Introducción a los sistemas

Objetivos

- Conocer a los docentes y presentarse ante el resto de los alumnos
- Conocer los objetivos, componentes y características del curso
- Conocer cómo acceder y utilizar el Aula Virtual
- Conocer las pautas generales de cumplimiento obligatorio
- Conocer el programa del curso
- Conocer el cronograma del curso
- Conocer la carga horaria del curso
- Conocer la metodología de evaluación y cómo obtener la certificación
- Reconocer el vocabulario básico usado en sistemas

Contenidos

Introducción a los sistemas. Sistemas informáticos. Componentes de los sistemas informáticos: Hardware y Software, definiciones. Hardware: el computador. Componentes y funciones. Software. Sistema operativo.

- Desarrollo de software. Ciclos de vida del software. Metodologías de desarrollo de sistemas. Lenguaje de programación. Tipos de Lenguajes de programación. Lenguajes interpretados y compilados.
- Historia de la Programación.
- Instalación y nociones básicas de **PSeInt**. Primer Programa.

¿Qué es un sistema?

Iniciamos el recorrido introduciendo los conceptos básicos relacionados con sistemas, sistemas informáticos y software. Esta unidad nos servirá para unificar las definiciones que debemos conocer antes de comenzar a programar.

Seguramente el término **sistema** lo habrán conocido cuando cursaban su escuela secundaria, en la asignatura Biología.

También, en la vida cotidiana utilizamos este concepto para hablar del sistema ferroviario, sistema de salud, sistema inmunológico, por nombrar algunos.

¿Cómo definimos un sistema? *Hacé clic en el botón para ver la respuesta.*

Un sistema es es un conjunto de elementos relacionados entre sí que tienen un cierto orden u organización y que cumplen una función.

Te contamos que el concepto de sistema surge del denominado enfoque sistémico, basado en la Teoría General de los Sistemas formulada a mediados del siglo XX.

Inicialmente, esta teoría estaba basada en el estudio de los organismos como sistemas biológicos, pero luego se generalizó su alcance a todo tipo de sistemas. De tal manera que hoy se utiliza el término sistema en todas las áreas del conocimiento humano.

En este curso nos interesan los **sistemas de información**, que son aquellos *sistemas que procesan datos con la finalidad de generar, transformar y distribuir la información.*

Los sistemas de información están formados por **hardware** (elementos físicos) y **software** (elementos lógicos que se llaman programas). En este curso nos ocuparemos del desarrollo del software, estudiando distintas técnicas de programación para la construcción de los programas.

¿Qué es un software?

El software en un sistema corresponde a todos los elementos lógicos, intangibles.

Ahora bien, las computadoras, los cables, los celulares, los dispositivos en donde guardamos información, ¿son Software?

Seguramente estarás pensando que no, todos estos elementos son tangibles.

Entonces:

¿Qué es un software?

El software de un sistema informático es el **conjunto de programas** necesarios para que una computadora funcione. Un **programa** es un conjunto de instrucciones que indican a una computadora las tareas que tiene que realizar.

Para la construcción de **software** se debe seguir un determinado procedimiento que garantice que el resultado sea un producto de calidad. Hay varios modelos o procedimientos a seguir, lo que se traducirá en distintas metodologías de desarrollo, algunas de las cuales veremos más adelante.

Sistema Operativo

Un **sistema operativo SO** (Operating System, OS) es tal vez la parte más importante del software del sistema y es el software que controla y gestiona los recursos de la computadora. En la práctica el sistema operativo es la colección de programas de computadora que controla la interacción del usuario y el hardware de la computadora. El sistema operativo es el administrador principal de la computadora, y por ello a veces se la compara con el director de una orquesta ya que este software es el responsable de dirigir todas las operaciones de la computadora y gestionar todos sus recursos. El sistema operativo asigna recursos, planifica el uso de recursos y tareas de la computadora, y monitoriza las actividades del sistema informático. Estos recursos incluyen memoria, dispositivos de E/S (Entrada/Salida), y la UCP (Unidad Central de Proceso). El sistema operativo proporciona servicios tales como asignar memoria a un programa y manipulación del control de los dispositivos de E/S tales como el monitor, el teclado o las unidades de disco.

Cuando un usuario interactúa con una computadora, la interacción está controlada por el sistema operativo. Un usuario se comunica con un sistema operativo a través de una interfaz de usuario de ese sistema operativo. Los sistemas operativos modernos utilizan una **interfaz gráfica de usuario, IGU** (Graphical User Interface, **GUI**) que hace uso masivo de iconos, botones, barras y cuadros de diálogo para realizar tareas que se controlan por el teclado o el ratón (mouse), entre otros dispositivos. Normalmente el sistema

operativo se almacena de modo permanente en un chip de memoria de sólo lectura (ROM), de modo que esté disponible tan pronto la computadora se pone en marcha (“se enciende” o “se prende”). Otra parte del sistema operativo puede residir en disco, que se almacena en memoria RAM en la inicialización del sistema por primera vez en una operación que se llama carga del sistema (booting). El sistema operativo dirige las operaciones globales de la computadora, instruye a la computadora para ejecutar otros programas y controla el almacenamiento y recuperación de archivos (programas y datos) de cintas y discos. Gracias al sistema operativo es posible que el programador pueda introducir y grabar nuevos programas, así como instruir a la computadora para que los ejecute.

Ciclo de vida de software

Todo programa está formado por un conjunto de **módulos o subprogramas** que interactúan entre sí. En muchos casos, durante todo el proceso de desarrollo, intervienen distintas personas con diferentes responsabilidades. Es por eso que los programas atraviesan diferentes etapas durante su proceso de construcción, y es lo que se conoce como **"ciclo de vida**".

El desarrollo de un sistema se realiza durante todo el ciclo de vida, que es el período de tiempo que se extiende desde la idea original del problema a resolver hasta el mantenimiento y desarrollo de las mejoras.

Independientemente de las metodologías de desarrollo que se utilicen, podemos identificar las siguientes **etapas o momentos en el desarrollo del software:**

Análisis del problema

En esta etapa se debe determinar cuál es el problema a resolver y los límites y alcances que tendrá el software que lo resolverá. Es el momento de reunirse con quien nos solicita el programa para saber cuáles son los requerimientos.

Especificación del software

En este momento los profesionales de sistemas se encargan de definir las entradas y las salidas del software, y qué restricciones tendrán los datos. También se describen los componentes que se deberán desarrollar, qué características y comportamiento tendrán y cómo estarán relacionados.

Desarrollo del software

Corresponde al proceso de construcción de software propiamente dicho. Es en donde los programadores escriben el código fuente utilizando algún lenguaje de programación.

Verificación del software

Una vez que el software está desarrollado se debe probar para verificar que responde a las definiciones y no tiene errores.

Mantenimiento del software

Todo sistema debería tener mantenimiento ya que siempre habrá que realizar alguna modificación, agregando nueva funcionalidad o bien cambiando alguna característica porque se ha modificado alguna especificación.

El siguiente esquema nos presenta una visualización del ciclo de vida del software:

Lenguajes de programación

Para que una computadora pueda realizar un proceso es necesario que se le brinde una lista de instrucciones que sea capaz de comprender y de ejecutar. Como vimos anteriormente, *un programa o software es ese conjunto de instrucciones*. Esos programas se escriben utilizando un lenguaje de programación.

Entonces:

¿Qué es un lenguaje de programación?

Un lenguaje de programación es un lenguaje utilizado para escribir programas que sean entendidos y procesados por una computadora.

Ahora te presentaremos los principales **tipos de lenguajes de programación** utilizados en la actualidad:

Lenguaje de máquina

Este lenguaje es el que entiende directamente una computadora. Está formado por instrucciones binarias (dígitos 0 y 1) que especifican operaciones y las direcciones de memoria implicadas en dichas operaciones.

El código máquina es el denominado código binario. Estas instrucciones dependen del hardware de la computadora, y por lo tanto, varían de una en otra.

Este lenguaje al ser entendible por computadoras es muy poco legible y claro para los programadores, es por eso que se han desarrollado otros lenguajes de programación más cercanos a los idiomas para que sea mucho más fácil programar. Estos lenguajes se dividen en lenguajes de bajo y de alto nivel.

Lenguaje de bajo nivel

Es más simple de utilizar que el lenguaje máquina, pero también depende del hardware en donde se esté ejecutando.

El principal lenguaje de bajo nivel es el ensamblador.

Lenguajes de alto nivel

Estos lenguajes son los más utilizados por los programadores, ya que están diseñados para que las personas lo pueden entender de manera mucho más fácil que los lenguajes de bajo nivel.

Otra ventaja que tiene es que un programa escrito en este tipo de lenguaje es independiente del hardware en donde se va a ejecutar.

Los principales lenguajes de alto nivel son C, C++, Pascal, Java, VisualBasic.

Los programas fuente escritos en estos lenguajes tiene que ser traducidos al lenguaje máquina. Para eso se utiliza un **programa traductor** que es un intérprete o un compilador.

En la siguiente sección te presentaremos los programas traductores utilizados por los lenguajes de alto nivel.

Lenguajes interpretados y compilados

Como hemos visto, un programa escrito en un lenguaje de alto nivel es entendible solo por una persona, es decir, el programador. Por lo tanto, para que pueda ser entendible y ejecutado por una computadora debe ser traducido al lenguaje máquina, por medio de un intérprete o un compilador.

De esta manera tenemos **lenguajes interpretados o compilados**.

¿Qué es un lenguaje interpretado?

Los lenguajes interpretados son aquellos que se ejecutan por medio de un intérprete.
Un intérprete es un traductor que toma el código fuente, lo traduce y luego lo ejecuta.

Un lenguaje interpretado clásico es el "Basic" que hoy, prácticamente no se utiliza.

Por otro lado, los **lenguajes compilados** son aquellos que necesitan un compilador para poder ejecutarse.

¿Qué es un compilador?

Un compilador es un programa especial que toma el código fuente de alto nivel y lo convierte en instrucciones de código entendible por una computadora, denominado código objeto.

En general, el **programa objeto** es la traducción al código máquina, y que depende de cada modelo de computadora. Por lo tanto, el compilador debe ser específico para el modelo de computadora que va a ejecutar el programa.

El proceso de compilación abarca las siguientes etapas:

1. Escritura del programa fuente.
2. Compilación del programa fuente.
3. Verificación de errores de compilación.
4. Obtención del programa objeto.
5. Enlace del programa objeto con todos los programas necesarios del sistema operativo.
6. Obtención del programa ejecutable.

Historia de la Programación

En el año 1801, un comerciante textil francés llamado **Joseph Marie Jacquard** inventó un telar gobernado por un sistema de tarjetas perforadas, que presentó en una exhibición industrial de Lyon en 1805. El telar en sí no fue revolucionario, pero sí el sistema de tarjetas perforadas, que permitían el movimiento independiente de los hilos a través de unos ligamentos insertados en diferentes zonas del tejido. Cada tarjeta perforada correspondía a una línea del diseño y la suma de todas las tarjetas creaba el patrón.

Basado en las ideas de Jacquard, entre 1833 y 1842, **Charles Babbage**, matemático y científico británico, impulsa la creación de una máquina capaz de realizar cálculos aritméticos, denominada **"máquina analítica"**. La misma tenía dispositivos de entrada basados en las tarjetas perforadas del telar de Jacquard, un procesador aritmético, que calculaba números, una unidad de control que determinaba qué tarea debía ser realizada, un mecanismo de salida y una memoria donde los números podían almacenarse hasta ser procesados. Se la considera como **la primera computadora de la historia.**

Mientras Babbage intentó conseguir financiación para su proyecto, **Lady Ada Lovelace,** matemática hija de Lord Byron, se interesó plenamente en él. Ambos matemáticos concebían a la máquina de maneras diferentes: a Babbage no le interesaban mucho sus usos prácticos, pero, por el contrario, Ada se obsesionó con las aplicaciones del invento. Fue la primera en intuir que la máquina significaba un progreso tecnológico. Entendió que podía ser aplicada a todo proceso que implicara tratar datos, abriendo camino a una nueva ciencia: **la digitalización de la información**. Ada escribió varios programas para la máquina analítica y diferentes historiadores concuerdan que esas instrucciones la convierten en **la primera programadora de computadoras de la historia**.

Un inventor nacido en Estados Unidos, llamado **Herman Hollerith** desarrolló un tabulador electromagnético de tarjetas perforadas que patentó en 1884. Observó que la mayoría de las preguntas en los censos de la época podían contestarse con opciones binarias: **SÍ** o **NO**. Bajo este principio, ideó una tarjeta perforada compuesta por 80 columnas con 2 posiciones, a través de la cual se contestaban este tipo de preguntas.

Los resultados del censo de 1880 en Estados Unidos demandaron unos siete años de análisis y, según proyecciones de aumento de población, se preveía que el censo de 1890 tardaría casi diez años en procesarse. El gobierno estadounidense eligió la máquina tabuladora de Hollerith para elaborar el censo de 1890, logrando que el resultado del recuento y análisis censal de los aproximadamente sesenta millones de habitantes estuviera listo en sólo seis semanas.

En 1896, con el fin de explotar comercialmente su invento, Hollerith fundó la empresa **Tabulating Machine Company**, que tras algunas fusiones, se convierte, en 1924, en la **International Business Machines**, cuya sigla es **IBM**. ¿Te suena?

A partir del siglo XX, más específicamente en la década del 40, se crearon las primeras computadoras electrónicas, basadas en el sistema binario (sistema de numeración de base 2 que trata las distintas combinaciones de VERDADERO y FALSO, o 0 y 1). La **Z3**, a cargo de **Konrad Zuse**, en 1941. La **Atanasoff Berry Computer** (**ABC**) , obra de **John Vincent Atanasoff** y **Clifford Edward Berry**, entre 1937 y 1942. Luego, llegó el aporte de **Howard Aiken**, quien, en colaboración con IBM, desarrolló la **Mark I** entre 1939 y 1944.

En 1945, el genio matemático **Johann Ludwig Von Neumann** publicó un artículo acerca de una arquitectura que permitía el almacenamiento del programa junto a los datos en un conjunto de celdas que permitía guardar datos binarios, algo que llamó **memoria**. Las computadoras ya no necesitaban recibir las instrucciones una por una, sino que interpretaban los datos binarios guardados en memoria, logrando así, además, una aceleración en los cálculos y prevención de fallas mecánicas.

Ese conjunto de bits (dígitos binarios) se denomina **código máquina** o **lenguaje máquina**. Es lo que finalmente éstas entienden y continúan entendiendo hoy. Para aquella época, la tarea del programador era muy complicada, dado que debía conocer en detalle el hardware de la computadora para poder realizar el programa, paso a paso, en forma de números, lo que hacía el trabajo sumamente propenso a errores y dificultando su comprensión y mantenimiento.

En 1948 se crea en los **Laboratorios Bell** el **transistor**, un componente electrónico semiconductor que impulsó el desarrollo digital, reemplazando a los tubos de vacío.

Para la década del 50, se desarrolla el **lenguaje ensamblador** o **assembler**, que consistía en el uso de **mnemónicos**, es decir, palabras que sustituían los códigos numéricos de las operaciones a realizar, siendo más fácilmente para los programadores humanos recordar palabras abreviadas como **MOV** (mover) o **INC** (incrementar) que números. El código en lenguaje ensamblador finalmente pasaba por un **ensamblador**, (dada la ambigüedad, el primero es el lenguaje y el segundo el utilitario), que convertía estos mnemónicos en código máquina, directamente interpretable por la computadora. Sin embargo, programar en lenguaje ensamblador seguía siendo engorroso, dado que se continuaba más del lado de la máquina. Además, cada una de ellas era distinta, haciendo que ejecutar un programa en lenguaje ensamblador en una máquina diferente requiriera reescribirlo nuevamente.

En años posteriores comienzan a incorporarse científicos de otras ramas, como la física, la química y las matemáticas, a quienes les resultaba muy complicado enfrentarse a un lenguaje ligado a la computación pura. Nace entonces el concepto de **lenguajes de alto nivel**, que son aquellos que contienen una sintaxis más de lado del ser humano, siendo más fáciles de asimilar.

Para que estos códigos puedan ser ejecutados por la máquina, fue necesario el uso de un **compilador**, es decir, un programa que traducía las instrucciones de alto nivel a lenguaje ensamblador y, posteriormente, a código máquina. La pionera fue nuevamente una mujer, **Grace Hopper**, que desarrolló el primer compilador de la historia en 1952.

Los científicos vieron más facilidad en el mundo de la computación al desarrollarse el lenguaje **FORTRAN** (**FOR**mula **TRAN**slation), que tal como su nombre indica, permite la introducción de fórmulas que luego son traducidas a código máquina mediante un compilador desarrollado por el equipo de IBM en 1957.

Posteriormente nacen lenguajes de alto nivel como **LISP** (1958) y **COBOL** (1959). Éstos permitían una abstracción que brindaba una vida más fácil al programador, dejando los detalles específicos de la máquina para los ensambladores y compiladores.

Thomas Kurtz y **John Kemeny** desarrollan, en 1964, el lenguaje **BASIC** (**Beginner' s All-purpose Symbolic Instruction Code**) cuyo objetivo era servir a la enseñanza de la programación. En 1968, **Niklaus Wirth** crea **Pascal**.

Comienzan a aparecer los **paradigmas de la programación**, es decir, filosofías y enfoques diferentes según sea el lenguaje, para resolver problemas. Un artículo de 1966 publicado por **Corrado Böhm** y **Giuseppe Jacopini** sienta las bases del paradigma estructurado. Donde según su **teorema del programa estructurado**, establecen que toda rutina puede implementarse en un lenguaje de programación que solo combine tres estructuras lógicas: **secuencia**, **selección** e **iteración**.

Esto vino a solucionar grandes problemas que había para la época con la sentencia **GOTO** (instrucción que permitía transferir el control a un punto determinado del código), ya que la misma daba grandes dificultades a la hora del seguimiento y la corrección de los programas. En 1968, **Edsger Dijkstra** escribe un artículo donde muestra las contras de emplear esta sentencia y desalienta su utilización.

La década del 70 marcó hitos importantes con el desarrollo del lenguaje **C,** estructurado, por **Dennis Ritchie** en los Laboratorios Bell entre 1969 y 1972. Aparecen otros paradigmas, como la **programación lógica** con el lenguaje **Prolog** en 1972, un dialecto de Lisp llamado **Scheme** (**programación funcional**) en 1975 y a finales de la década, el primer lenguaje de **programación orientada a objetos**, llamado **Smalltalk**.

En los 80s, se profundizó en el estudio de los paradigmas existentes. El paradigma estructurado mostraba dificultades con el empleo de variables globales, por lo que la programación orientada a objetos comienza a ser estudiada con mayor interés. En 1980, se crea un lenguaje **C con clases**, que en 1983 pasa a llamarse **C++**. Evoluciona el hardware con avances en los microprocesadores, haciéndolos más eficientes para los compiladores de lenguajes de alto nivel y ya no tanto para los programadores humanos de lenguaje ensamblador.

En la década del 90, Internet comienza a tener mayor auge, logrando abrir nuevos horizontes. Se crea **Haskell** en 1990. **Python** y **Visual Basic** ven la luz al año siguiente. Para esta época los sistemas operativos ya ofrecían las grandes ventajas de las **interfaces gráficas de usuario** (**GUI**), por lo que los lenguajes de programación se adaptan a ello. El año 1995 vio nacer a **PHP**, **Java** y **JavaScript**, y con ello herramientas que mejoraban la productividad del programador, como los **entornos de desarrollo integrado** (**IDE**) con recolectores de basura y herramientas de depuración.

A partir del tercer milenio, Microsoft crea, en 2001, el lenguaje **C#**, basado en **Java** y lo incorpora a su plataforma **.NET**.

Las tendencias actuales muestran un auge del lenguaje **JavaScript**, ya que, con el avance de la infraestructura de las redes de comunicaciones, todo se orienta hacia la web. De todas maneras, **Java** continúa dominando gran parte del mundo empresarial. El lenguaje C sigue teniendo gran demanda en aplicaciones de sistemas embebidos.

Las grandes empresas de hoy invierten grandes capitales en **machine learning** e **inteligencia artificial**, para desarrollar aplicaciones que permitan analizar grandes cantidades de datos en busca de mejores soluciones y estrategias de marketing.

Programar no es inventar y razonar las cosas desde cero, sino reutilizar componentes de software ya creados para ensamblarlos y generar aplicaciones más robustas. Hoy en día hay presentes cientos de **frameworks** y **librerías** para diversos propósitos que abstraen y simplifican ciertas cuestiones al programador.

Instalación y nociones básicas de PSeInt

Preparando el ambiente

Los algoritmos son independientes de cualquier medio y lenguaje, por eso, la propuesta es verter los primeros conceptos de programación mediante diagramas de flujo y pseudocódigo. Es clave asimilar los fundamentos de la programación antes de pasar a un lenguaje formal de programación, donde cada uno puede tener sus bemoles.

Existe un software cuyo objetivo es promover la enseñanza de los fundamentos de la programación llamado **PSeInt** (abreviatura para Pseudo Intérprete).

Según el autor, **"*PSeInt es una herramienta para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos*".**

Instalación de PSeInt

Instalación

Para proceder a la instalación de PSeInt, se debe descargar el instalador desde la siguiente web:

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

El software está disponible para varias plataformas, aunque las indicaciones serán para **Windows**, dado que es el sistema operativo más utilizado.

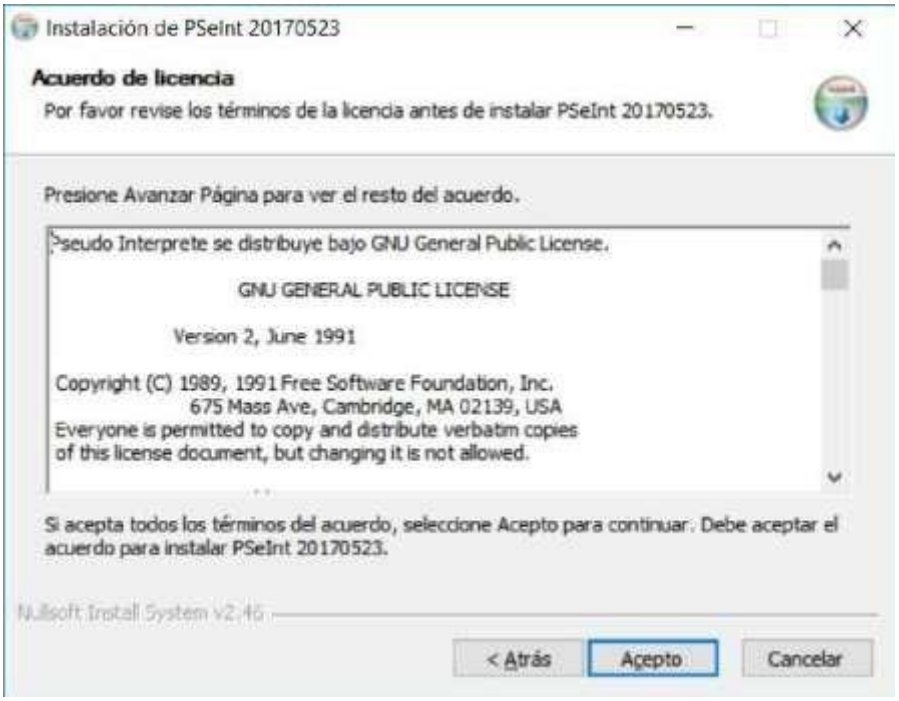
Una vez descargado el instalador, hay que ejecutarlo para que aparezca la siguiente ventana:

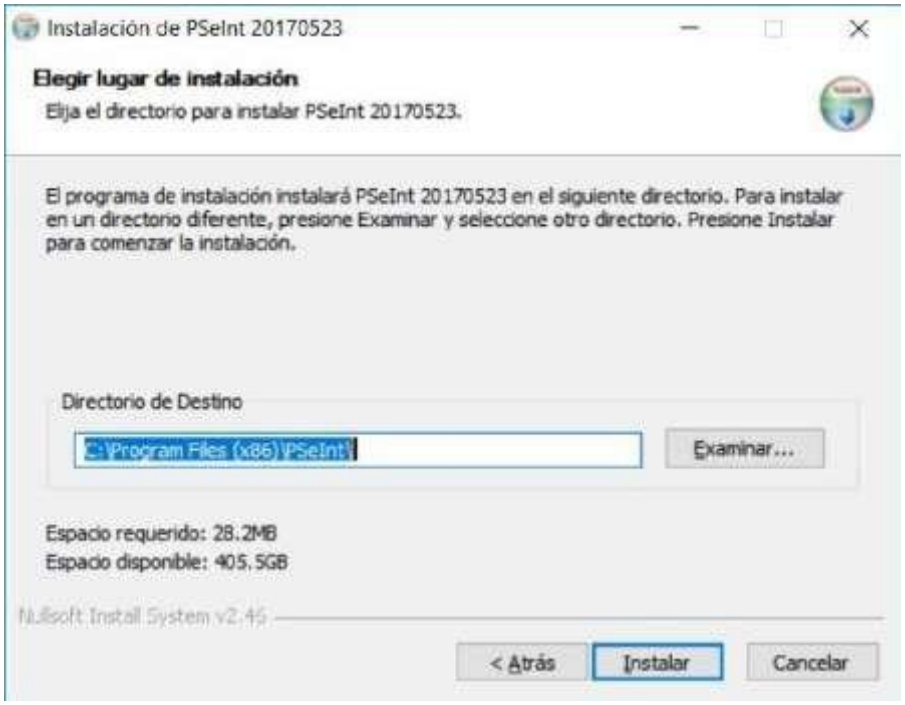
Ventana inicial del instalador

Presionar **Siguiente**. A continuación, aparecerá el acuerdo de licencia:

Acuerdo de licencia de PSeInt

Si estás de acuerdo, se debe presionar **Acepto**. A continuación, aparecerá la selección del directorio de instalación, el cual por defecto es **C:\Program Files (x86)\PSeInt**. Si se desea modificar, se deberá hacer click en **Examinar**.



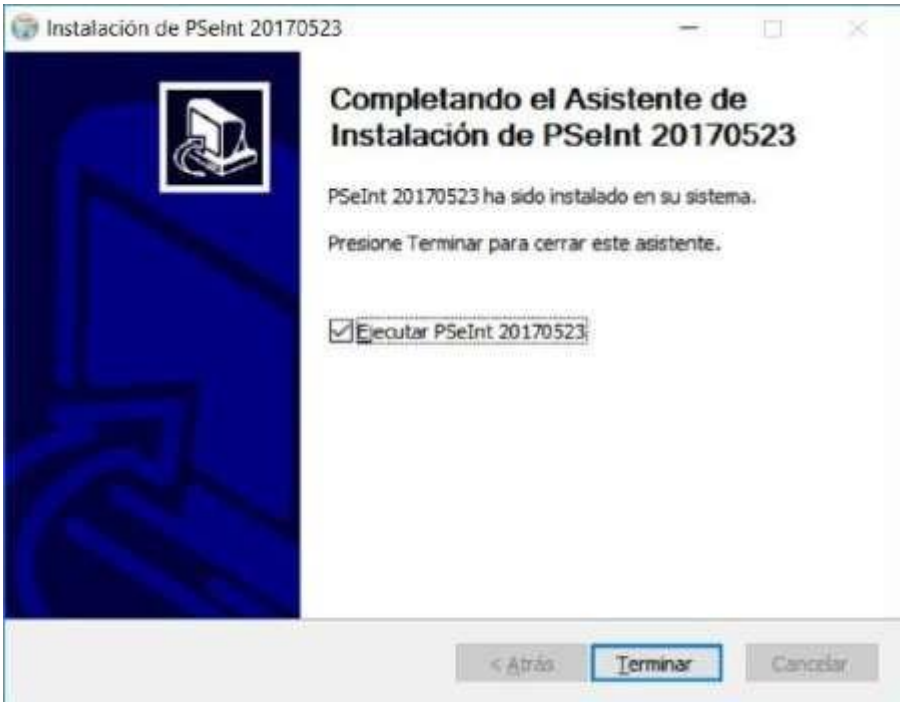


Directorio de instalación de PSeInt

Hacer click en **Instalar** para que comience la copia de archivos. Una vez finalizada, aparecerá la última ventana:

Ventana final del instalador de PSeInt

Dejar tildada la opción de **Ejecutar** y presionar **Terminar** para que se abra el programa.



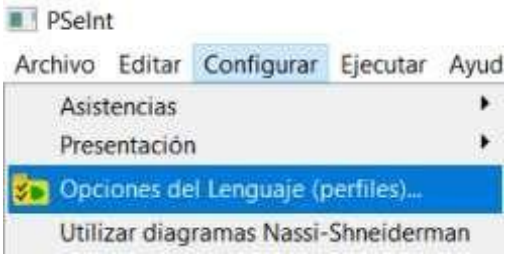
Configuración

Configuración de la sintaxis

PSeInt trabaja con un pseudolenguaje flexible y personalizable a gusto del docente o estudiante. Es importante establecer la misma sintaxis que usaremos en este curso para poder llevar a cabo las prácticas de manera satisfactoria.

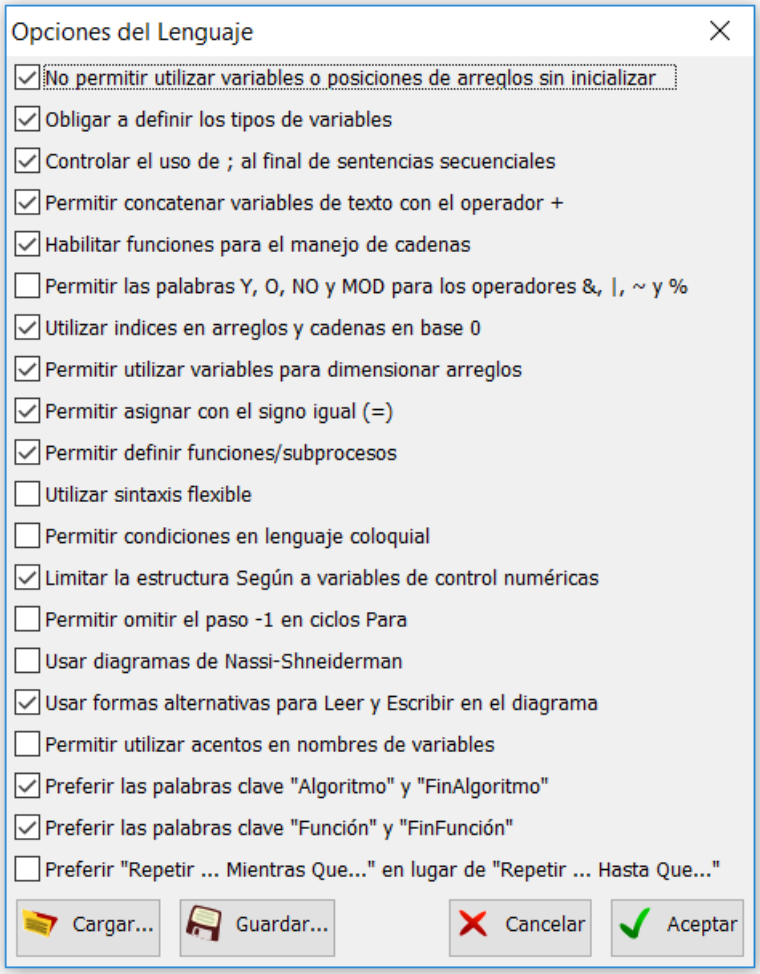
Para configurar la sintaxis del pseudolenguaje, debemos ir al menú **Configurar** y elegir

Opciones del Lenguaje (perfiles)…



Menú Configurar

Allí aparecen listados distintos centros educativos, cada uno con su sintaxis predefinida. Como no está listado el perfil que usaremos, hacer click en el botón **Personalizar**... que se encuentra debajo a la izquierda y asegurase que los parámetros del lenguaje estén marcados exactamente igual que en la siguiente ilustración:



Parámetros del lenguaje usados en este módulo

Una vez terminado, **aceptar todas las ventanas**. El perfil quedará guardado hasta que se modifique, por lo que este proceso se realiza solo la primera vez.

Primer Programa

La interfaz de PSeInt es bastante intuitiva. En el centro de la ventana vemos el editor de código para escribir las instrucciones a ejecutarse.

Para comprobar que todo está en orden, vamos a realizar el primer programa. Dicho sea

de paso, todo estudiante que inicia en el mundo de la programación realiza el famoso **"Hola Mundo!"**.

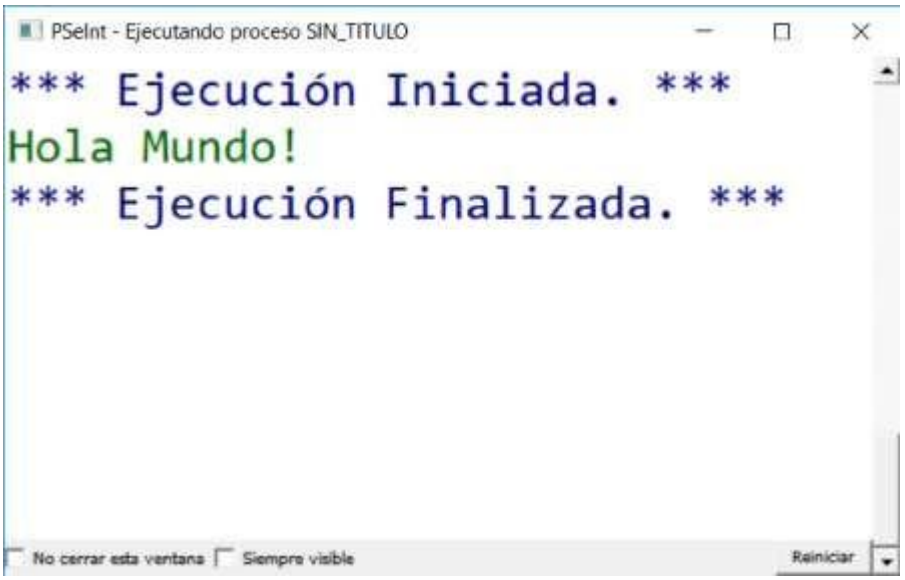
Dentro de las sentencias **Algoritmo** y **FinAlgoritmo**, escribir literalmente la siguiente instrucción:

Escribir "Hola Mundo!";

Editando el primer programa

Si todo va bien (PSeInt no marca errores), se esta en condiciones para ejecutar el programa. Hasta acá, lo que se ve no es más que un texto que sigue una sintaxis particular. La magia ocurre cuando PSeInt hace honor a su nombre e interpreta el pseudocódigo. Presionar el botón **Ejecutar**… o la tecla **F9**.

A continuación, se abre una nueva ventana llamada **consola**, con el resultado del programa que, tal como se lo escribió, es un mensaje "**Hola Mundo!**", sin las comillas, lo cual es correcto.



Ejecución del primer programa

Antes de continuar, sería bueno guardar los pseudocódigos para repasarlos o ejecutarlos en otro momento. A través del botón **Guardar**, se puede guardar el pseudocódigo en un archivo, cuya extensión es **.psc**, simplemente para identificar que se trata de un código de PSeInt, aunque en realidad es un archivo de texto plano que puede abrirse con cualquier editor, como por ejemplo, el bloc de notas.

Para abrir un pseudocódigo ya guardado, debemos presionar el botón **Abrir…** y buscarlo entre los archivo