compuesta de números y devuelve la cadena como una variable numérica. |
| ConvertirACadena(cadena) | Recibe un número y devuelve una variable cadena de caracteres de dicho número. |

✓ Repetitivas: las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles, y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones.  
Hay distintos tipos de bucles:  
• Mientras  
• Hacer Mientras   
• Para

Subprogramas:

Un método muy útil para solucionar un problema complejo es dividirlo en subproblemas —problemas más sencillos— y a continuación dividir estos subproblemas en otros más simples, hasta que los problemas más pequeños sean fáciles de resolver.  
Este método de diseñar la solución de un problema principal obteniendo las soluciones de sus subproblemas se conoce como diseño descendente (top-down). Se denomina descendente, ya que se inicia en la parte superior con un problema general y se termina con varios subproblemas de ese problema general y los soluciones a esos subproblemas. Luego, las partes en que se divide un programa deben poder desarrollarse independientemente entre sí.  
Este método de diseñar la solución de un problema principal obteniendo las soluciones de sus subproblemas se conoce como diseño descendente (top-down). Se denomina descendente, ya que se inicia en la parte superior con un problema general y se termina con varios subproblemas de ese problema general y los soluciones a esos subproblemas. Luego, las partes en que se divide un programa deben poder desarrollarse independientemente entre sí.

Tipos de variables:

Una **variable local** es aquella que está declarada y definida dentro de un subprograma, en el sentido de que está dentro de ese subprograma, es distinta de las variables con el mismo nombre declaradas en cualquier parte del programa principal y son variables a las que el algoritmo principal no puede acceder de manera directa.   
El significado de una variable se confina al procedimiento en el que está declarada. Cuando otro subprograma utiliza el mismo nombre se refiere a una posición diferente en memoria. Se dice que tales variables son locales al subprograma en el que están declaradas.   
Una **variable global** es aquella que está declarada en el programa o algoritmo principal, del que dependen todos los subprogramas y a las que pueden acceder los subprogramas, a través del paso de argumento. La parte del programa/algoritmo en que una variable se define se conoce como ámbito o alcance (scope, en inglés).

Cuando un programa **llama** a un subprograma, la información se comunica a través de la lista de parámetros y se establece una correspondencia automática entre los parámetros y los argumentos. Los parámetros son “sustituidos” o “utilizados” en lugar de los argumentos. La declaración del subprograma se hace con:   
Subproceso nombre (PA1, PA2, ...,PAn)   
FinSubproceso   
y la llamada al subprograma con:   
nombre (AR1, ARG2,…, ARGn)   
donde PA1, PA2, ..., PAn son los parámetros y ARG1, ARG2, ..., ARGn son los argumentos.

Para esto existen **dos tipos** más empleados para realizar el paso de argumentos, el paso por valor y el paso por referencia:   
PASO POR VALOR: Los argumentos se tratan como variables locales y los valores de dichos argumentos se proporcionan copiando los valores de los argumentos originales. Los parámetros (locales a la función o procedimiento) reciben como valores iniciales una copia de los valores de los argumentos y con ello se ejecutan las acciones descritas en el subprograma. Los cambios que se produzcan dentro del subprograma no producen cambios en los argumentos originales y, por consiguiente, no se pueden poner argumentos como valores de retorno.

En PSeInt todas las **variables** que pasemos como argumentos pasan por defecto “Por Valor” sino se especifica lo contrario explícitamente.

PASO POR REFERENCIA: En numerosas ocasiones se requiere que ciertos argumentos sirvan como argumentos de salida, es decir, se devuelvan los resultados al programa que llama. Este método se denomina paso por referencia o también de llamada por dirección o variable. El programa que llama pasa al subprograma la dirección del argumento actual (que está en el programa que llama). Una referencia al correspondiente argumento se trata como una referencia a la posición de memoria, cuya dirección se ha pasado. Entonces una variable pasada como argumento real es compartida, es decir, se puede modificar directamente por el subprograma.  
En este método los argumentos son de entrada/salida y los argumentos se denominan **argumentos variables**.

Funciones:

Matemáticamente una función es una operación que toma uno o más valores llamados argumentos y produce un resultado.  
Cada función se evoca utilizando su nombre en una expresión con los argumentos encerrados entre paréntesis. A una función no se le llama explícitamente, sino que se le invoca o referencia mediante un nombre y una lista de parámetros.  
Los parámetros no son obligatorios a la hora de usar un subprograma, podemos tener una función sin parámetros, aunque es poco común.  
Por último, irá el cuerpo de la función, que será una serie de acciones o instrucciones cuya ejecución hará que se asigne un valor al nombre de la función. Esto determina el valor particular del resultado que ha de devolverse al programa llamador.

Funcion variable\_de\_retorno <- Nombre ( Argumentos )  
 Definir variable\_de\_retorno como Tipo de Dato  
 acciones // cuerpo de la función  
Fin Funcion

Se denominan argumentos a las variables o valores declarados en el algoritmo. Cuando se realiza una llamada a la función, los "valores" pasados o enviados a la función se denominan argumentos.  
Debe haber exactamente el mismo número de parámetros que de argumentos en la declaración de la función y se presupone una correspondencia uno a uno de izquierda a derecha entre los argumentos y los parámetros.

La función tiene cinco componentes importantes:   
• Identificador  
• Parámetros son los valores que recibe la función para realizar una tarea.   
• Argumentos son los valores que envía el algoritmo a la función.   
• Acciones de la función, son las operaciones que hace la función.   
• valor de retorno (o el resultado), es el valor final que entrega la función. La función devuelve un único valor.

Procedimientos

Los procedimientos son similares, pero no devuelven ningún valor, sólo realizan una tarea.  
Un procedimiento es un subprograma que ejecuta un proceso específico. En PseInt lo llamaremos SubProceso. Ningún valor está asociado con el nombre del procedimiento; por consiguiente, no puede ocurrir en una expresión. Un procedimiento se llama escribiendo su nombre. Cuando se invoca el procedimiento, los pasos que lo definen se ejecutan y a continuación se devuelve el control al programa que le llamó.   
SubProceso Nombre (parámetros)   
 Acciones  
FinSubProceso

Arreglos

Si viéramos las variables como cajas, estas nos permitían guardar un solo dato. Los arreglos son como cajas, pero dentro hay compartimentos, que permiten guardar varios datos, siempre y cuando sean del mismo tipo. Un array o arreglo (matriz o vector) es un conjunto finito y ordenado de elementos homogéneos.  
Hay que tener en cuenta que los arreglos, a diferencia de los tipos de datos simples, pasan siempre como parámetro “**Por Referencia**”, ya que usualmente en nuestros subprogramas usamos los arreglos para rellenar, mostrar nuestros arreglos, etc.

Arreglo unidimensional: Vector

Un vector es un arreglo de n elementos, uno detrás de otro, que posee las siguientes características:   
• Se identifica por un único nombre de variable.   
• Sus elementos se almacenan en posiciones del vector y cada a posición le corresponde un subíndice.   
• Se puede acceder a cada uno de sus elementos a través del subíndice de forma ordenada o en forma aleatoria.   
• Su tamaño es finito, esto significa que una vez definido su tamaño, este no puede cambiar. El tamaño es la cantidad de elementos que puede guardar nuestro vector.

v = [1, 2, 3 ,4 …n]

El valor inicial del primer subíndice depende del lenguaje; la mayoría de los modernos inician con el cero, por lo tanto, en PSeInt comenzarán en cero y los posibles valores de los subíndices irán desde 0 hasta N-1  
Declaración:   
Definir nombre\_vector como Tipo\_de\_Dato  
Dimension nombre\_vector(tamaño)

Arreglo multidimensional: matrices

En una matriz un subíndice no es suficiente para especificar un elemento. En general, un arreglo de n-dimensiones requiere que los valores de los n subíndices puedan ser especificados a fin de identificar un elemento individual del arreglo. Si cada componente de un arreglo tiene n subíndices, el arreglo se dice que es sólo de n-dimensiones.  
Declaración:   
Definir nombre\_matriz como Tipo\_de\_Dato   
Dimension nombre\_matriz(tamañoDim1, tamañoDim2, …, tamañoDimN)