PROGRAMACIÓN UD-1

ALGORITMOS

ÍNDICE

[DEFINICIÓN DE ALGORITMO 3](#_Toc115277134)

[ELEMENTOS PARTICIPANTES EN LA REALIZACIÓN DE UN ALGORITMO 3](#_Toc115277135)

[FORMAS DE DESCRIBIR UN ALGORITMO 4](#_Toc115277136)

[A. LENGUAJE NATURAL 4](#_Toc115277137)

[B. ORGANIGRAMAS 5](#_Toc115277138)

[C. PSEUDOCÓDIGO 6](#_Toc115277139)

[D. DIAGRAMAS DE CAJAS O DE CHAPIN O NASSI-SHNEIDERMAN 7](#_Toc115277140)

[E. DIAGRAMAS WARNIER 8](#_Toc115277141)

[F. MÉTODO JACKSON 9](#_Toc115277142)

[G. MÉTODO BERTINI 9](#_Toc115277143)

[H. MÉTODO TABOURIER 10](#_Toc115277144)

[I. LENGUAJES DE PROGRAMACIÓN 10](#_Toc115277145)

[DESARROLLO E IMPLEMENTACIÓN DE UN ALGORITMO 11](#_Toc115277146)

[CONCEPTOS FUNDAMENTALES 12](#_Toc115277147)

[INTRODUCCIÓN DE LA NOTACIÓN ALGORÍTMICA A EMPLEAR 12](#_Toc115277148)

# DEFINICIÓN DE ALGORITMO

Los algoritmos se pueden definir como:

*“Una secuencia de pasos que son llevados a cabo de forma mecánica y sistemática por un actor que se desenvuelve en un entorno dado para resolver un problema determinado en un tiempo infinito.”*

O también como:

*“Una combinación de instrucciones combinadas de forma adecuada para resolver un determinado problema en una cantidad finita de tiempo. Cada instrucción es una indicación y no ambigua.”*

Por tanto, como propiedades generales los algoritmos:

* Resuelven un **problema específico**.
* Son llevados a cabo por una **entidad** que trabaja en un **entorno adecuado**.
* Constan de una **serie de pasos** que se llevan a cabo siguiendo una **secuencia marcada**.
* Se aplican de **forma mecánica**.
* Terminan en un **tiempo finito**.

# ELEMENTOS PARTICIPANTES EN LA REALIZACIÓN DE UN ALGORITMO

Los elementos que participan en un algoritmo son:

* **Procesador:** Es un ente que es capaz de entender los pasos que componen el algoritmo y llevarlos a cabo.
* **Entorno:** Es el conjunto de materiales necesarios para la ejecución del algoritmo.
* **Acción:** Es un suceso llevado a cabo por el procesador y que modifica el entorno.

Usando el ejemplo de un ordenador tendríamos. Un procesador que sería el propio ordenador, un entorno que serían los datos con los que se va a trabajar y unas acciones que serían las operaciones que el ordenador puede realizar con esos datos.

# FORMAS DE DESCRIBIR UN ALGORITMO

## LENGUAJE NATURAL

Es la forma más sencilla. Por ejemplo, el algoritmo para encontrar las raíces de una ecuación de segundo de segundo grado sería:

1. Definir los coeficientes de la ecuación de segundo grado:
2. Determinar el valor del discriminante:
3. Si el discriminante es cero sólo hay una solución posible:
4. Si el discriminante es positivo hay dos soluciones:
5. Si el discriminante es negativo no hay soluciones reales.

|  |  |
| --- | --- |
| VENTAJAS | INCONVENIENTES |
| Facilidad de comprensión  Cualquier hispanohablante podría entenderlo sin problemas | **No es universal**  Este ejemplo sólo valdría para hispanohablantes |
|  | **Es ambiguo**  Por tanto, susceptible a errores |
|  | **Es demasiado amplio**  Una misma instrucción puede ser sencilla para una persona pero no para otra. |

## ORGANIGRAMAS

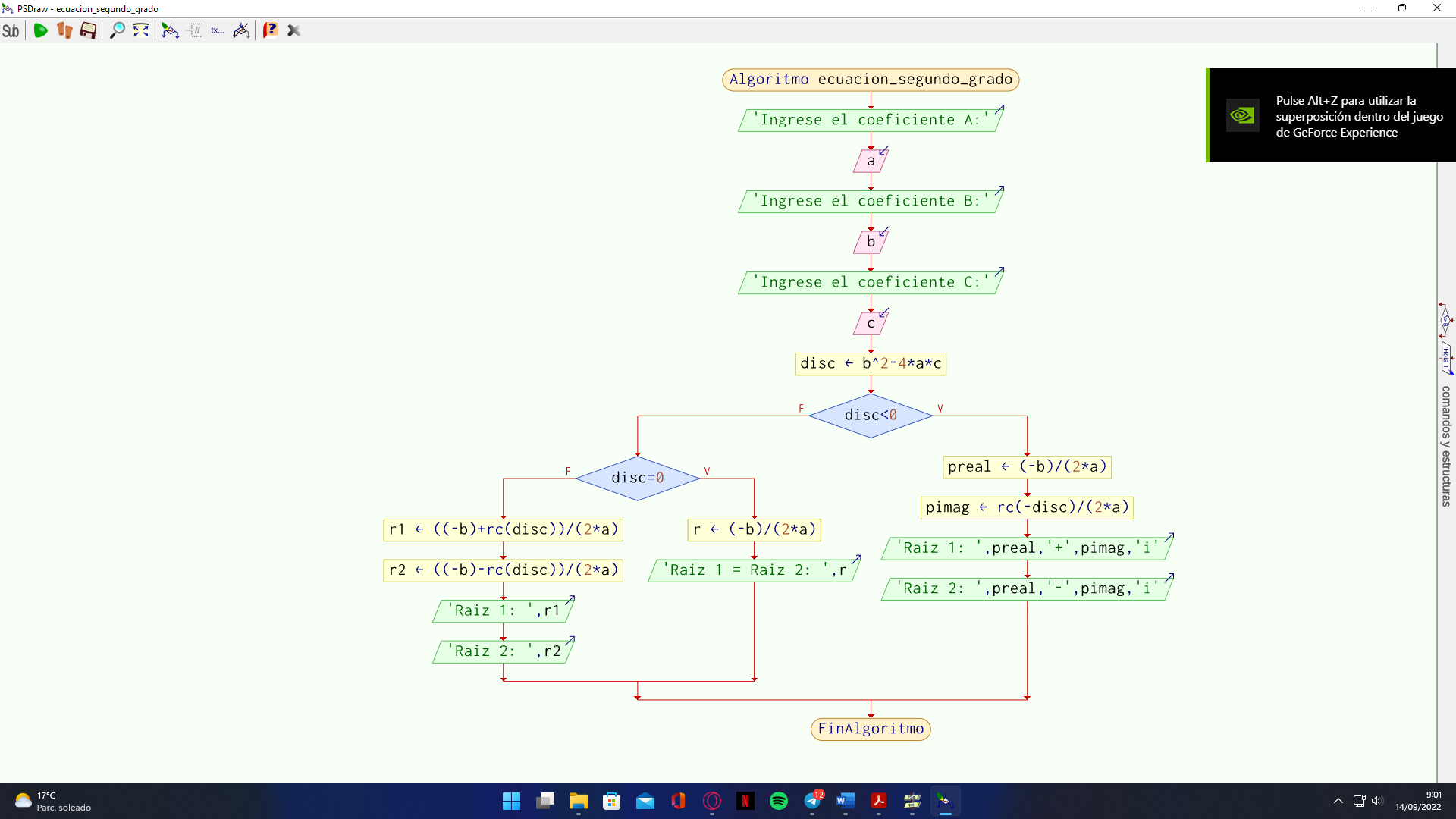
También llamados diagramas de flujo permiten describir los algoritmos de forma gráfica.

Algunos de los símbolos que se emplean son los siguientes:

|  |  |
| --- | --- |
| **TERMINAL** | Punto de comienzo o final de un programa. |
| **ENTRADA** | Información introducida para su proceso. |
| **SALIDA** | Información generada como resultado. |
| **DECISIÓN** | Operación que determina varios caminos alternativos a seguir. |
| **PROCESO** | Cualquier proceso distinto de la entrada/salida o de las decisiones. |

|  |  |
| --- | --- |
| VENTAJAS | INCONVENIENTES |
| Son universales | **Dificultad de comprensión** |
| Son menos propensos a la ambigüedad | **Texto en lenguaje natural** |
| Delimitan mejor los algoritmos | **Representación tediosa y a veces confusa** |
| Se aproximan a la forma en que trabaja el ordenador | **Un ordenador no es capaz de utilizar una representación como descripción algorítmica.** |

En el ejemplo de las ecuaciones de las raíces cuadradas el diagrama de flujo sería el siguiente:



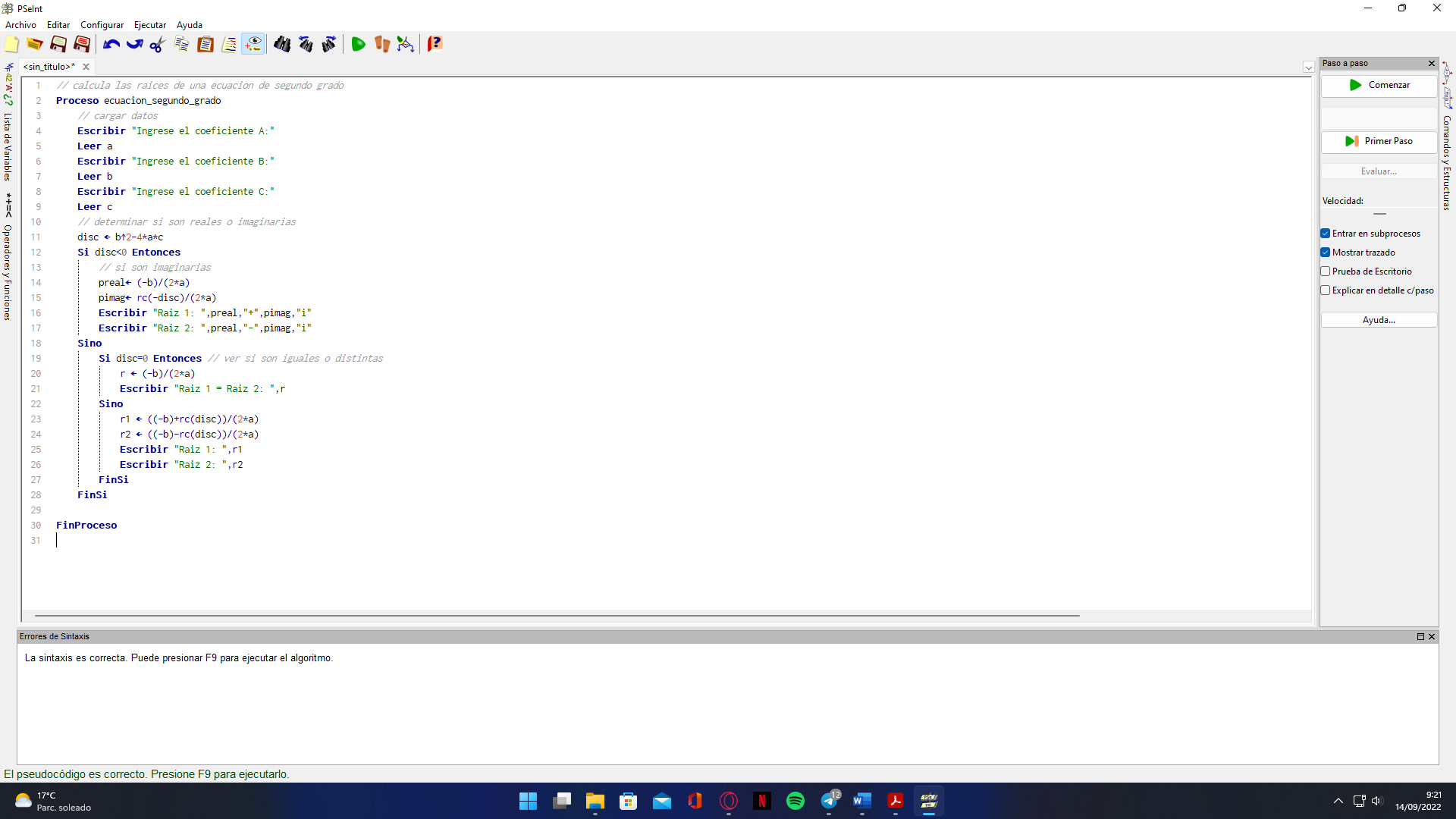
## PSEUDOCÓDIGO

Combina las ventajas del lenguaje natural y de los organigramas sin ninguno de sus problemas. Por tanto:

* + Es **fácilmente comprensible** para una persona que lo vea por primera vez.
  + Está **bien delimitado**.
  + **Elimina las ambigüedades** del lenguaje natural.
  + Se representa de **forma compacta**.

El pseudocódigo se suele ver como un subconjunto del lenguaje natural que proporciona un número limitado de operaciones para la construcción de algoritmos. Su finalidad es la comunicación entre seres humanos.

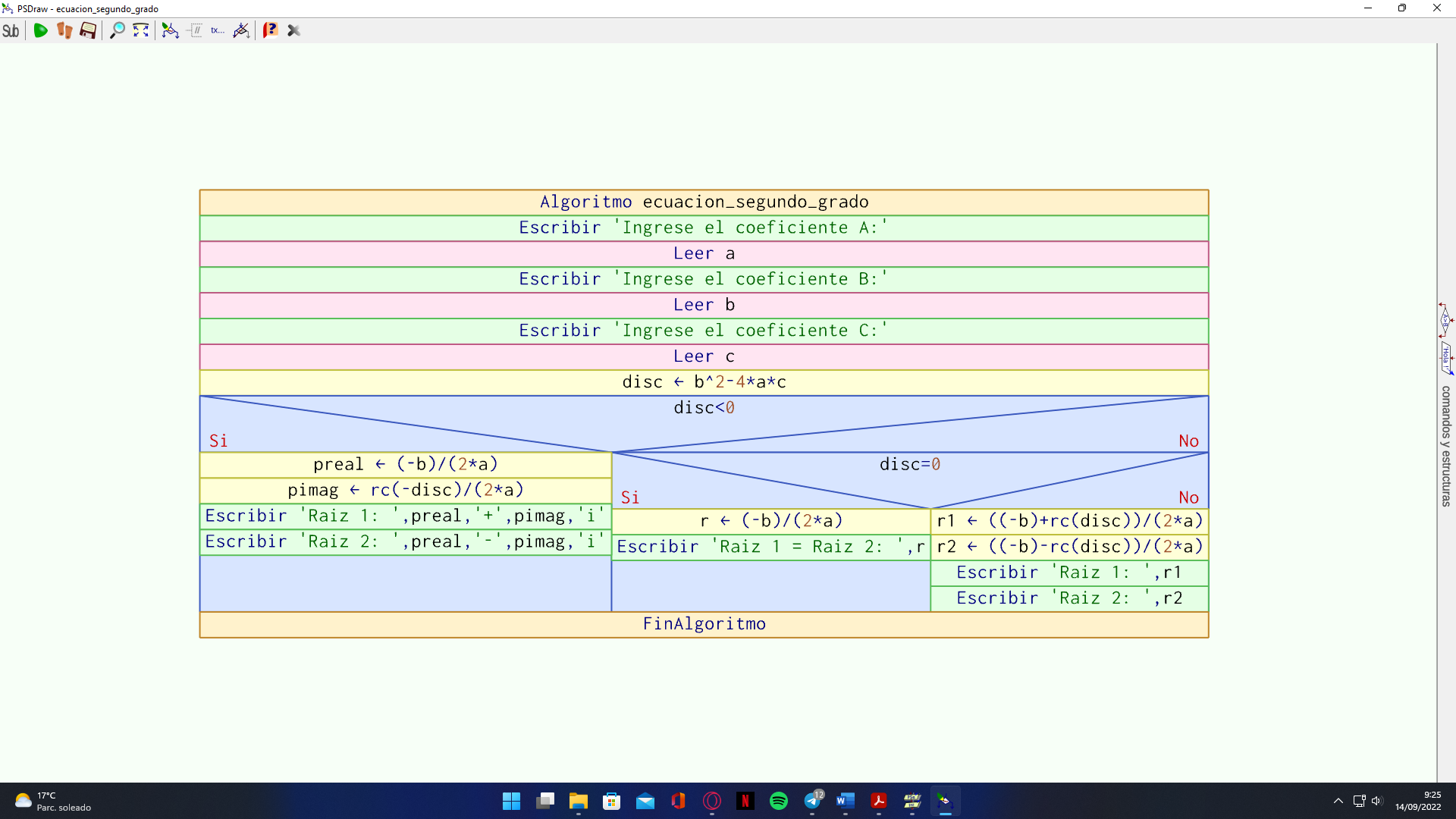
La representación en pseudocódigo del ejemplo de las raíces de ecuaciones de segundo grado es la siguiente:



## DIAGRAMAS DE CAJAS O DE CHAPIN O NASSI-SHNEIDERMAN

Son similares a los diagramas de flujo, pero omiten las flechas que representan el flujo, en su lugar se usan rectángulos los cuales contienen las instrucciones en forma de pseudocódigo u otro conjunto de rectángulos o símbolos para representar una condición o ciclo.

Su representación gráfica es la siguiente:



## DIAGRAMAS WARNIER

Permiten la descripción de los procedimientos y los datos de la organización.

Además, posibilitan diseñar un programa identificando primero su salida y, a partir de ahí, trabajar hacia atrás para determinar las combinaciones de pasos y de entradas para producirlas.

Su principal ventaja es su apariencia simple y sencilla de entender.

Los elementos que componen estos diagramas son:

|  |  |
| --- | --- |
| {  Conjunto | Delimita un bloque de información jerarquizada. De derecha a izquierda denota los niveles de abstracción. De arriba abajo muestra la secuencia y las relaciones lógicas entre las funciones. |
| (0,1)  Condicionalidad | La información contenida (variable o cantidad) indica el número de veces que ocurrirá el conjunto. Si se coloca la letra C el ciclo termina cuando la condición se cumpla. |
| +  Secuencia de acciones mutuamente excluyentes | Indica que una acción o grupo de acciones son mutuamente excluyentes dadas las condiciones que se establezcan. |

La representación gráfica de este tipo de diagramas es la siguiente:



## MÉTODO JACKSON

Es la representación de un programa en forma de árbol. También se llaman diagramas arborescentes de Jackson y constan de:

* Definición detallada de los datos de entrada y salida incluyendo los archivos lógicos utilizados.
* Representación del proceso o algoritmo.

La lectura del árbol se realiza en preorden raíz, subárbol izquierdo y subárbol derecho.

Los nodos se representan como rectángulos.

## MÉTODO BERTINI

Es otra forma de representación en árbol. Se denomina diagrama arborescente de Bertini y consta de:

* + Definición detallada de los datos de entrada y salida.
  + Representación del proceso o algoritmo.

La lectura se realiza en orden inverso: raíz, subárbol derecho y subárbol izquierdo.

Los nodos se presentan como círculos.



## MÉTODO TABOURIER

Se representa en forma de árbol con recorriendo en preorden y utilizando rectángulos y rombos (condicionales y bucles).

Se comienza con un rectángulo dividido horizontalmente con el nombre del programa en la parte superior y la palabra “BLOCK” en la inferior.

Consta de:

* + Definición detallada de los datos de entrada y salida incluyendo los archivos lógicos utilizados.
  + Representación del proceso o algoritmo.



## LENGUAJES DE PROGRAMACIÓN

Comparten las mismas características que el pseudocódigo, pero no son sólo comprensibles para los humanos sino también para los ordenadores.

Existen múltiples lenguajes de programación. La representación en lenguaje FORTRAN del algoritmo de las raíces de las ecuaciones de segundo grado sería el siguiente:

program segundoGrado

print \*,'Deme los coeficientes y resolveré una ecuación de 2º grado'

print \*, '¿Cuánto vale A?'

read \*, a

print \*, '¿Cuánto vale B?'

read \*, b

print \*, '¿Cuánto vale C?'

read \*, c

discr = b\*b-4\*a\*c

if (discr==0) then

s = -b/(2\*a)

print \*, 'Sólo hay una solución: ', s

else

if (discr>0) then

s1 = (-b+sqrt(discr))/(2\*a)

s2 = (-b-sqrt(discr))/(2\*a)

print \*, 'Las soluciones son: ',s1,s2

else

print \*, 'No hay soluciones reales'

end if

end if

end

# DESARROLLO E IMPLEMENTACIÓN DE UN ALGORITMO

Cuando se debe resolver un problema mediante la utilización de un ordenador no se codifica directamente un programa en un lenguaje dado. A pesar de que los lenguajes sean inteligibles para los humanos se alejan de nuestra forma de pensar y si el problema es complejo resulta imposible escribir un programa en un paso.

Por tanto, a la hora de solucionar un problema se deben llevar a cabo una serie de fases:

* **Primera fase o “análisis”.** Consiste en el estudio del problema a resolver, pero **no se buscará la forma de resolver el problema** sino comprender la naturaleza de este. ¿Qué datos son necesarios?, ¿En qué consistirá la solución?, ¿Qué errores puede presentar?... Una forma de enfocar el análisis es dividiéndolo en: fase de entrada, proceso, salida y condiciones erróneas.
* **Segunda fase o “diseño”.** Consiste en **buscar una forma de resolver el problema.** La mejor forma de afrontar esta fase es resolver el problema usando distintos niveles de abstracción:
  + **Nivel de abstracción 1.** En este nivel se plantea el problema usando términos del mismo problema.
  + **Nivel de abstracción 2.** Se descompone en varios subproblemas expresados en términos del problema y tratando de hacerlos lo más independientes entre sí que sea posible.
  + Esta fase se **repite para cada subproblema tantas veces como sea necesario hasta llegar a una descripción que emplee instrucciones sencillas que puedan ser codificadas fácilmente.**

En esta fase se pueden emplear **gráficos en árbol** para representar los diferentes subproblemas y las relaciones entre ellos o **pseudocódigo** para una descripción más “formal” de la resolución de cada subproblema.

# CONCEPTOS FUNDAMENTALES

* **Acción**: Suceso llevado a cabo por un procesador que modifica su entorno.
* **Indicador o variable**: El entorno de una acción está compuesto de indicadores que pueden tener distintos valores. Cada uno tendrá un nombre asignado.
* **Información**: Es la asociación entre una variable (v), un valor (70.5) y la interpretación que se hace (velocidad en km/h).
* **Estado**: Conjunto de valores de los indicadores en un instante. Por tanto, si una acción modifica su entorno, las acciones hacen pasar el entorno de un estado a otro.
* **Datos y resultados**: Las acciones emplean un tiempo finito para su ejecución. Suponiendo que una acción se inicia en t0 y termina en t1, el estado del sistema en t0 serían los datos de la acción mientras que el estado del sistema en t1 serían los resultados de la acción.
* **Léxico**: Conjunto finito de acciones que el procesador puede entender y ejecutar.
* **Tipo de datos**: Los datos se clasifica en función de ciertas características importantes: naturales, enteros, reales, complejos, etc. Estos conjuntos pueden estar formados por elementos individuales (R) o agregaciones de valores (Rn).
* **Operador**: El procesador conoce la forma de realizar asignaciones, comparar valores, etc. Dichas acciones son operadores y las acciones que realizan son operaciones.

# INTRODUCCIÓN DE LA NOTACIÓN ALGORÍTMICA A EMPLEAR

* **Variables**: Los nombres de las variables son identificadores y se expresan usando caracteres alfanuméricos incluyendo el subrayado (barra baja) y excluyendo los blancos (espacios) y caracteres no anglosajones (ñ, vocales acentuadas…).

Los identificadores tampoco podrán empezar por un número y en algunos lenguajes las mayúsculas se consideran como caracteres diferentes de las minúsculas (**lenguajes *key sensitive***). Por ejemplo,

* + Identificadores válidos: v, aceleracion, K, v1, b\_n, Pot…
  + Identificadores no válidos: 1n (empieza por número), año (incluye un carácter no válido, la ñ), aceleración (incluye un carácter no válido, la ó), p v (incluye un espacio en blanco) …

El **nombre** de las variables debe ser **representativos del valor que contienen**, de forma que no serán válidas variables con el nombre a o b, pero si dni o nombre\_cliente.

Estos nombres van precedidos de un indicador del tipo de datos que contenga. Por ejemplo:

* + **Enteros (int)**: intCoeficiente.
  + **Reales (rln)**: rlnNota.
  + **Carácter (chr)**: chrLetra.
  + **Cadena (str)**: strNombre.
  + **Booleandos (bln)**: blnBandera.
  + **Arrays (arr)**: arrRlnNotas1DAW.
* **Tipos de datos**: Se describen en función de las operaciones que admiten.
  + **El tipo entero**. Corresponde a un subconjunto finito de números enteros (el tamaño del rango varía en función del ordenador). Las operaciones soportadas son: suma (+), resta (-), producto (\*), división entera (div) y módulo/resto (mod). El resultado de operar dos valores de tipo entero resulta en otro valor de tipo entero con un valor exacto (salvo que resulte un valor fuera del rango).
  + **El tipo real**. Corresponde a un subconjunto de números reales. A diferencia de los valores enteros, los valores reales son valores aproximados susceptibles de errores de redondeo. Admite las mismas operaciones que el tipo entero salvo la división entera y el módulo (la división real emplea el operador /). Además, admiten las funciones reales: raíz cuadrada, exponenciación, potenciación, logaritmo y funciones trigonométricas. Al operar con valores de tipo real resultan valores de tipo real. Se puede operar entre valores de tipo entero y real y aplicar operaciones de tipo real al tipo entero, en todos estos casos, los resultados serán de tipo real.
  + **El tipo lógico o booleano**. Admite dos valores: verdadero y falso; y las operaciones booleanas: y-logico (and o ∧), o-lógico (or o ∨) y no-logico (not o ￢). Hay operadores que, aunque no operan con valores de tipo lógico dan resultados de dicho tipo, son los operadores de comparación.
  + **El tipo caracter**. Comprende el conjunto de caracteres imprimibles:
    - 26 letras del alfabeto latino. 10 dígitos arábicos. Caracteres como símbolos de puntuación.
    - Los subconjuntos de letras y números están ordenados; es decir si un carácter es mayor que ‘A’ y menor que ‘Z’ es una letra mayúscula, si es mayor que ‘a’ y menor que ‘z’ es minúscula y si es mayor que ‘0’ y menor que ‘9’ es un número.

Este tipo tiene una operación propia, la concatenación de cadenas de caracteres empleando el operador (+).

Todos los tipos admiten las operaciones de comparación:

* + Mayor (>).
  + Menor (<).
  + Mayor o igual (≥).
  + Menor o igual (≤).
  + Igual (=).
  + Distinto (≠).

El resultado de estas comparaciones es siempre un valor de tipo lógico.

* **Literales**: Son expresiones de valor constante de cualquier tipo. Hay literales de tipo entero (por ejemplo, 45, 0 o -17), de tipo real (por ejemplo, 3.141592, 10.5e20 o –5.24e–5), de tipo carácter (por ejemplo, ‘E’, ‘e’) y de tipo cadena (por ejemplo, “Esto es una cadena”).

En muchas ocasiones es necesario usar un valor literal de forma reiterada, en esos casos puede referirse a dicho valor mediante un identificado (por ejemplo, PI). Para ello se utilizan las denominadas **constantes**, identificadores a los que se les asigna un valor que no puede ser modificado en ningún punto del algoritmo. Se representan con letra mayúscula en todo el nombre.

* **Expresiones**: Son combinaciones de variables, constantes y literales de tipos compatibles entre sí. Estos elementos están combinados mediante operadores válidos. Es posible tener expresiones de cualquier tipo de dato, por ejemplo:

2 \* PI \* r

intVelocidad \* intTiempo

(intCont>5) and (intCont<10)

‘Hola’ + ‘mundo’

* **Asignación**: El operador de asignación (ß) se emplea según la notación siguiente:

izquierda ß derecha

El elemento situado a la izquierda es una variable a la que se le asigna el valor de la expresión de la derecha. Ambos elementos deben ser del mismo tipo.

* **Entrada/salida**: Los algoritmos necesitan de una interacción usuario-algoritmo, por ejemplo, es posible que el usuario deba indicar una serie de valores (como los coeficientes de una ecuación de segundo grado) o que el algoritmo deba proporcionar unos resultados al usuario (las posibles soluciones de la ecuación). Las acciones de entrada y salida son: leer y escribir.

El formato de leer es: leer variable. El resultado de la acción es la lectura de un valor por teclado que será asignado a variable.

El formato de escribir es: escribir expresión (expresión puede ser de cualquier tipo). El resultado de la acción es la visualización de la expresión.