1.A. Introducción a la programación.

Sitio: Aula Virtual
Curso: Programación_DAM
Libro: 1.A. Introducción a la programación.
Imprimido por: LUIS PUJOL
Día: martes, 5 de noviembre de 2019, 09:29

Tabla de contenidos

1 Introducción.

En esta primera unidad realizaremos un recorrido por los conceptos fundamentales de la programación de aplicaciones. Iniciaremos nuestro camino conociendo con qué vamos a trabajar, qué técnicas podemos emplear y qué es lo que pretendemos conseguir. Continuando con el análisis de las diferentes formas de programación existentes, identificaremos qué fases conforman el desarrollo de un programa, avanzaremos detallando las características relevantes de cada uno de los lenguajes de programación disponibles, para posteriormente, realizar una visión general del lenguaje de programación Java. Finalmente, tendremos la oportunidad de conocer con qué herramientas podríamos desarrollar nuestros programas, escogiendo entre una de ellas para ponernos manos a la obra utilizando el lenguaje Java.

2 Programas y programación.

Resolver problemas de la vida real, es sin duda, uno de los principales objetivos desde que el hombre es hombre y así ha sido a lo largo de los siglos a través de las diferentes sociedades y culturas . Disciplinas como las matemáticas o la física, han ayudado, sin duda, a resolver miles de problemas sencillos y complejos, concretos y abstractos.

Con el desarrollo de la informática y computación, ha cobrado un especial valor, además, resolver los problemas de manera más rápida y precisa. La programación (o mejor dicho, los lenguajes de programación) será el medio que permitirá incorporar en las computadoras los algoritmos para resolver los problemas.

2.1 Buscando una solución.

Generalmente, la primera razón que mueve a una persona hacia el aprendizaje de la programación es utilizar el ordenador como herramienta para resolver problemas concretos. Como en la vida real, la búsqueda y obtención de una solución a un problema determinado, utilizando medios informáticos, se lleva a cabo siguiendo unos pasos fundamentales. En la siguiente tabla podemos ver estas analogías.

Resolución de problemas	
En la vida real...	**En Programación...**
Observación de la situación o problema.	**Análisis del problema:** requiere que el problema sea definido y comprendido claramente para que pueda ser analizado con todo detalle.
Pensamos en una o varias posibles soluciones.	**Diseño o desarrollo de algoritmos:** procedimiento paso a paso para solucionar el problema dado.
Aplicamos la solución que estimamos más adecuada.	**Resolución del algoritmo elegido en el ordenador:** consiste en convertir el algoritmo en programa, ejecutarlo y comprobar que soluciona verdaderamente el problema.

¿Qué virtudes debería tener nuestra solución?

- **Corrección y eficacia:** si resuelve el problema adecuadamente.
- **Eficiencia:** si lo hace en un tiempo mínimo y con un uso óptimo de los recursos del sistema.

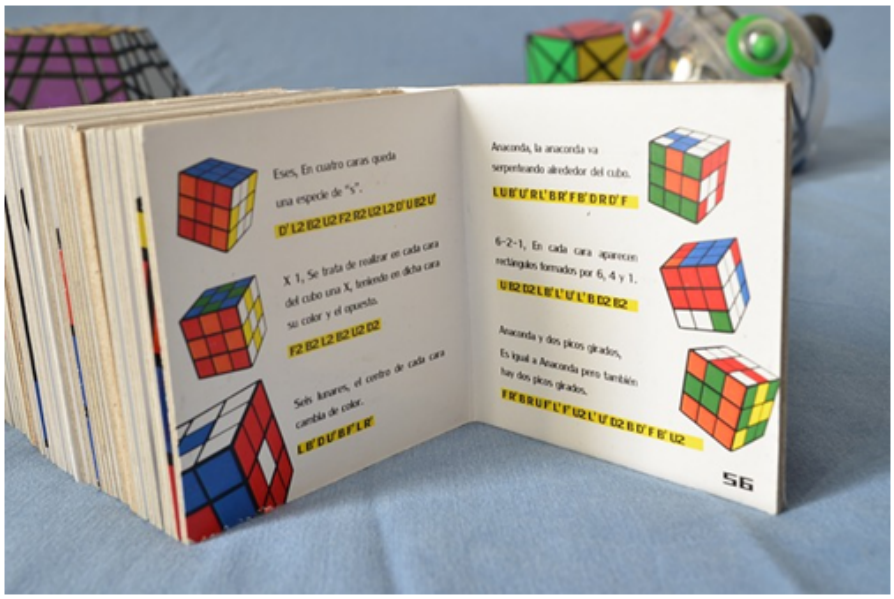
Para conseguirlo, cuando afrontemos la construcción de la solución tendremos que tener en cuenta los siguientes conceptos:

1. **Abstracción:** se trata de realizar un análisis del problema para descomponerlo en problemas más pequeños y de menor complejidad, describiendo cada uno de ellos de manera precisa. **Divide y vencerás,** esta suele ser considerada una filosofía general para resolver problemas y de aquí que su nombre no sólo forme parte del vocabulario informático, sino que también se utiliza en muchos otros ámbitos.
2. **Encapsulación:** consiste en ocultar la información para poder implementarla de diferentes maneras sin que esto influya en el resto de elementos.
3. **Modularidad:** estructuraremos cada parte en módulos independientes, cada uno de ellos tendrá su función correspondiente.

2.2 Algoritmos y programas.

Después de analizar en detalle el problema a solucionar, hemos de diseñar y desarrollar el algoritmo adecuado. Pero, **¿qué es un algoritmo?**

Algoritmo: secuencia ordenada de pasos, descrita sin ambigüedades, que conducen a la solución de un problema dado.



Los algoritmos son independientes de los lenguajes de programación y de las computadoras donde se ejecutan. Un mismo algoritmo puede ser expresado en diferentes lenguajes de programación y podría ser ejecutado en diferentes dispositivos. Piensa en una receta de cocina, ésta puede ser expresada en castellano, inglés o francés, podría ser cocinada en fogón o vitrocerámica, por un cocinero o más, etc. Pero independientemente de todas estas circunstancias, el plato se preparará siguiendo los mismos pasos.

La diferencia fundamental entre algoritmo y **programa** es que, en el segundo, los pasos que permiten resolver el problema, deben escribirse en un determinado **lenguaje de programación** para que puedan ser ejecutados en el ordenador y así obtener la solución.

Los lenguajes de programación son sólo un medio para expresar el algoritmo y el ordenador un procesador para ejecutarlo. El diseño de los algoritmos será una tarea que necesitará de la creatividad y conocimientos de las técnicas de programación. Estilos distintos, de distintos programadores a la hora de obtener la solución del problema, darán lugar a algoritmos diferentes, igualmente válidos.

En esencia, todo problema se puede describir por medio de un algoritmo y las características fundamentales que éstos deben cumplir son:

- Debe ser **preciso** e indicar el orden de realización paso a paso.
- Debe estar **definido**, si se ejecuta dos o más veces, debe obtener el mismo resultado cada vez.
- Debe ser **finito**, debe tener un número finito de pasos.

Pero cuando los problemas son complejos, es necesario descomponer éstos en subproblemas más simples y, a su vez, en otros más pequeños. Estas estrategias reciben el nombre de **diseño descendente** o **diseño modular** (**top-down design**). Este sistema se basa en el lema **divide y vencerás**.

Para representar gráficamente los algoritmos que vamos a diseñar, tenemos a nuestra disposición diferentes herramientas que ayudarán a describir su comportamiento de una forma precisa y genérica, para luego poder codificarlos con el lenguaje que nos interese. Entre otras tenemos:

· **Diagramas de flujo:** Esta técnica utiliza símbolos gráficos para la representación del algoritmo. Suele utilizarse en las fases de análisis.

· **Pseudocódigo:** Esta técnica se basa en el uso de palabras clave en lenguaje natural, constantes, variables, otros objetos, instrucciones y estructuras de programación que expresan de forma escrita la solución del problema. Es la técnica más utilizada actualmente.

· **Tablas de decisión:** En una tabla son representadas las posibles condiciones del problema con sus respectivas acciones. Suele ser una técnica de apoyo al pseudocódigo cuando existen situaciones condicionales complejas.

Debes conocer

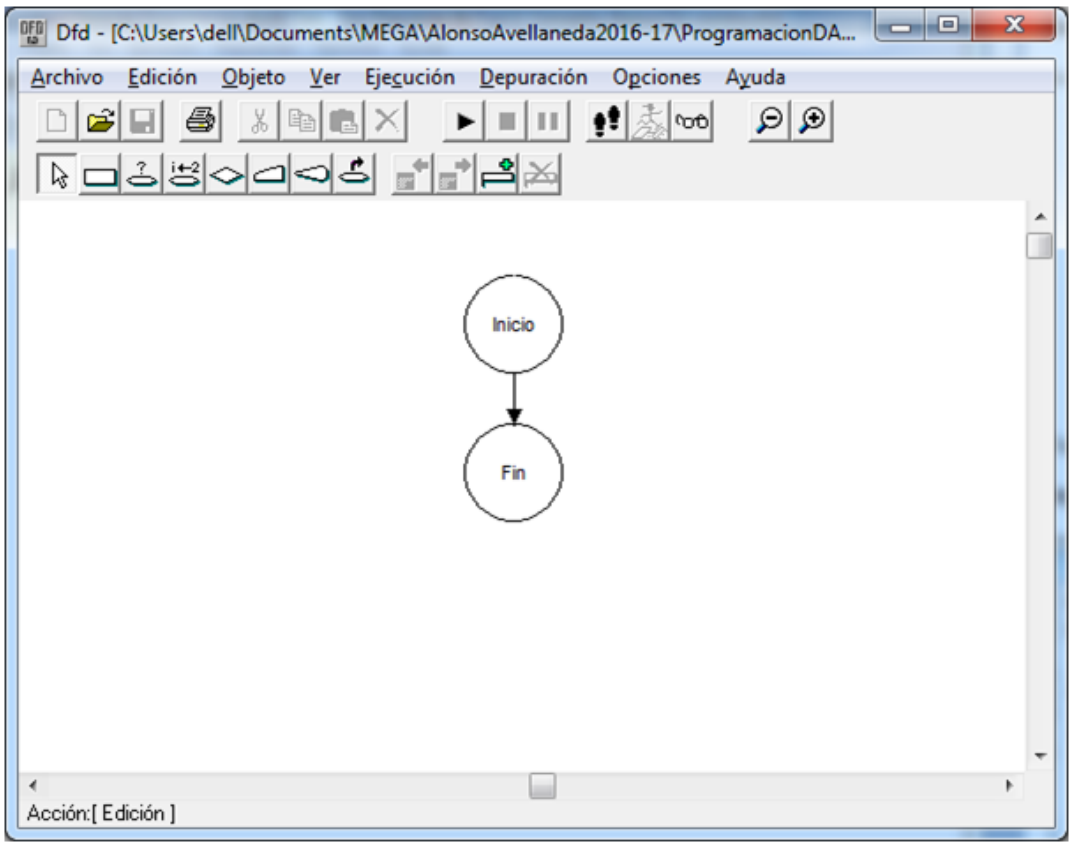
A continuación te ofrecemos dos enlaces muy interesantes:

- En el primer enlace puedes ver los elementos gráficos fundamentales que se utilizan para la generación de diagramas de flujo.

https://www.heflo.com/es/blog/modelado-de-procesos/significado-simbolos-diagrama-flujo/

· El segundo recurso, consiste en una aplicación que permite construir diagramas de flujo para la resolución de tareas mediante algoritmos. El programa se llama DFD, y se puede obtener de https://www.descargarsoft.com/descargar-dfd-para-crear-diagramas-de-flujo/

Se trata de una aplicación portable. Tras su descarga, simplemente lanzamos el ejecutable y veremos la siguiente ventana. Se tratará de incorporar comandos de la barra de herramientas superior al panel de trabajo:



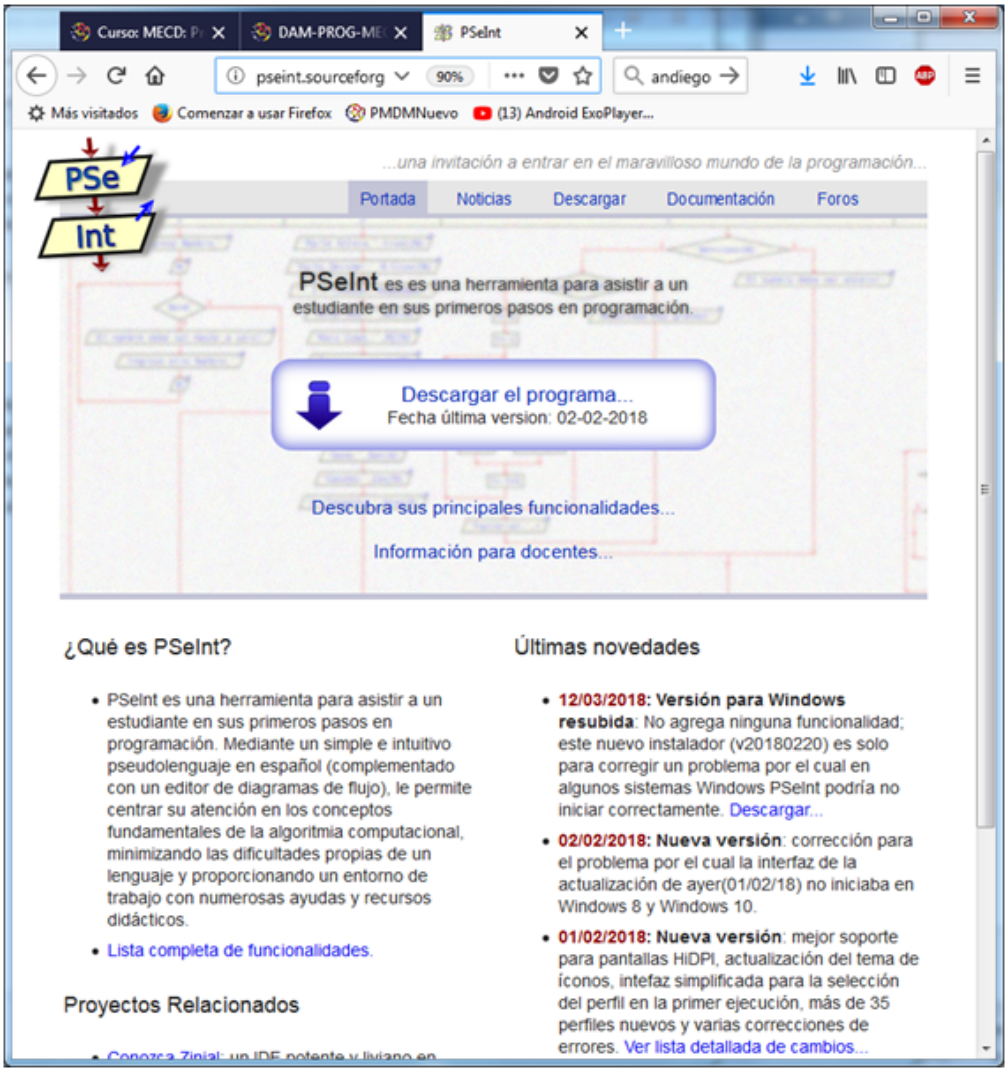
Podemos ver un video sobre cómo crear un algoritmo con esta herramienta:

5. Diagrama de Flujo en DFD || Conceptos Básicos || Analizando DFD

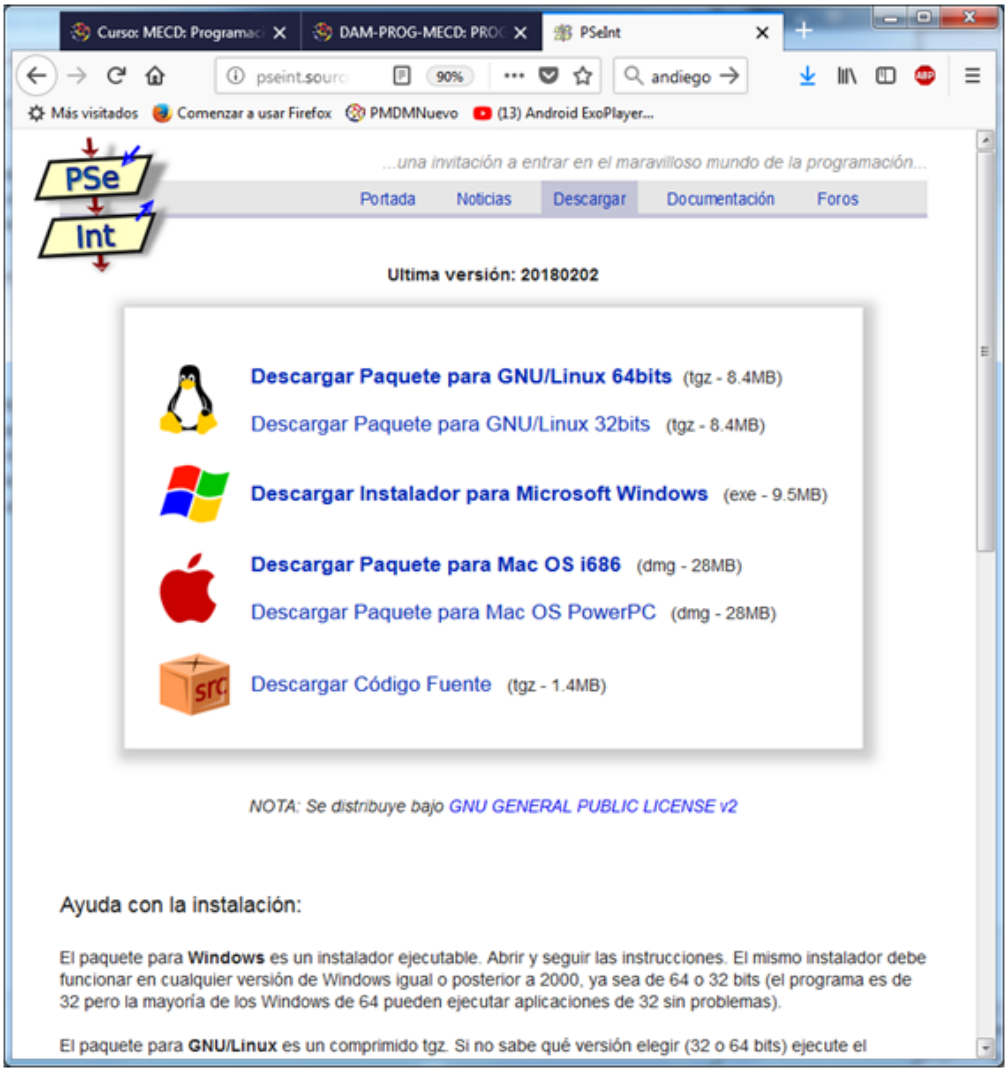
- El tercer recurso permite la construcción de un diagrama de flujo con otra herramienta gráfica y su transformación a pseudocódigo. Se trata del programa PSeInt:

http://pseint.sourceforge.net/

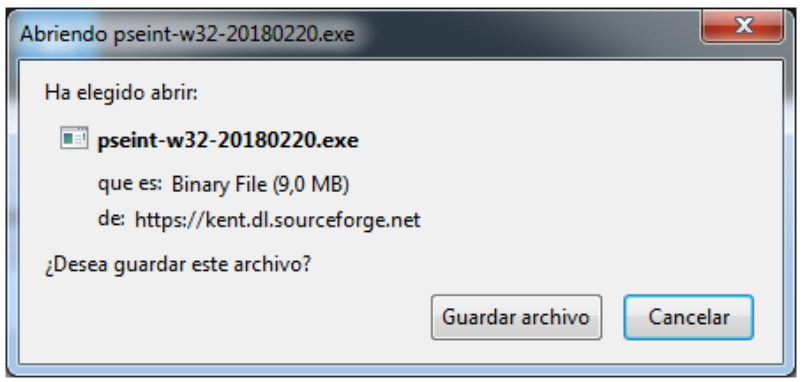
Veamos aquí su instalación y puesta en funcionamiento:



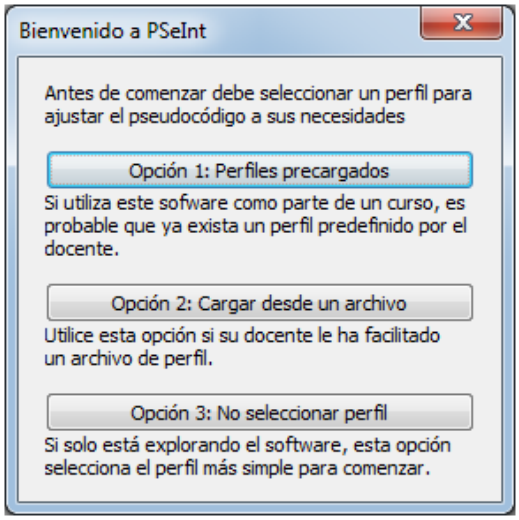
Vamos a "Descargar el programa...":



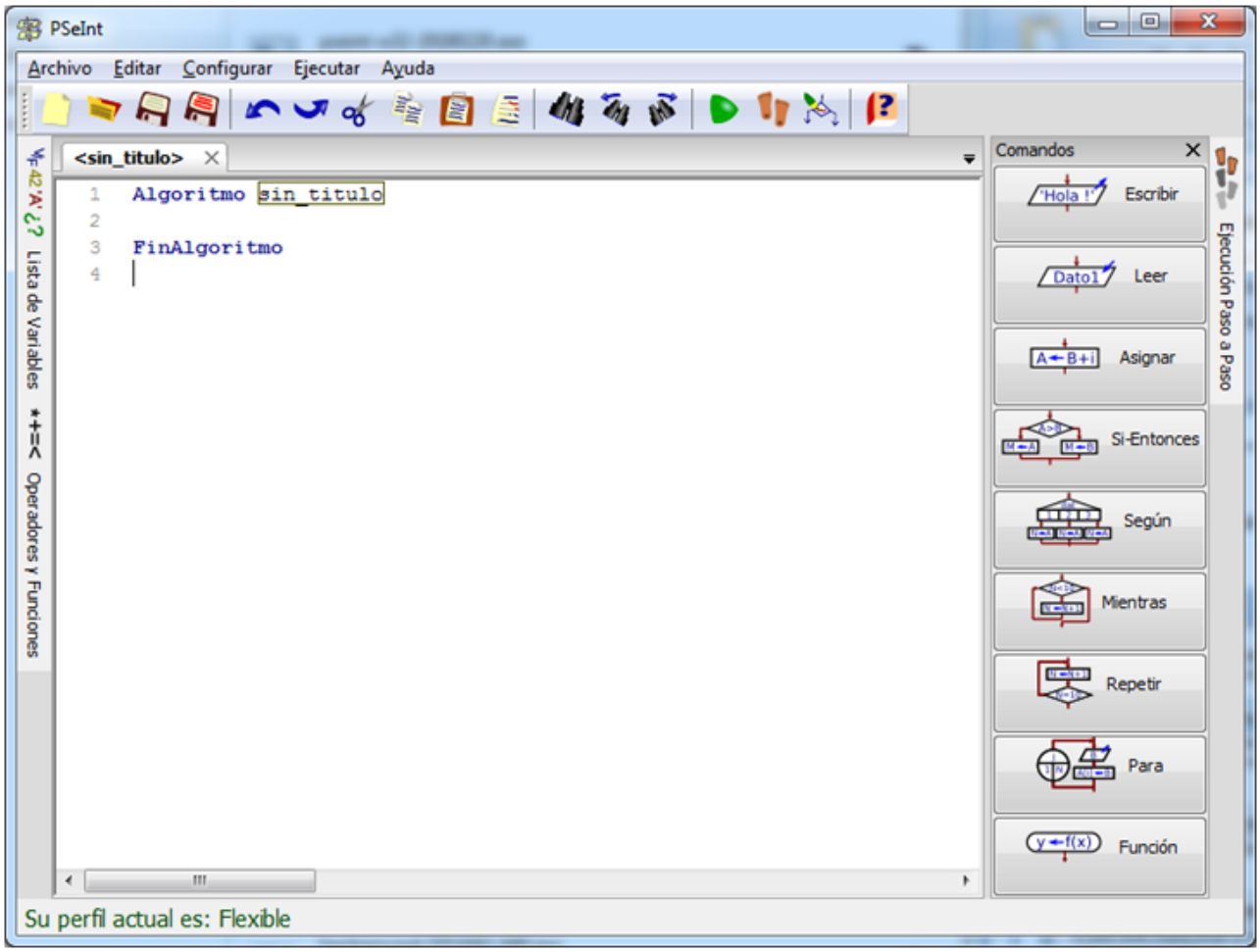
Si por ejemplo instalamos la versión para Windows:



Procedemos a su instalación, de manera básica. Si lo ejecutamos por primera vez aparecerá la ventana:



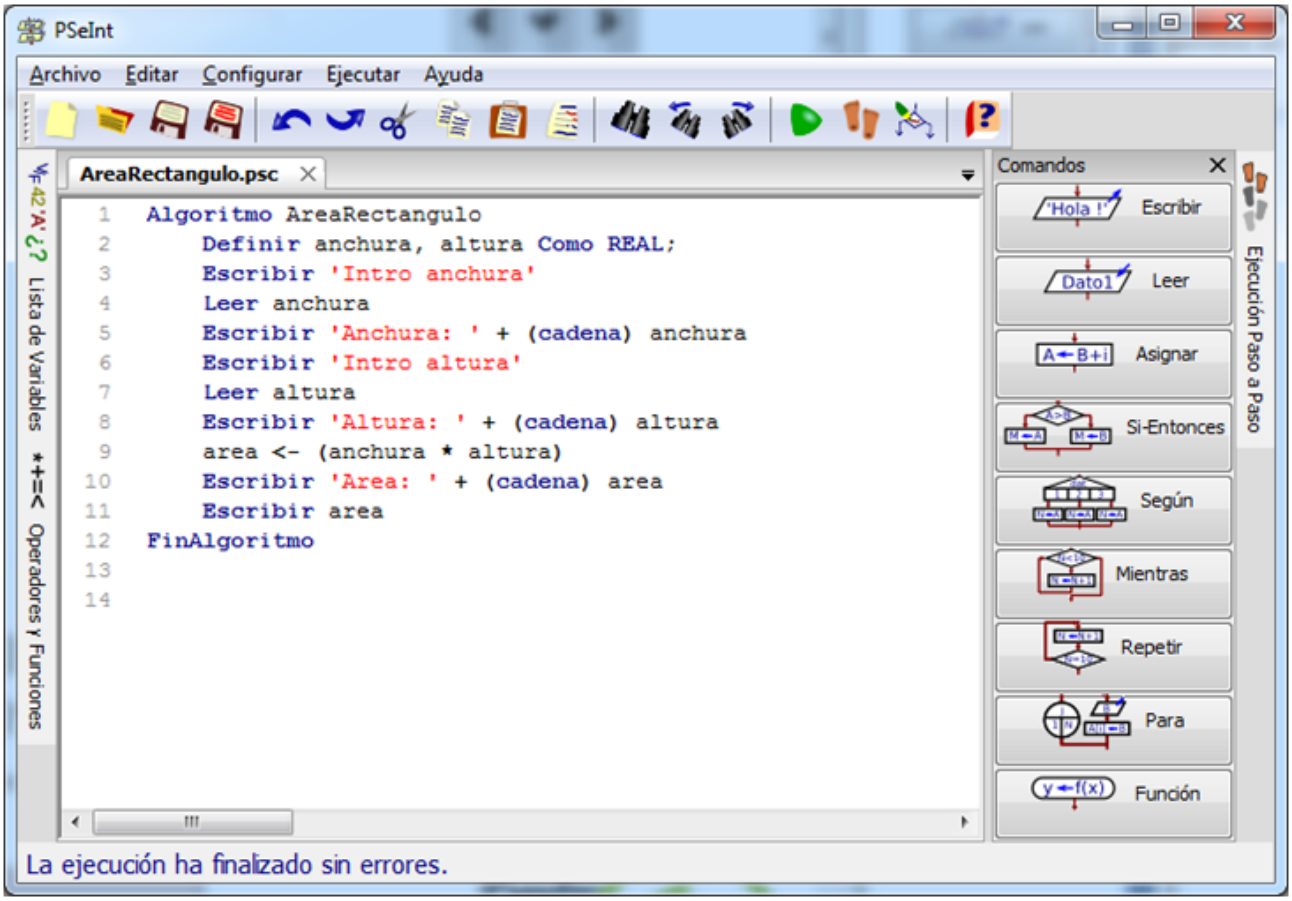
Simplemente seleccionamos la "Opción 3: No seleccionar perfil", tras lo cual, ya se cargará el entorno:



Como vemos, dispondremos de 2 paneles principales, el de la izquierda para incorporar pseudocódigo directamente, y el de la derecha, con los comandos disponibles para ayudar, si se desconoce la sintaxis en la escritura de instrucciones o estructuras de control.

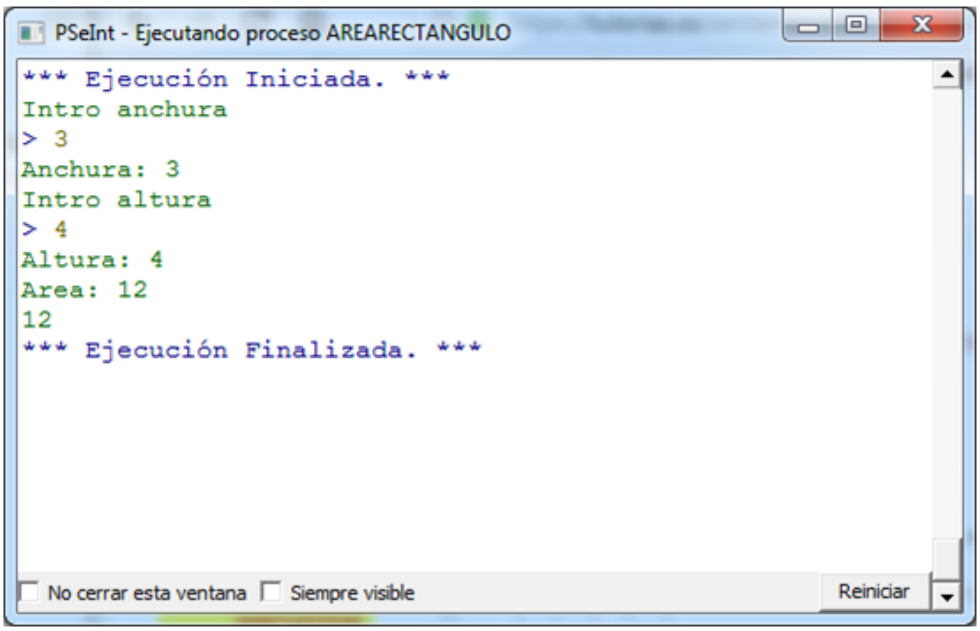
Para mayor flexibilidad en el lenguaje: Menú Configurar --> Opciones del Lenguaje (perfiles)... --> Elegir "Flexible"

Podemos ver cómo crear un sencillo algoritmo del cálculo del área de un cuadrado:



En dicho algoritmo, quizá lo más significativo sea la adaptación o "casteo" de tipos, para poder concatenar los números considerados como cadenas con cadenas de texto.

La salida resultante de este algoritmo la obtenemos al dar al icono de reproducir verde de la ventana anterior apareciendo la siguiente ventana:



Autoevaluación

Rellena los huecos con los conceptos adecuados:

A los pasos que permiten resolver el problema, escritos en un lenguaje de programación, para que puedan ser ejecutados en el ordenador y así obtener la solución, se les denomina: _____ .

Resolver

3 Paradigmas de la programación.

¿Cuántas formas existen de hacer las cosas? Supongo que estarás pensando: varias o incluso, muchas. Pero cuando se establece un patrón para la creación de aplicaciones nos estamos acercando al significado de la palabra paradigma.

Paradigma de programación: es un modelo básico para el diseño y la implementación de programas. Este modelo determinará como será el proceso de diseño y la estructura final del programa.

El paradigma representa un enfoque particular o filosofía para la construcción de software. Cada uno tendrá sus ventajas e inconvenientes, será más o menos apropiado, pero no es correcto decir que exista uno mejor que los demás.

Como habrás podido apreciar, existen múltiples paradigmas, incluso puede haber lenguajes de programación que no se clasifiquen únicamente dentro de uno de ellos. Un lenguaje como **Smalltalk** es un lenguaje basado en el paradigma orientado a objetos. El lenguaje de programación **Scheme**, en cambio, soporta sólo programación funcional. **Phyton**, soporta múltiples paradigmas.

Para saber más

Te proponemos el siguiente enlace en el que encontrarás información adicional sobre los diferentes paradigmas de programación.

Paradigmas de programación y lenguajes

¿Cuál es el objetivo que se busca con la aplicación de los diferentes enfoques? Fundamentalmente, reducir la dificultad para el mantenimiento de las aplicaciones, mejorar el rendimiento del programador y, en general, mejorar la productividad y calidad de los programas.

Autoevaluación

¿En qué paradigma de programación podríamos enmarcar el lenguaje de programación Java?

- Programación estructurada
- Programación declarativa
- Programación orientada a objetos