UNIDAD 1: IDENTIFICACIÓN DE LOS ELEMENTOS DE UN PROGRAMA INFORMÁTICO

Módulo Profesional: Programación

Índice

RESUMEN INTRODUCTORIO

A lo largo de esta unidad veremos una visión histórica de la informática y la programación de los ordenadores. También explicaremos el concepto de algoritmo y cómo se pueden representar, conoceremos los distintos lenguajes de programación, los distintos datos existentes y sus tipos, la definición de variable y constante. Seguiremos viendo las expresiones y los distintos tipos de operadores existentes. Por último, veremos qué significa la entrada y salida de información y el concepto de programa, instrucciones y elementos del mismo.

INTRODUCCIÓN

En la actualidad todo el mundo utiliza la tecnología para casi todos los ámbitos de sus vidas. El software está presente no solo en los ordenadores, sino en los dispositivos móviles en los cuales almacenamos gran parte de aquella información que necesitamos prácticamente a diario: números de teléfono, número de la cuenta del banco, avisos para las tareas que tenemos que realizar día a día, las fechas de los cumpleaños de las personas más cercanas a nosotros, podemos acceder a un mundo de información a través de las conexiones a Internet, etc. Realmente tenemos dos opciones: podemos ser simplemente consumidores de todo ese software o podemos ser una de las personas que crea esos programas. Los programadores pueden dividir un problema grande en problemas más pequeños. Eso es pensar de manera diferente, tal y como expresaba Steve Jobs. Hay muchas personas que de forma autodidacta aprenden algún lenguaje de programación para hacer sus propios programas. La tarea de sentarse delante del ordenador para realizar un programa requiere además de mucha paciencia, unos conocimientos básicos que le permitan saber que lo que se está creando va a funcionar. Aprender a programar puede ser fácil si empezamos desde lo más básico y se va evolucionando de forma gradual hasta que somos capaces de realizar un programa bien construido y que funciona. La programación se aprende practicando, como se aprende a montar en bicicleta o a realizar experimentos en un laboratorio o a resolver problemas de matemáticas. Los primeros pasos consisten en aprender a escribir un algoritmo, elegir un lenguaje de programación que se adapte a nuestras necesidades y traducir ese algoritmo a ese lenguaje. Una vez que sepamos programar en un lenguaje, el hecho de aprender otros lenguajes de programación supone una tarea mucho más fácil que empezar desde cero. Para aprender todos estos conceptos básicos de programación se ha elegido el lenguaje Java, por su sintaxis sencilla y por su gran potencia como lenguaje orientado a objetos.

CASO INTRODUCTORIO

Trabajas en una empresa de desarrollo de software que está desarrollando un proyecto. Estás en la fase de codificación y lo primero en lo que piensas es en teclear el código directamente en el lenguaje de programación elegido, pero antes de nada es necesario realizar un boceto de lo que tienes que hacer para estar perfectamente planificado. Por ello, realizas el algoritmo en lenguaje natural para resolver el problema.

Al finalizar la unidad el alumnado:

- ✓ Será capaz de realizar dado un problema el algoritmo que lo resuelve.
- ✓ Conocerá los distintos métodos de representación de algoritmos.
- ✓ Utilizará los diagramas de flujo, diagramas N-S y pseudocódigo para realizar el paso previo al desarrollo del programa en un lenguaje de programación.
- ✓ Conocerá el concepto de programa y los componentes de éste.

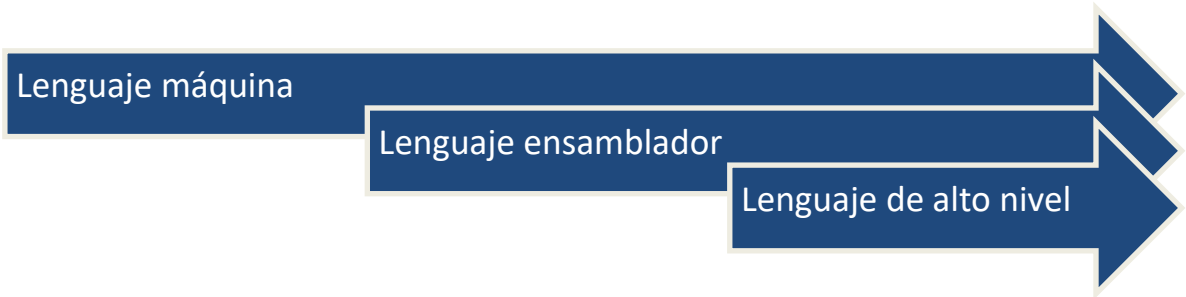
1.VISIÓN HISTÓRICA

La **Informática** se define como la ciencia del tratamiento automático y racional de la información, así como el soporte del conocimiento y las comunicaciones. Al adentrarnos en el estudio del mundo de la informática se puede comprobar que existe un amplio abanico de áreas y especialidades. Por ello, antes de empezar a estudiar cualquiera de ellas, es necesario asentar unas bases o pilares sobre los que sustentar los futuros conocimiento que se vayan adquiriendo.

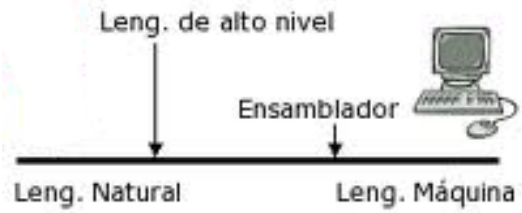
De la misma forma que el comportamiento y el pensamiento humano se rigen por métodos de razonamiento lógicos que nos permiten la ejecución de acciones o tareas concretas, el comportamiento o actuación de un ordenador se rige por lo que denominamos programación, entendiendo como tal el planteamiento, desarrollo y puesta en marcha de soluciones a problemas concretos, mediante una secuencia de instrucciones o conjunto de acciones lógicas que debe ejecutar el ordenador y que son transmitidas a éste por la figura del programador en forma de programa.

Cuando se empezaron a diseñar las primeras computadoras, el único lenguaje de programación del que se disponía era el lenguaje máquina. Estos lenguajes eran difíciles de leer y modificar, por lo que se desarrollaron lenguajes ensambladores, que facilitaban el trabajo a los programadores. Dichos lenguajes están definidos por un código nemotécnico que establece una cierta equivalencia entre las instrucciones del código máquina y las instrucciones del lenguaje ensamblador.

Posteriormente aparecieron los lenguajes de programación de alto nivel (como Pascal o C), con los que el programador puede escribir un programa aun sin conocer extensivamente la máquina en que dicho programa va a ser ejecutado. El lenguaje en estos casos es independiente de la máquina.



Un lenguaje de alto nivel está más cercano a los lenguajes naturales (generalmente el inglés), teniendo así menos limitaciones.



2.LA PROGRAMACIÓN DE ORDENADORES

Los ordenadores no son inteligentes por sí mismos, todo lo que hacen es porque alguien le ha indicado qué tiene que hacer. La forma de comunicar a un ordenador qué debe hacer es, generalmente, mediante el uso de programas que contienen las instrucciones e información necesaria para resolver el problema planteado.

Para comunicarle al ordenador las instrucciones a ejecutar se debe utilizar un lenguaje que pueda entender. Existen muchos lenguajes de programación, pero todos ellos tienen en común una serie de normas semánticas y sintácticas.

Con el objeto de facilitar el planteamiento, desarrollo y puesta en marcha de los programas, surge la ***metodología de la programación*** (en sus dos vertientes más destacadas, la metodología estructurada y modular, también conocida como metodología tradicional y por otro lado la metodología orientada a objetos), que facilita el estudio de programas de forma abstracta permitiendo el desarrollo de los mismos sin tener que utilizar las reglas sintácticas de un determinado lenguaje de programación, proporcionando la lógica necesaria para especificar las estructuras de datos que hay que utilizar y los algoritmos necesarios para el tratamiento de los mismos.

CITA

"Se suele decir que una persona no entiende algo de verdad hasta que puede explicárselo a otro. En realidad, no lo entiende de verdad hasta que puede explicárselo a un computador". Donald Knuth.

La esencia de la metodología de la programación se basa en el empleo de técnicas de **programación estructurada,** que consisten en:

1. Desarrollar un programa en módulos con un diseño descendente.
2. Emplear únicamente tres clases de estructuras básicas de control (secuencial, alternativa y repetitiva) en el desarrollo de cada módulo.

Con el objeto de facilitar el desarrollo, depuración y futuro mantenimiento de los programas, permitiendo el uso de parte de los programas ya existentes, a finales de la década de los 50 y comienzo de los 60 se desarrollaron los denominados lenguajes de programación con técnicas estructuradas.

Años después, en la década de los años 80 y utilizando como pilar la programación estructurada, surge la **programación orientada a objetos**, con nuevos lenguajes de programación (***Lenguajes Orientados a Objetos*****)** que dotan al programador de nuevos elementos para el análisis y desarrollo del software, aportando un nuevo enfoque al mundo de la programación.

En la actualidad se tiene, por otra parte, el concepto de desarrollo web. Este tipo de desarrollo utiliza básicamente lenguajes de programación (PHP, JSP, ASP.NET…), lenguajes de marcas (HTML) y bases de datos (MySQL, Oracle, SQL Server…), para la creación de páginas web. Para conseguir todo esto se usan tecnologías de programación del lado del cliente y del lado del servidor.

ENLACE DE INTERÉS

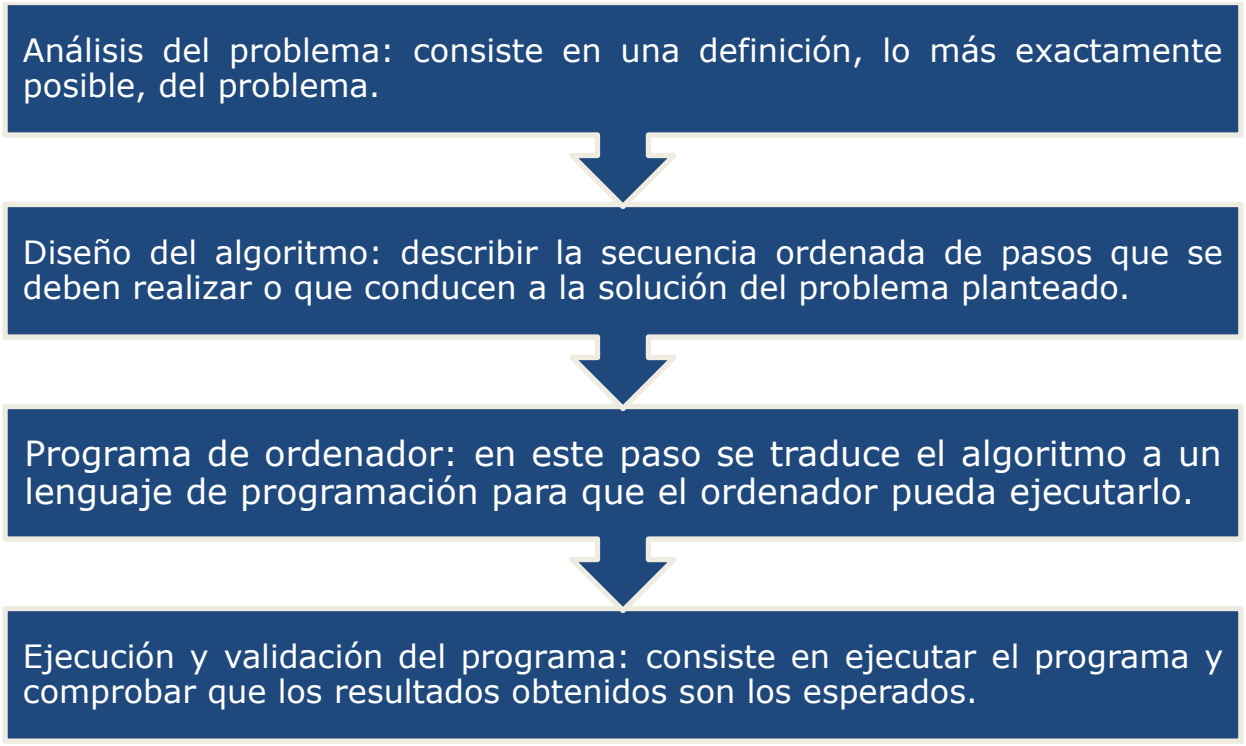
Se puede ampliar información sobre la evolución de los lenguajes de programación, así como sus principales características en la web:

http://es.wikipedia.org/wiki/Historia_de_los_lenguajes_de_programaci%C3%B3n

3.CONCEPTO DE ALGORITMO

Un algoritmo es un conjunto de acciones u operaciones a realizar por el ordenador, de forma clara y detallada, así como el orden en que deben ejecutarse, que nos conducen a la solución del problema, facilitando así su traducción posterior al lenguaje de programación correspondiente.

La resolución de todo problema exige el diseño de un algoritmo, y los pasos para dicha resolución son:



Los algoritmos son independientes, tanto del lenguaje de programación al que serán traducidos como del ordenador en que se ejecutarán.

Características del algoritmo

Las características fundamentales que debe cumplir todo algoritmo son:

1. Debe ser conciso y detallado: reflejar *al máximo detalle* el orden de ejecución de cada acción que vaya a realizar el ordenador.
2. Debe ser exacto o preciso: si se sigue el mismo n veces con los mismos datos de entrada, se deben obtener *siempre los mismos resultados.*
3. Debe ser finito o limitado: debe terminar en algún momento.

4. No debe ser rígido en su diseño, debe ser flexible a las diferentes representaciones gráficas y facilitar las modificaciones o actualizaciones del diseño realizado.
5. Debe ser claro y sencillo para facilitar su entendimiento al programador.

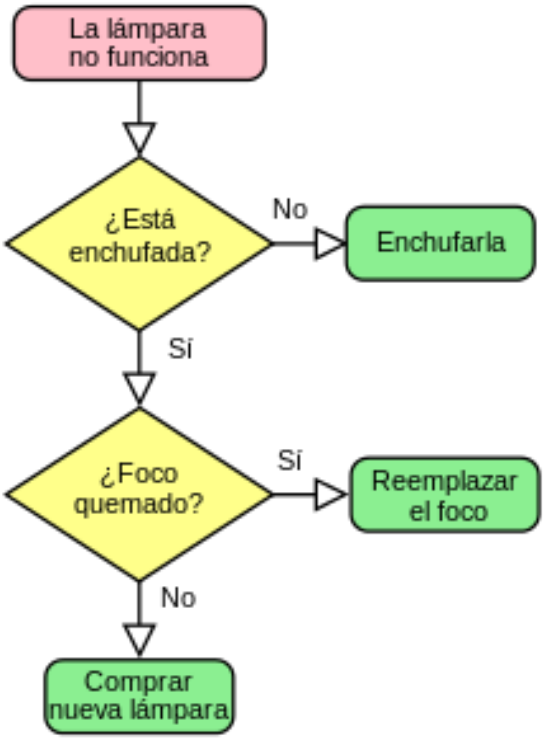


Ilustración: ejemplo algoritmo para resolver un problema de la vida diaria

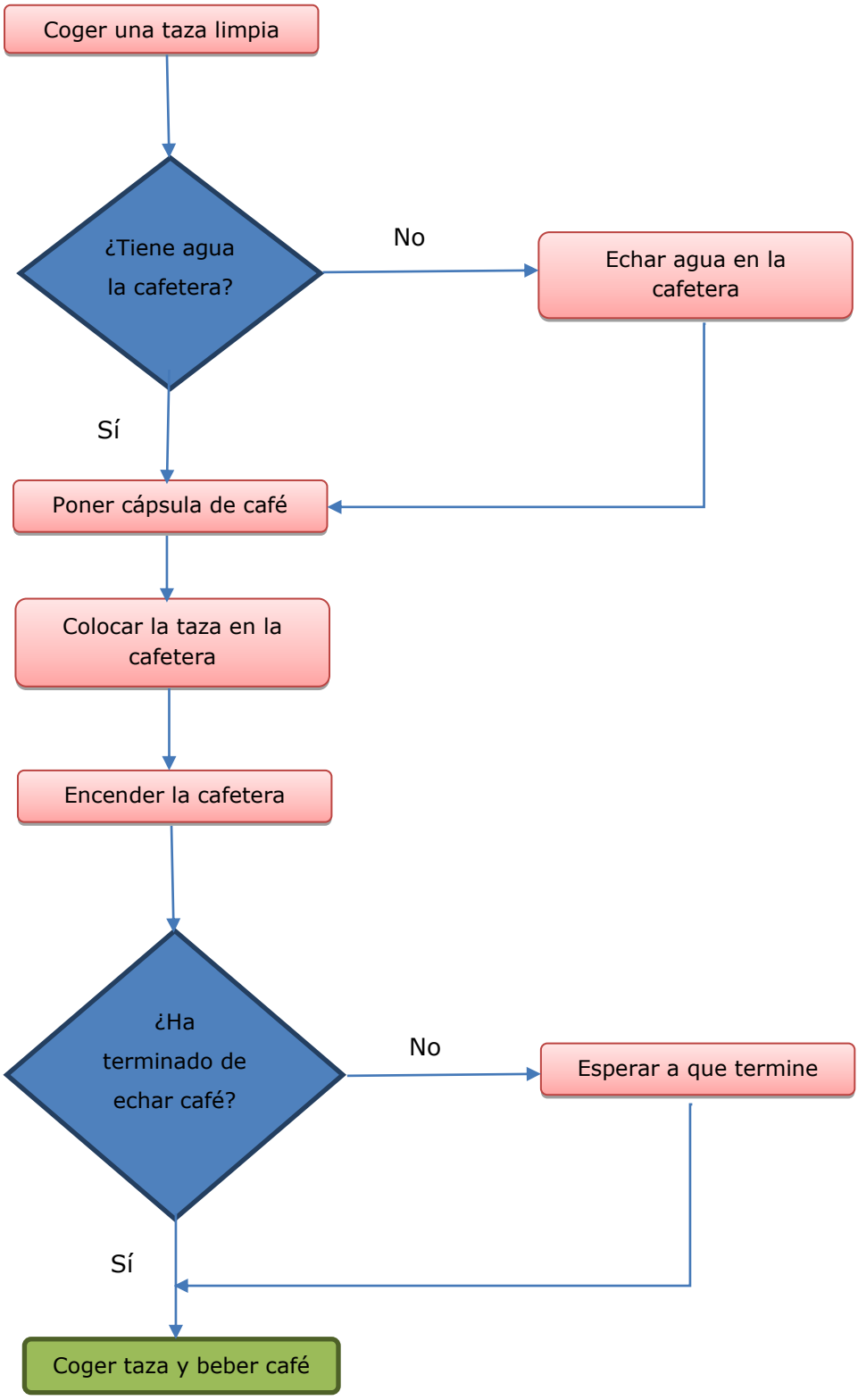


Ilustración: ejemplo algoritmo para resolver un problema de la vida diaria: preparar un café

ARTÍCULO DE INTERÉS

Se recomienda leer la historia de Ada Lovelace, considerada como la primera programadora de computadoras:

http://es.wikipedia.org/wiki/Ada_Lovelace

ENLACE DE INTERÉS

Si desea obtener más información de los algoritmos, podrá consultar en el siguiente enlace:

https://es.wikipedia.org/wiki/Algoritmo

ENLACE DE INTERÉS

En el siguiente enlace podrá obtener diversos cursos sobre algoritmia:

http://www.lawebdelprogramador.com/cursos/Algoritmia/index1.html

4.LENGUAJES DE PROGRAMACIÓN

Una vez definido, el algoritmo debe ser suministrado al procesador, el cual debe ser capaz de interpretarlo, lo que significa que debe comprender las instrucciones de cada paso y realizar las operaciones correspondientes.

Para corregirlo, cuando el procesador es un ordenador, el algoritmo deberá ser expresado en un formato denominado programa, que se escribe en un lenguaje de programación.

Un lenguaje de programación es una colección de símbolos y caracteres combinados entre sí con una sintaxis ya definida, para permitir transmitir instrucciones a la computadora.

Existen 3 tipos principales de lenguajes de programación:

Lenguaje máquina o binario

- Aquellos que están escritos en *lenguajes directamente inteligibles por la máquina*. Es decir: mediante cadenas binarias, que son secuencias de ceros y unos. Un inconveniente es que el lenguaje máquina depende del hardware.

Lenguaje de bajo nivel o ensamblador

- Son más fáciles de usar que los lenguajes máquina, pero, al igual que el lenguaje máquina, depende de la máquina en particular. Al programar en bajo nivel, el programa tiene que ser traducido a binario para que lo entienda la máquina. El programa de bajo nivel se llama "programa fuente" y el traducido al binario "programa objeto.

Lenguaje de alto nivel

- Son los que se utilizan a día de hoy. Por sus características se encuentran más próximos al usuario o programador. Una de las características más importantes es que son independientes de la arquitectura del ordenador. Estos lenguajes no se pueden ejecutar directamente en el ordenador, tienen que ser traducidos a lenguaje máquina. También tenemos "programa fuente" en alto nivel y se traduce a "programa objeto" por medio de dos tipos de programas: compiladores e intérpretes. La principal diferencia entre un compilador y un intérprete es que el intérprete acepta un programa fuente que traduce y ejecuta simultáneamente analizando cada instrucción por separado, y el compilador efectúa dicha operación en dos fases independientes: primero traduce todo y a continuación lo ejecuta.

Una de las primeras y principales distinciones que se pueden hacer al clasificar los lenguajes de programación es si el lenguaje es compilado o interpretado.

Un lenguaje de programación se dice que es **compilado** cuando existe un traductor que recoge el programa/aplicación realizada por un programador y lo compila para un determinado sistema operativo (o arquitectura de forma más exacta), es decir lo convierte en un ejecutable que un sistema operativo (arquitectura) entiende y puede ejecutarlo.

Ejemplos de lenguajes de programación compilados son: C, C++, C#, Swift, Objective-C...

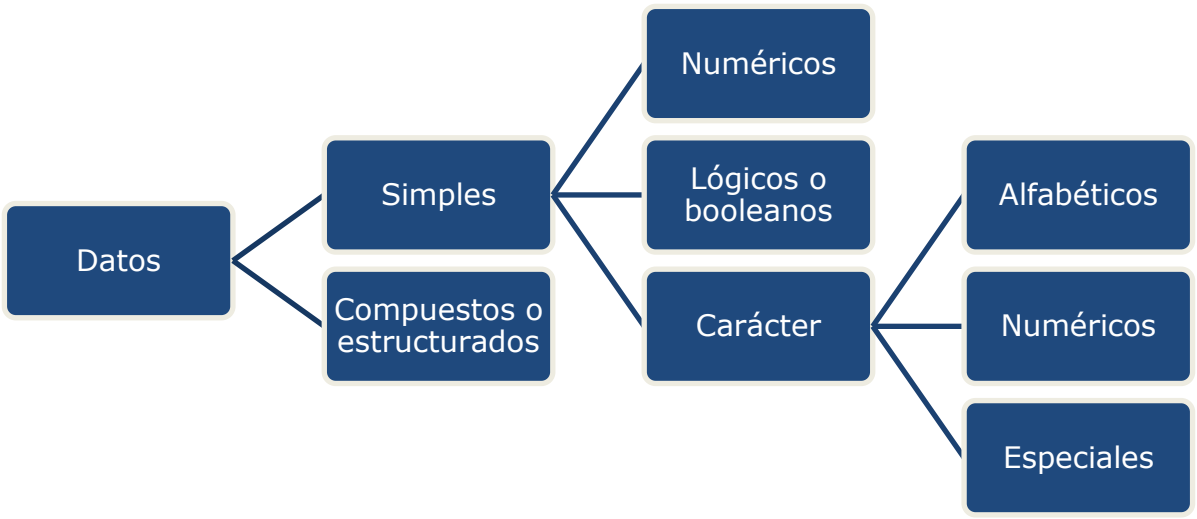
Los lenguajes interpretados, tal y como indica la palabra, son lenguajes de programación que necesitan de un intérprete para ser ejecutados. Ejemplos de estos lenguajes son el lenguaje PHP, JavaScript, Ruby…

El caso del lenguaje **Java** es un caso especial, ya que se encuentra entre lo que es un lenguaje compilado y uno interpretado. Java no realiza la compilación pura como lo realizan el resto de los lenguajes compilados, sino que realiza un paso intermedio que consiste en que el compilador traduce el código de alto nivel a un código intermedio denominado **bytecode**. Este código intermedio es posible distribuirlo a cualquier plataforma, pero para poder ser ejecutado, el ordenador destino deberá tener instalado un ejecutor denominado Java Virtual Machine, el cual sí que es dependiente del sistema operativo y por lo tanto del hardware.

En la actualidad los lenguajes de programación más populares son: Java, C, C++, Python, Visual Basic .NET, C#, PHP, JavaScript, SQL, Objective-C, Swift, Ruby y Kotlin.

5.DATOS Y TIPOS DE DATOS

El primer objetivo de todo ordenador es el manejo de datos. Un dato es la expresión general que describe los objetos con los cuales opera un ordenador. La mayoría de los ordenadores pueden trabajar con varios tipos de datos. Estos datos se suministrarán al programa, el cual los procesará y transformará en datos de salida o información.



ENLACE DE INTERÉS

Para ampliar información sobre los tipos de datos de uno de los lenguajes de programación de alto nivel más usados en la actualidad: lenguaje C, visite:

http://recursostic.educacion.es/observatorio/web/es/software/programacion/972-tipos-de-datos

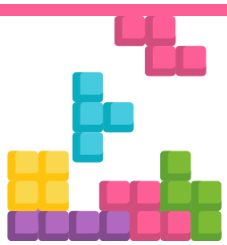
El lenguaje de programación Java utiliza tipos de datos primitivos y complejos. Java nos proporciona 8 tipos de datos primitivos, que se pueden poner en cualquier lugar del código fuente de Java.

TIPO DE VARIABLE	BYTES QUE OCUPA	RANGO DE VALORES
boolean	2	true, false
byte	1	-128 a 127
short	2	-32.768 a 32.767
int	4	-2.147.483.648 a 2.147.483.649
long	8	-9 · 10^18 a 9 · 10^18
double	8	-1,79 · 10^308 a 1,79 · 10^308
float	4	-3,4 · 10^38 a 3,4 · 10^38
char	2	Caracteres (en Unicode)

6. CONSTANTES Y VARIABLES

Los programas de ordenador contienen ciertos valores que no deben cambiar durante la ejecución del programa, estos valores se denominan **constantes**. Hay otros valores que cambian durante la ejecución del programa, para almacenar dichos valores se utilizan **variables**. En ambos casos lo que estamos haciendo es reservar un espacio en memoria en el cual se almacenarán el valor que tome la variable o el valor de la constante para cada uno de los identificadores definidos.

```
tipo_de_datos identificador;
```



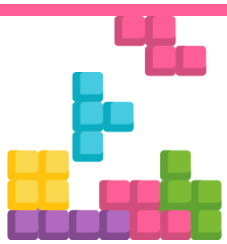
EJEMPLO PRÁCTICO

En Java, la forma más simple de representar una variable es la siguiente:

```
int contador;
```

Para representar una **constante** en Java, se utilizará:

```
static final tipo_de_datos identificador;
```



EJEMPLO PRÁCTICO

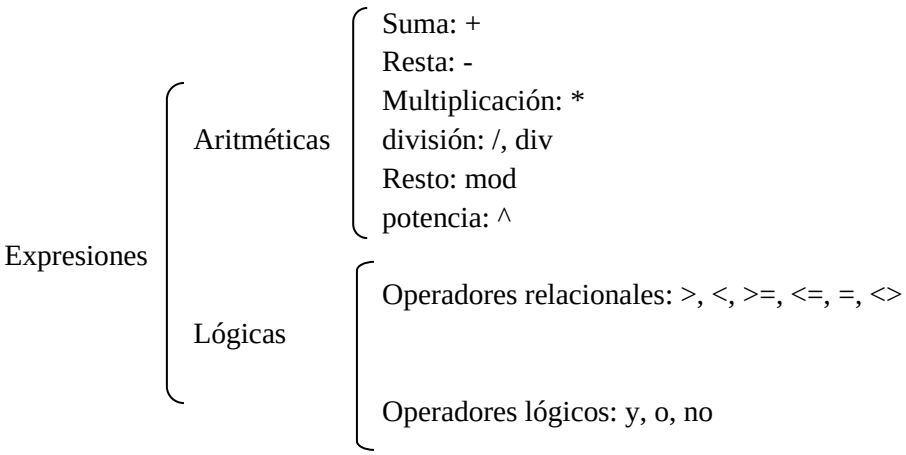
En este ejemplo se asigna el valor 250 a la constante NUMERO (en Java las constantes se escriben en mayúsculas por convención).

```
static final int NUMERO = 250;
```

7. EXPRESIONES

Una expresión es una combinación de constantes, variables, símbolos de operaciones, paréntesis y nombres de funciones especiales, de cuya evaluación se obtiene un único resultado. Dependiendo del resultado obtenido en la evaluación de una expresión se puede hablar de expresiones aritméticas y expresiones lógicas.

- **Expresión aritmética** es aquella compuesta por operaciones matemáticas.



En Java los operadores anteriores se representan de la siguiente forma:

Aritméticos

- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- Resto: %

Relacionales

>, <, >=, <=, ==, !=

Lógicos

&&, ||, !

- **Expresión lógica**: el resultado de su evaluación es un valor lógico o booleano. Pueden usar operadores relacionales u operadores lógicos.

- Un ***operador Y*** devuelve V sólo cuando los 2 operandos sean verdaderos.

Operando 1	Operando 2	Resultado
V	V	**V**
V	F	**F**
F	V	**F**
F	F	**F**

- Un ***operador O*** devuelve V siempre que algún operando sea verdadero.

Operando 1	Operando 2	Resultado
V	V	**V**
V	F	**V**
F	V	**V**
F	F	**F**

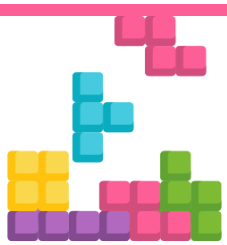
- ***Operador NO*** es un operador unario (sólo 1 operando). Devuelve el valor opuesto al operador.

Operando	Resultado
V	**F**
F	**V**

Precedencia de operadores: Cuando una operación está compuesta por distintos tipos de operadores, para evaluarla hay que seguir un orden de prioridad:

1º ( )
2º ^ , no
3º * , / , div , mod, y
4º + , - , o
5º operaciones relacionales

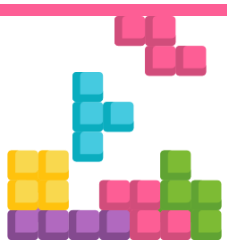
Si todos los operadores tienen el mismo grado de prioridad se evalúa de izquierda a derecha.



EJEMPLO PRÁCTICO

En Java se usan los operadores aritméticos de la siguiente forma:

```
double op1, op2, result1;
int op3, op4, result2;
op1=20.0;
op2=4.0;
op3=20;
op4=6;
result1=op1+op2;
      System.out.println("La suma de op1 y op2 es = "+result1);
result2=op3/op4;
      System.out.println("La división de op3 entre op4 ="+result2);
result2=op3%op4;
      System.out.println("El resto de dividir op3 entre op4
                                                = "+result2);
```



EJEMPLO PRÁCTICO

En Java se usan los operadores lógicos y relacionales de la siguiente forma:

```
int op1, op2, result;
op1=20;
op2=4;
if (((op1/op2)>4) && (op2>0))
      System.out.println("El resultado es mayor que 4");
else
      System.out.println("El resultado es menor que 4");
```

8. OPERACIONES DE ASIGNACIÓN

Es una de las formas de proporcionar valores a una variable. Se representa con el operador: ⇐

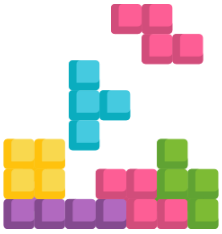
Por ejemplo:

Var
    a, b, c: entero
    a ⇐ (3+b)*c

En java:

```
int a,b,c;
a=(3+b)*c;
```

La operación de asignación es una operación *destructiva*, ya que el valor que tuviese anteriormente la variable será modificado por el valor obtenido o asignado.

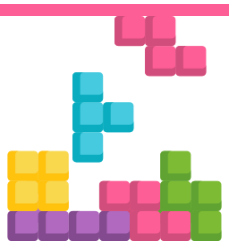


EJEMPLO PRÁCTICO

Se tienen 3 variables: a, b, c. Escribir las instrucciones necesarias para intercambiar entre si su valor del siguiente modo:

- b debe tomar el valor de a
- c debe tomar el valor de b
- a debe tomar el valor de c

Var
    a, b, c, aux: entero
    aux ⇐ b
    b ⇐ a
    a ⇐ c
    c ⇐ aux



EJEMPLO PRÁCTICO

En Java, el ejemplo anterior queda de la siguiente forma:

```
int a, b, c, aux;
    aux = b;
b = a;
a = c;
c = aux;
```

ENLACE DE INTERÉS

Puede ampliar información sobre el uso de los operadores en el siguiente enlace:

https://www.arkaitzgarro.com/java/capitulo-4.html

9. ENTRADA Y SALIDA DE INFORMACIÓN

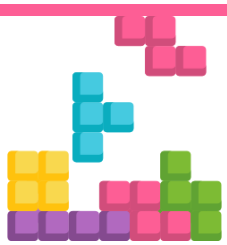
Los cálculos que realizan los ordenadores requieren, para ser útiles, la entrada de los datos necesarios para ejecutar las operaciones que posteriormente se convertirán en resultados o salidas.

Las operaciones de entrada permiten **leer determinados valores** introducidos por el usuario y **asignarlos** a determinadas variables.

Esta entrada se conoce como **operación de lectura o entrada**.

Con los datos de entrada, el ordenador realiza las operaciones oportunas en cada momento y el resultado obtenido aparece en un dispositivo de salida mediante una **operación de salida o escritura**.

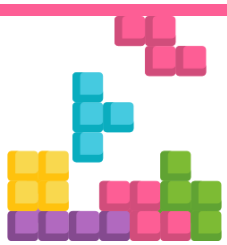
En **Java,** para las operaciones de salida, se utilizan los métodos **System.out.print**, o **System.out.println** (realiza la misma tarea que el método anterior, pero añade un salto de línea al final del texto).



EJEMPLO

```
int result, valor;
valor = 1540;
result = valor * 25 /100;
System.out.println("El resultado de la
                    operación es "+result);
```

Para las operaciones de entrada o lectura en Java, se utilizará por ahora el método **next()** de la clase **java.util.Scanner**.



EJEMPLO

```
Scanner sc = new Scanner(System.in);
String valor = sc.next();
System.out.println("El valor introducido es "
                                    +valor);
```

10. REPRESENTACIÓN DE ALGORITMOS

Hay varias formas de representar las instrucciones y el orden en un algoritmo:

1. **Diagramas de flujo:** es una de las técnicas de representación de algoritmos más antigua. Se basa en la utilización de símbolos gráficos denominados **cajas**, en las que escribimos las acciones que tiene que realizar el algoritmo. Estas cajas están conectadas mediante **líneas de flujo** que indican el orden de ejecución de las acciones:

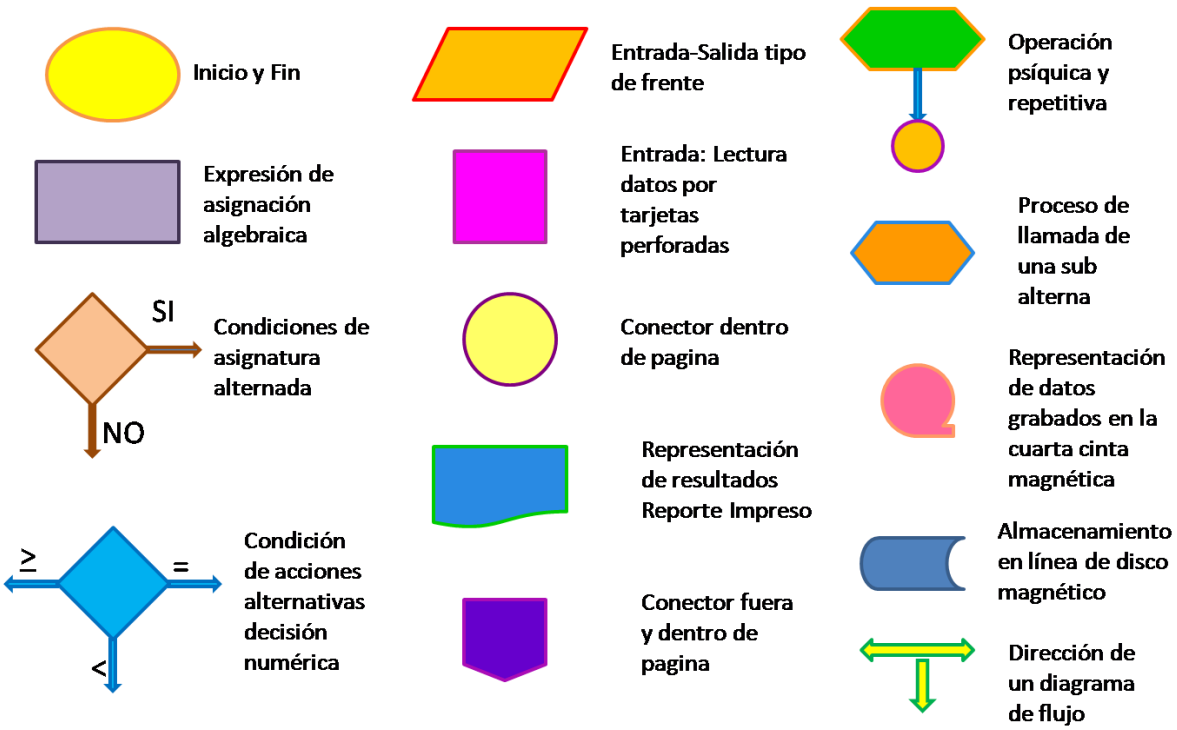


Ilustración: elementos de un diagrama de flujo

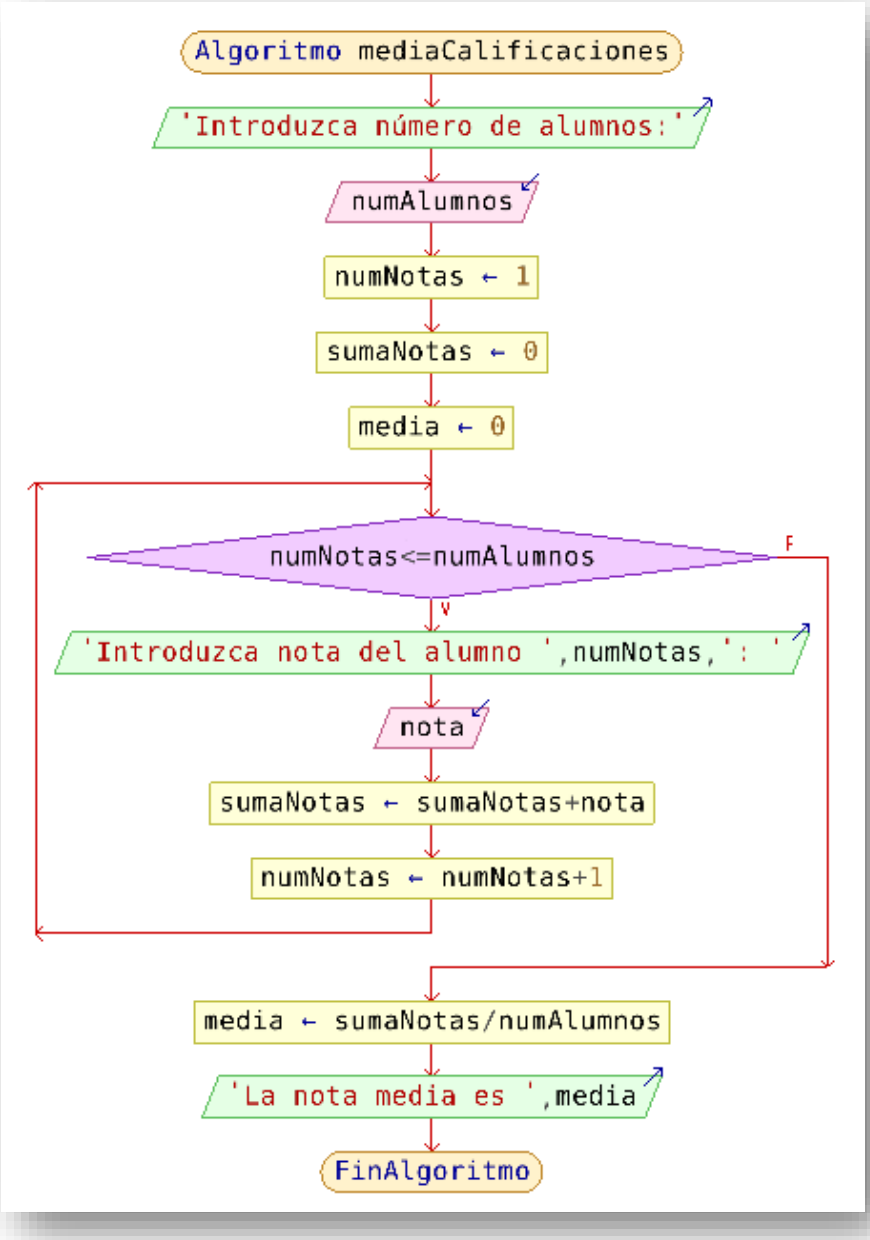


Ilustración: ejemplo de diagrama de flujo

ENLACE DE INTERÉS

Puede ampliar información sobre los Diagramas de Flujo, podrá consultar la siguiente web:

http://www.areatecnologia.com/diagramas-de-flujo.htm

2. **Diagramas de N-S (Nassi-Shneiderman):** es idéntico al diagrama de flujo, eliminando las líneas de flujo, con las **cajas contiguas** unas a otras.

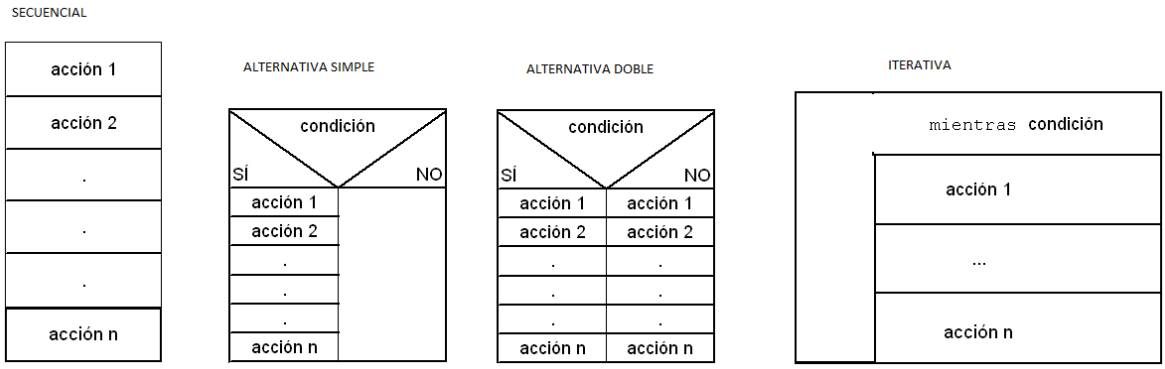


Ilustración: elementos diagramas N-S

```
Algoritmo mediaCalificaciones
Escribir 'Introduzca número de alumnos:'
Leer numAlumnos
numNotas ← 1
sumaNotas ← 0
media ← 0
Mientras numNotas<=numAlumnos
    Escribir 'Introduzca nota del alumno ',numNotas,': '
    Leer nota
    sumaNotas ← sumaNotas+nota
    numNotas ← numNotas+1
media ← sumaNotas/numAlumnos
Escribir 'La nota media es ',media
FinAlgoritmo
```

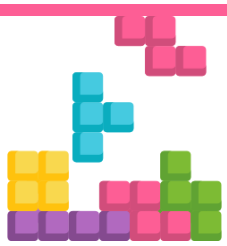
Ilustración: ejemplo diagrama N-S que calcula la nota media de un número de alumnos

ENLACE DE INTERÉS

Se recomienda ampliar información sobre estos diagramas en la web:

https://es.wikipedia.org/wiki/Diagrama_Nassi-Shneiderman

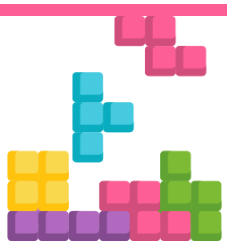
3. **Pseudocódigo:** este tipo de representación es el más utilizado, debido a que es el que más simplifica el paso de codificación al lenguaje de programación correspondiente.



EJEMPLO PRÁCTICO

Algoritmo que calcula la nota media de un número de alumnos introducido por teclado.

```
Algoritmo mediaCalificaciones
	Escribir 'Introduzca número de alumnos:'
	Leer numAlumnos
	numNotas<-1
	sumaNotas<-0
	media<-0
	Mientras numNotas<=numAlumnos Hacer
		Escribir 'Introduzca nota del alumno ',numNotas,': '
		Leer nota
		sumaNotas<-sumaNotas+nota
		numNotas<-numNotas+1
	Fin Mientras
	media<-sumaNotas/numAlumnos
	Escribir 'La nota media es ',media
FinAlgoritmo
```



EJEMPLO PRÁCTICO

Diseñar un algoritmo que permita calcular el área y el perímetro de una circunferencia, sabiendo que el radio será un dato introducido por el usuario.

```
Algoritmo area_y_perímetro
	numeropi<-3.1416
	escribir ("Introduzca la medida del radio")
	leer radio
	area <- numeropi*radio^2
	perimetro <- 2*numeropi*radio
	escribir 'El area de la circunferencia es ', area, ' y el
                                        perímetro es ', perimetro
FinAlgoritmo
```

ENLACE DE INTERÉS

Puede ampliar información sobre la representación de algoritmos visitando la web:

https://kesquivel.files.wordpress.com/2011/03/estructurado1.pdf

10.1 Software de utilidad

Actualmente existe una gran cantidad de software para la elaboración de **Diagramas de flujo**. Se recomienda el uso de los siguientes programas para la elaboración de Diagramas de Flujo:

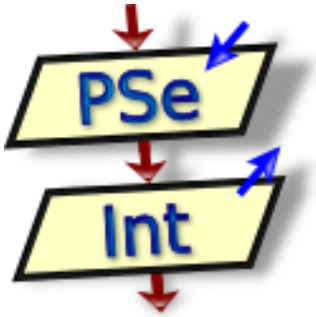
- **Software Comercial**: *Microsoft Office* ofrece 3 herramientas útiles para la elaboración de diagramas:
    - **Word**: Permite crear diagramas de flujo básicos a través de la opción de "Formas" que tiene un apartado especial para diagramas de flujo.
    - **PowerPoint**: Ofrece las mismas posibilidades de creación de diagramas.
    - **Visio**: Herramienta más sofisticada. Además de la simbología básica de diagramas de flujo, cuenta con una variedad de herramientas para elaborar otros tipos de diagramas.
- **Software No Comercial**: *DIA*. Solución rápida para la creación de diagramas de flujo además de otros tipos de diagramas.

ENLACE DE INTERÉS

Para la instalación del software DIA, se recomienda visitar la web oficial:

https://sourceforge.net/projects/dia-installer/

A la hora de trabajar con **Pseudocódigo**, también se dispone de herramientas gráficas que permiten el desarrollo de la materia. En este caso se dispone del software *PSeInt*, el cual es un software libre educativo multiplataforma dirigido a personas que se inician en la programación.



Dicho software está pensado para iniciarse en la construcción de programas o algoritmos computacionales. El pseudocódigo se suele utilizar como primer contacto para introducir conceptos básicos como el uso de estructuras de control, expresiones, variables, etc., sin tener que lidiar con las particularidades de la sintaxis de un lenguaje real.

Este software pretende facilitar la tarea de escribir algoritmos en pseudolenguaje presentando un conjunto de ayudas y asistencias, y brindando además algunas herramientas adicionales que ayudan a encontrar errores y comprender la lógica de los algoritmos.

ENLACE DE INTERÉS

Para la instalación del software *PSeInt*, se recomienda visitar la web oficial:

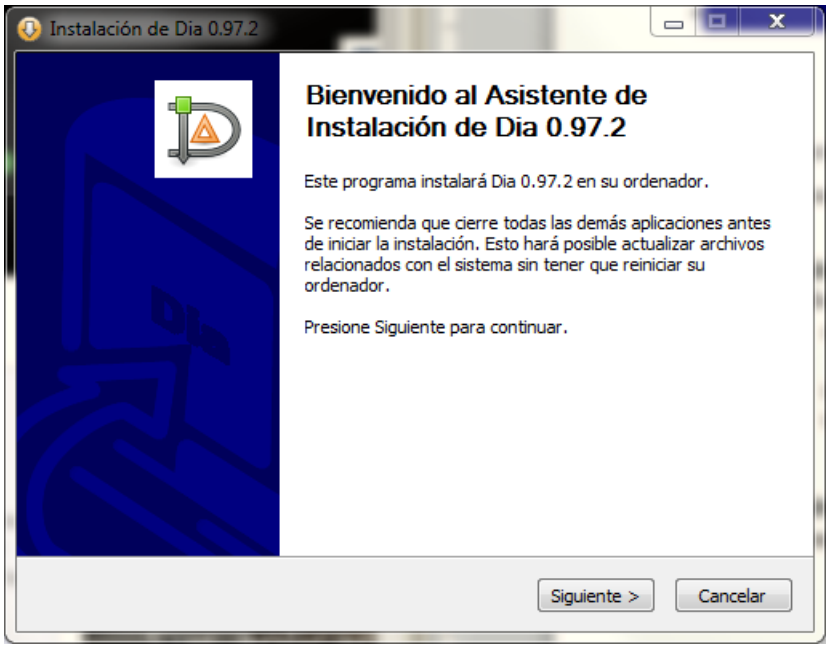
http://pseint.sourceforge.net/index.php?page=descargas.php

Instalación del Software No Comercial DIA

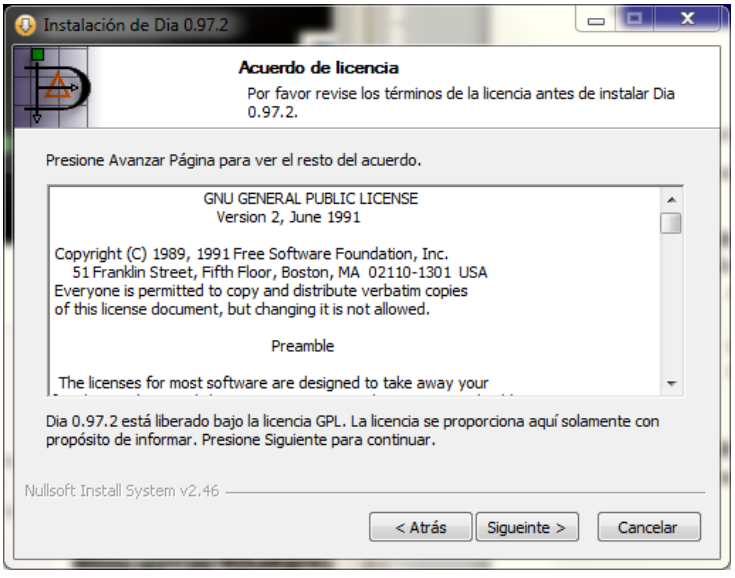
Se accede a la web oficial **https://sourceforge.net/projects/dia-installer/** y se procede a realizar la descarga del instalador del software.

Se ejecuta la instalación del software, para ello se elige el idioma:

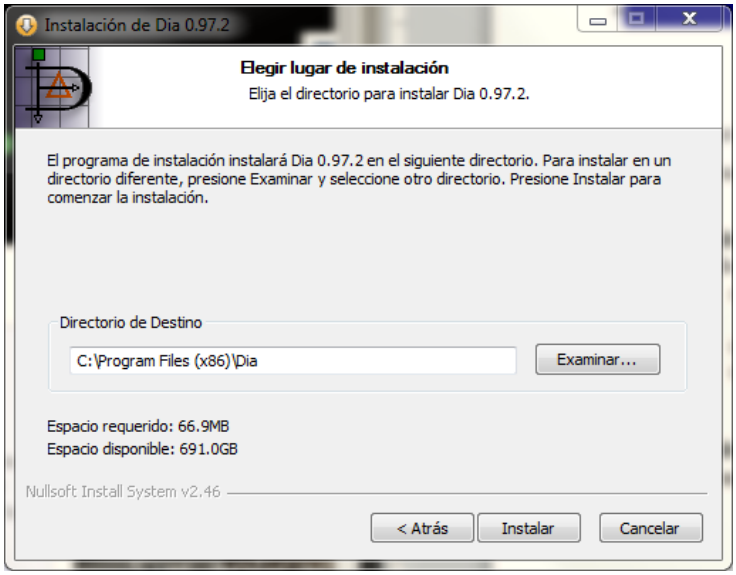
Se continúa con los pasos del asistente de instalación:



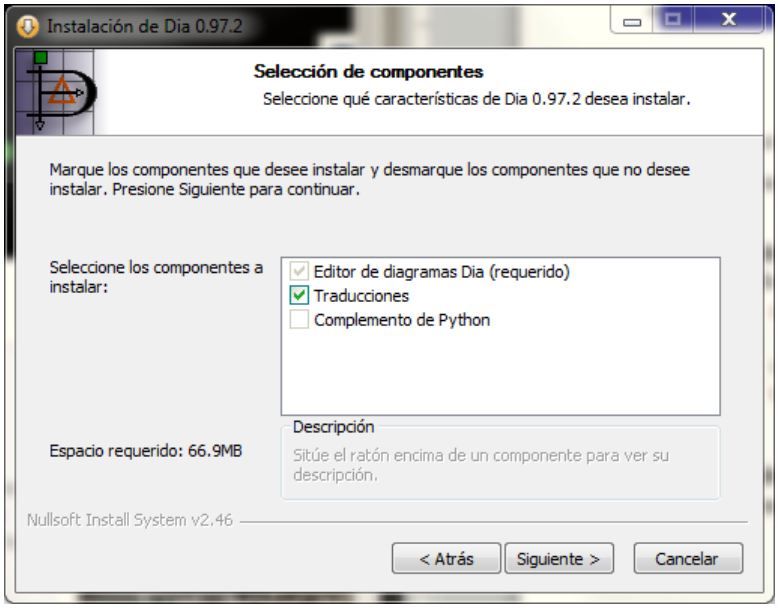
Se pulsa el botón Siguiente >

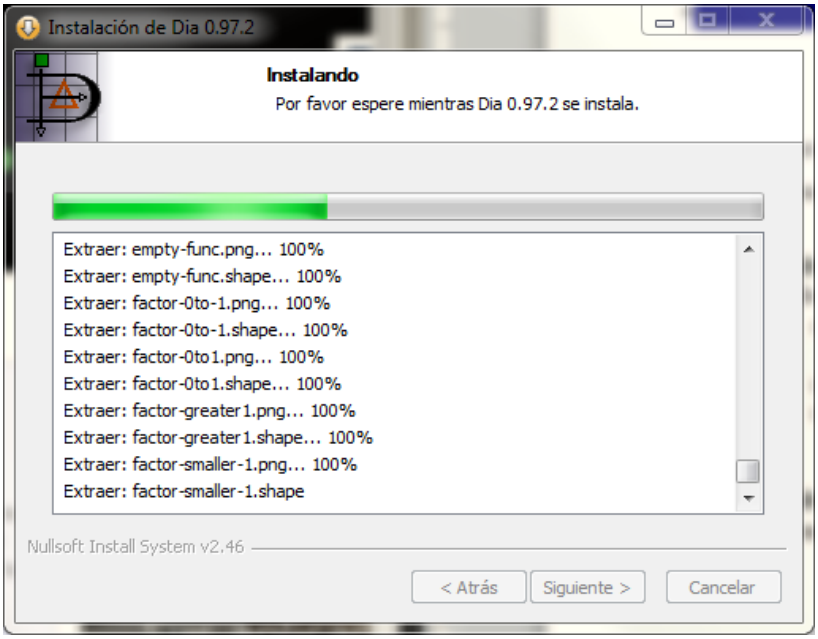


Se indica al instalador la ubicación en la que se desea instalar el software. Por defecto aparecerá la ruta c:\Program Files(x86)\Dia , pudiendo modificar la ubicación del mismo:

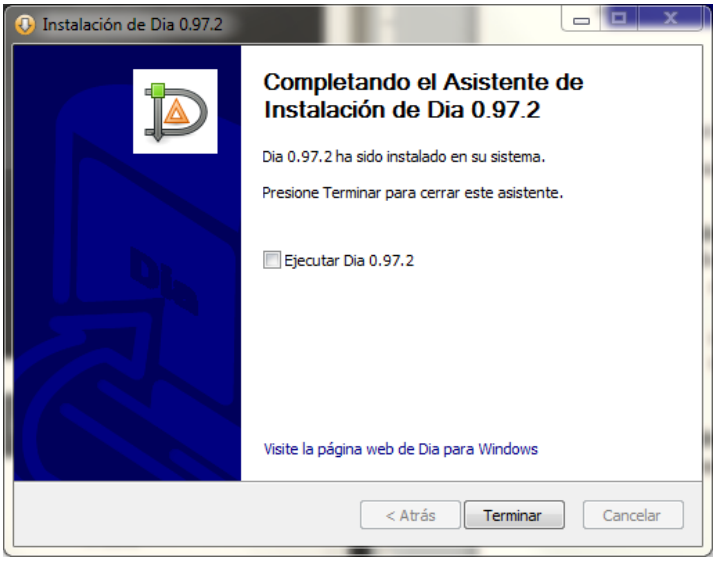


Se continúa con los pasos de la instalación, seleccionando las traducciones y pulsando el botón Siguiente >

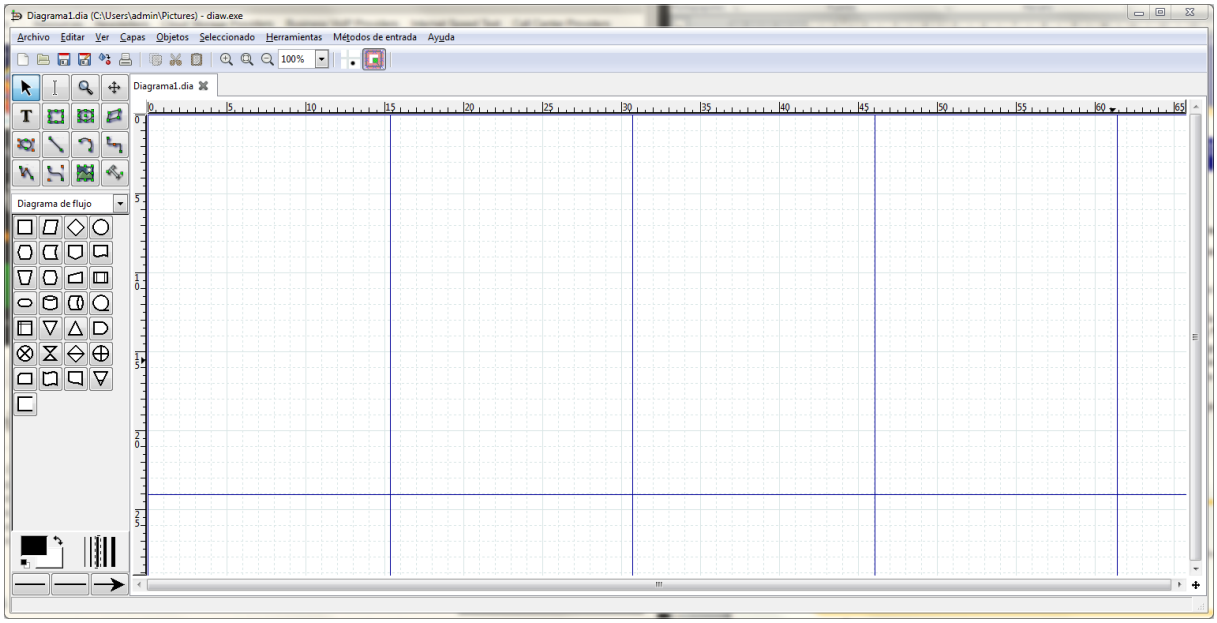




Una vez finalizada la instalación, ya se puede, por fin, utilizar el software de generación de Diagramas.



Al ejecutar el software, aparece la siguiente pantalla principal y se puede comenzar a generar los diagramas.



ENLACE DE INTERÉS

En la siguiente dirección se puede encontrar un buen manual de usuario para el software DIA:

https://issuu.com/jaif/docs/manual_de_editor_de_diagramas_dia

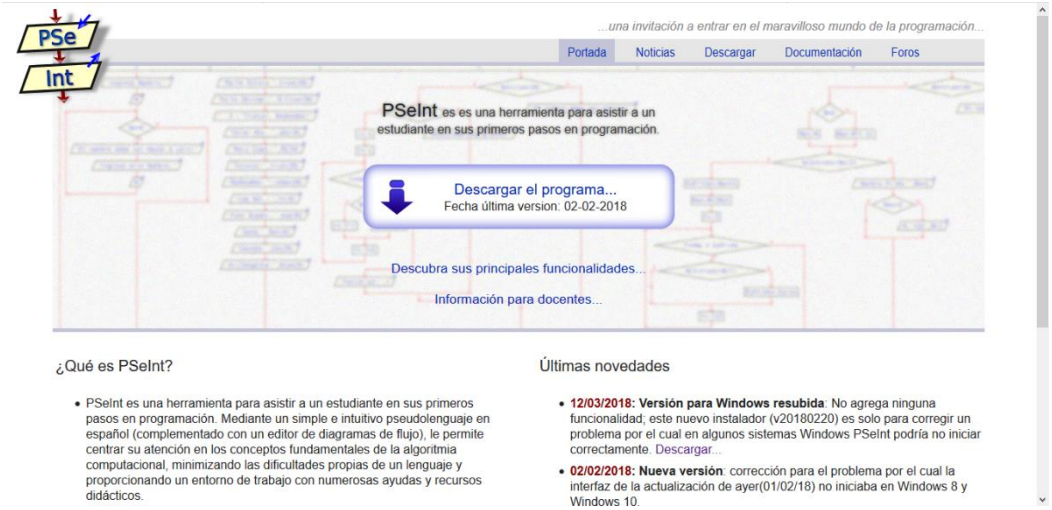
VIDEO DE INTERÉS

El siguiente video explica la utilización de la herramienta de diagramas DIA y la exportación de los diagramas a otros formatos:

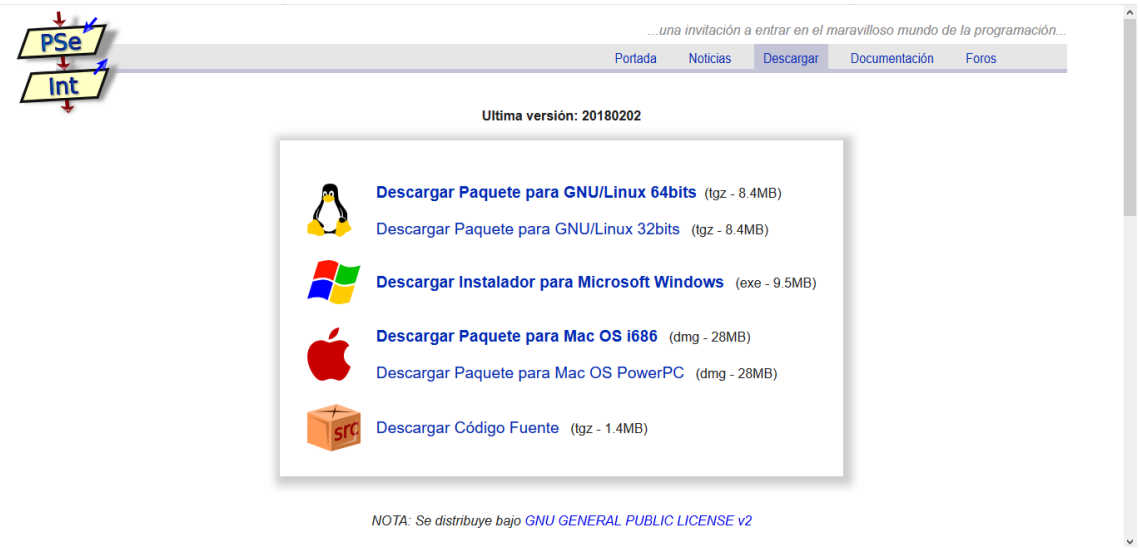
https://www.youtube.com/watch?v=xU3LB_0xodg

Instalación Software PSeInt

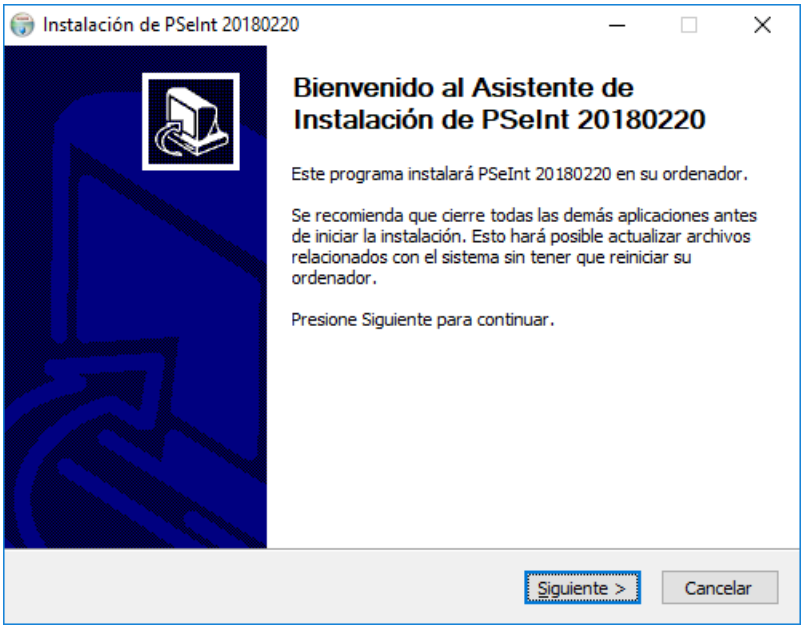
Se accederá a la web oficial del software PSeInt y se realiza la descarga del instalador a través del siguiente enlace: http://pseint.sourceforge.net/



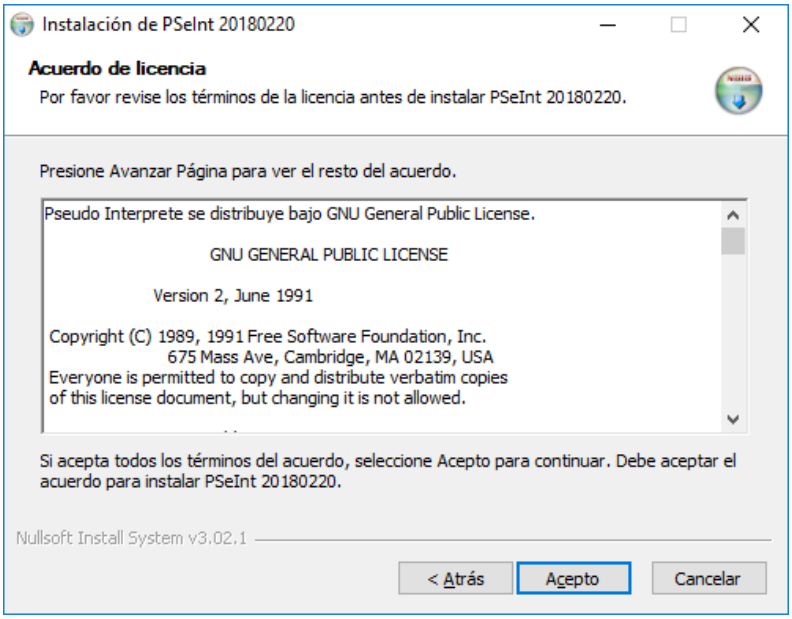
Se selecciona el tipo de instalador necesario para el sistema operativo que se vaya a usar:

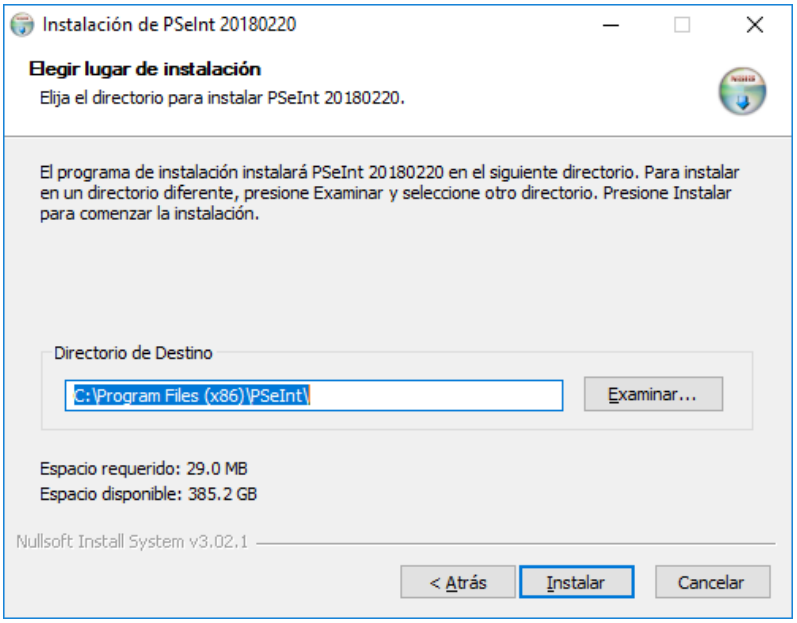


Una vez seleccionado el instalador, se procede a la descarga a través de la web oficial. Tras la descarga se realiza la instalación ejecutando el instalador:



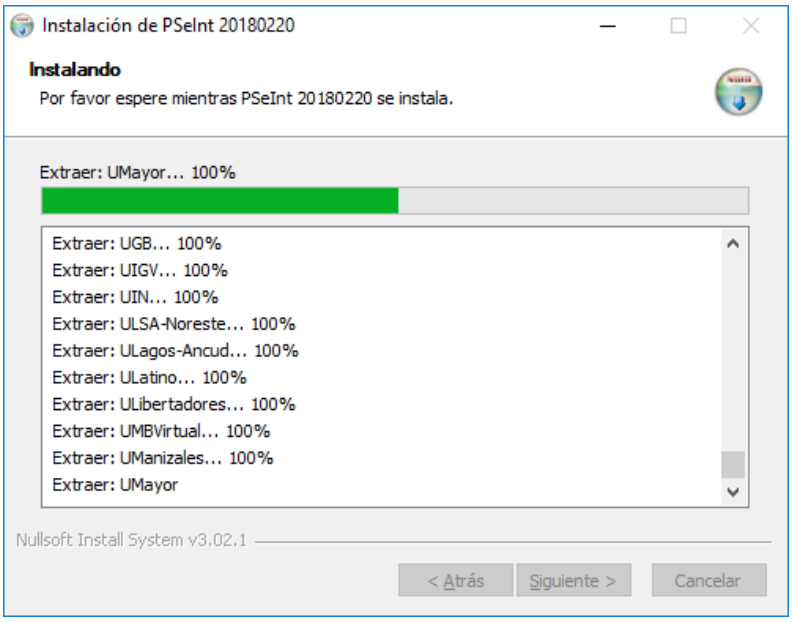
Se siguen los pasos del asistente de instalación, y se pulsa el botón `Siguiente >`

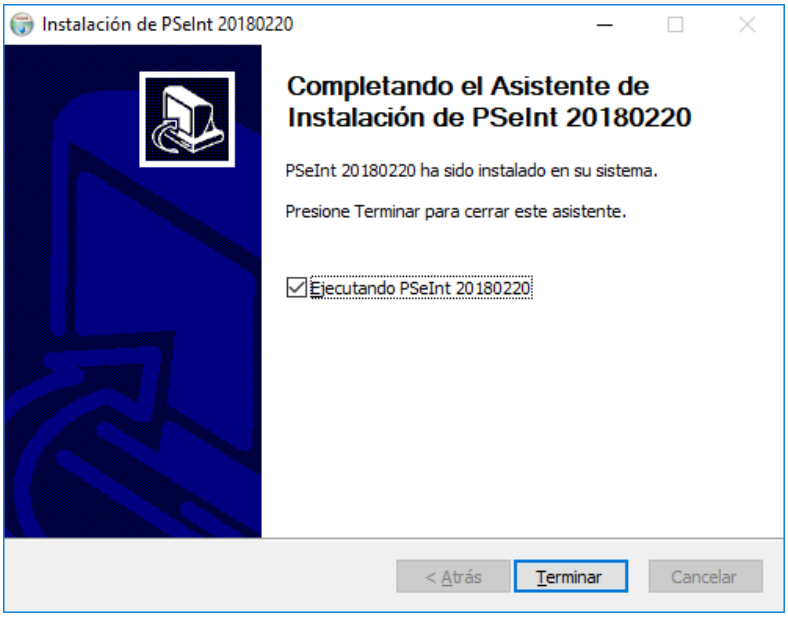




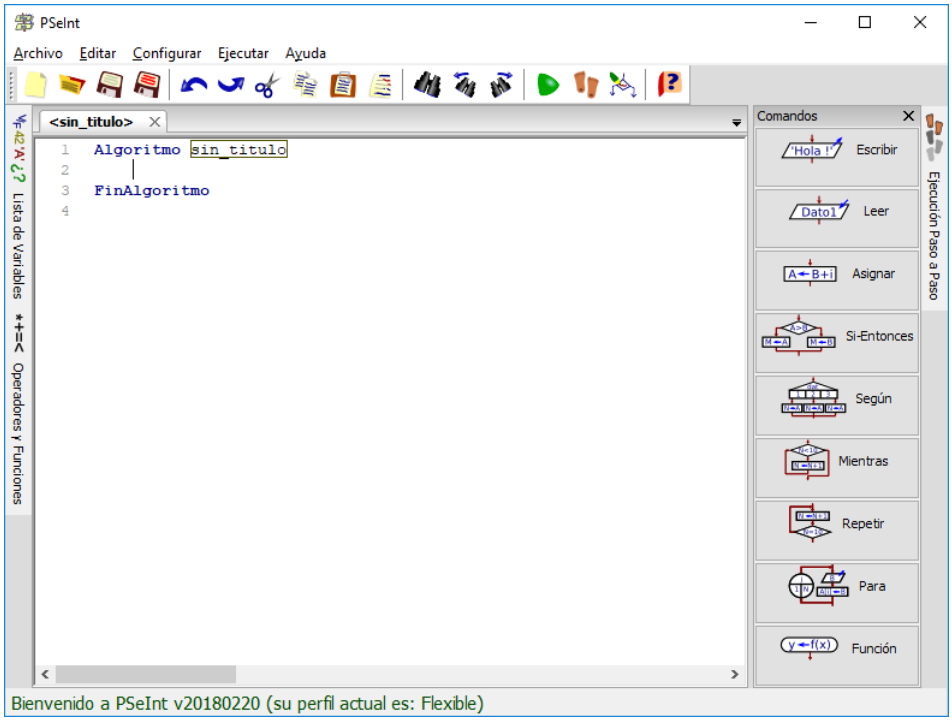
El programa de instalación por defecto instalará el software en la carpeta indicada `c:\Program Files(x86)\PSeInt\`, pudiendo el usuario modificar el destino o ubicación del software PSeInt.

Se continúa con la instalación y finaliza el proceso.





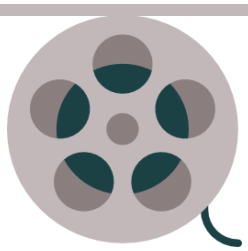
Una vez instalado en el sistema el software PSeInt, mostrará la siguiente pantalla principal



ENLACE DE INTERÉS

En la siguiente dirección se encuentra la documentación oficial del software PSeInt:

http://pseint.sourceforge.net/index.php?page=documentacion.php



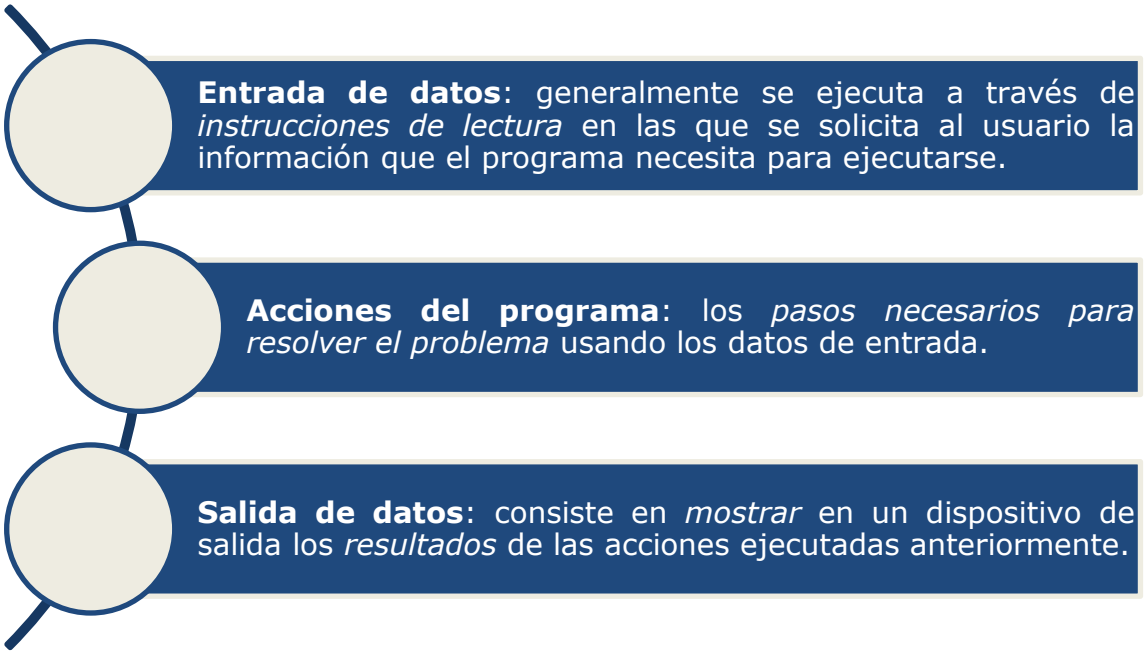
VIDEO DE INTERÉS

El siguiente video explica la utilización de la herramienta PSeInt y diversos ejercicios para inicializarnos en dicho software:

https://www.youtube.com/watch?v=DHIi4dcaMEc

11. CONCEPTO DE PROGRAMA

Un programa es un conjunto de instrucciones que al ejecutarse resuelven un problema. Todo programa tiene que estar formado por las siguientes partes:



11.1 Instrucciones y tipos

El proceso de diseño de un algoritmo y la posterior codificación del programa consiste en definir las acciones o instrucciones que resolverán el problema. Estas instrucciones se deben escribir y posteriormente almacenar en memoria en el mismo orden en el que se van a ejecutar, teniendo en cuenta que un programa puede ser **lineal** o **no lineal**, dependiendo del orden en que se ejecutan sus instrucciones.

Un programa es **lineal** si el orden de ejecución de sus instrucciones es **el mismo orden** que el de lectura, es decir: las acciones se ejecutan secuencialmente. Un programa **no lineal** es el programa en el que se **interrumpe la secuencia de ejecución** mediante instrucciones de bifurcación, es decir: hay saltos que mandan de unas instrucciones a otras. Toda bifurcación en un programa debe realizarse mediante estructuras de control.

Tipos de instrucciones: las instrucciones disponibles en un lenguaje de programación dependen del tipo de lenguaje, pero la clasificación más usual, independientemente del lenguaje es:

Instrucciones de inicio y de fin: indican el *comienzo* y el *final* de un programa.

Instrucciones de asignación: permiten *almacenar un valor* en una variable previamente definida.

Instrucciones de lectura: encargadas de *recoger un dato* de un dispositivo de entrada.

Instrucciones de escritura o de salida: permiten *mostrar información* en dispositivos de salida.

Instrucciones de bifurcación o salto: permiten *interrumpir la ejecución lineal* de un programa.

Estas bifurcaciones pueden ser hacia adelante o hacia atrás, pero siempre deben ser bifurcaciones condicionales.

11.2 Elementos de un programa

Los elementos básicos que forman un programa o un algoritmo son:

1. **Palabras reservadas**: conjunto de palabras especiales que tienen un significado propio dentro del lenguaje y, por lo tanto, sólo pueden ser utilizadas para ello.

2. **Identificadores**: son todos aquellos nombres que aparecen en el programa dados por el programador. Para crear estos identificadores hay unas normas:
   a) Deben estar formados por letras, dígitos y caracteres de subrayado (_).
   b) Tienen que comenzar por una letra.
   c) No pueden contener espacios en blanco.
   d) El nombre asignado debe tener relación con la información que contiene.

3. **Caracteres especiales**: sirven como separadores de instrucciones.

4. **Constantes**: elementos cuyo valor permanece fijo durante toda la

ejecución del programa.

5. **Variables**: elementos que sirven para almacenar datos cuyo valor puede sufrir modificaciones durante la ejecución del programa.

6. **Instrucciones**: los pasos que forman parte de un programa y sirven para indicarle la opción que debe recibir.

7. **Bucles**: cualquier fragmento de código cuyas instrucciones se repiten un número finito de veces. Cada vez que se repite un bucle es una iteración.

8. **Contadores**: una variable cuyo valor se incrementa o decrementa en valores constantes en cada iteración de un bucle.

9. **Acumuladores**: variables destinadas a almacenar cantidades variables provenientes de resultados de operaciones matemáticas. Tanto los contadores como los acumuladores es necesario inicializarlos para que la primera vez que lo incremente o decremente tengan un valor con el que operar.

10. **Interruptores**: son variables que sirven como indicador de una determinada información y sólo pueden tomar uno de los dos valores posibles.

RESUMEN FINAL

En esta unidad se ha analizado el concepto de lenguaje de programación y su evolución.

Además, se ha estudiado la manera de representar un problema, el algoritmo, para posteriormente analizar los distintos modos de descripción de los algoritmos: diagramas de flujo, diagramas N-S y pseudocódigo.

Por último, se ha aprendido qué es un programa y los elementos que lo componen: datos, tipos de datos, variables, constantes, expresiones, operadores y la entrada y la salida de información de un programa.