

Introducción

“Introducción a la programación | Carlos E. Cimino”

Te doy la bienvenida a este apasionante mundo de la programación. No hay requisito previo. Es probable que hayas escuchado que es necesario saber matemáticas o algo por el estilo. Aunque es deseable que tengas un nivel básico de matemática/lógica, quiero llevarte tranquilidad, ya que no necesitás saber ecuaciones diferenciales, derivadas o funciones lineales. No vayas a buscar tus apuntes de la universidad o la secundaria. No creo que tengas la necesidad al principio de saber más que calcular un porcentaje o realizar una regla de tres simple. Esto es como aprender un instrumento (te lo dice alguien que tiene a la guitarra como hobby). Requiere constancia y paciencia. Al principio parece algo intimidante, pero finalmente es cuestión de tomarle la mano. Debés aprender a pensar. No, no te estoy subestimando. Me refiero a que tenés desarrollar lo que se denomina "pensamiento computacional". Esto te ayudará a ser estructurado en tu vida misma, a diseñar soluciones inteligentes y desarrollar un nivel cognitivo que te hará ver las cosas de otra manera. La programación no es novedosa en absoluto, pero está tomando una trascendencia cada vez mayor: el lenguaje del futuro es el de las máquinas. La demanda es brutal, tal es así que cada año quedan miles de puestos vacantes. Pero no tan solo deberías aprender porque la industria requiere de trabajadores. De hecho, podrías armar desde cero tu propio emprendimiento sin mucho requerimiento. Programar es una herramienta que te da la posibilidad de crear. Una idea que ronde en tu cabeza puede ser plasmada a tu gusto. Te aseguro que se siente muy bien. Por favor, no te quedes solo con lo que lees en este libro. Es tanto lo que podés hacer y a su vez con posibilidades tan variadas que no existe escrito alguno que pueda mostrarte todas las maneras. Por eso, sé curioso, cambiá algunas cosas, probá otras. Explorá, agregá, modificá, quitá. ¡Jugá! Vas a aprender más significativamente a través de la prueba y el error que dándote yo todas las pautas. El cerebro es un músculo, y como tal se ejercita. Un rato cada día, con alguno de descanso en el medio, es suficiente. Es preferible sentarse a programar una hora al día para mantenerse activo. Por último, ánimo para las chicas. Aquellos mitos que dan cuenta de que los programadores somos en su mayoría hombres se está derribando cada vez más. Aquí cualquier persona de cualquier género es capaz de pensar y desarrollar soluciones a los problemas. Hecha tal presentación, a mí me dieron ganas de ponerme a programar. ¡Vamos!

Programación - Definición

La programación trata la creación, diseño, codificación, mantenimiento y depuración de un programa o aplicación informática, a través de un código fuente. Un código fuente es un conjunto de líneas de texto que describen a la computadora cómo realizar determinada tarea. Se trata de palabras que siguen determinadas reglas sintácticas según el lenguaje de programación. Existen multitud de lenguajes de programación de distintas características, que luego detallaré. El programa entonces, que constituye la parte lógica de una computadora (software), es ejecutado por la parte física (hardware) proveyendo los resultados correspondientes. Si bien parece que esto se limita solo a las computadoras, deberías saber que existe programación en multitud de dispositivos, como celulares, automóviles, lavarropas, relojes, microondas, etc.

Un poco de historia

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 del 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 (FORmula TRANslation), 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. De todas maneras, lo que hoy es furor, mañana será obsoleto. Son las reglas de juego de esta industria. Tomalo como un desafío y nunca dejes de capacitarte.

¿Por dónde empezar?

Es importante entender que la informática tiene como objetivo automatizar el proceso de trata de información. Los datos normalmente son brindados por el usuario vía dispositivos periféricos de entrada, como pueden ser el teclado, mouse, scanner, cámara, micrófono, etc. La computadora los procesa y genera una salida, vía dispositivos periféricos de salida, como la pantalla, parlantes, auriculares, impresora, etc. Para poder comenzar a programar la computadora, es necesario conocer el concepto de algoritmo.

Algoritmos

Definición

Se trata de un conjunto de indicaciones finitas (me refiero, a que dichas indicaciones tienen un comienzo y un final) y ordenadas de forma lógica que permite la resolución de un problema dado.

Se atribuye el término al matemático y astrónomo persa **Musa al-Juarismi**, quien vivió entre los años 780 y 850.

Con seguridad, habrás leído e interpretado algoritmos sin que supieras que se trataba de ellos. Se me ocurren algunos ejemplos cotidianos con los que quizás hayas interactuado:

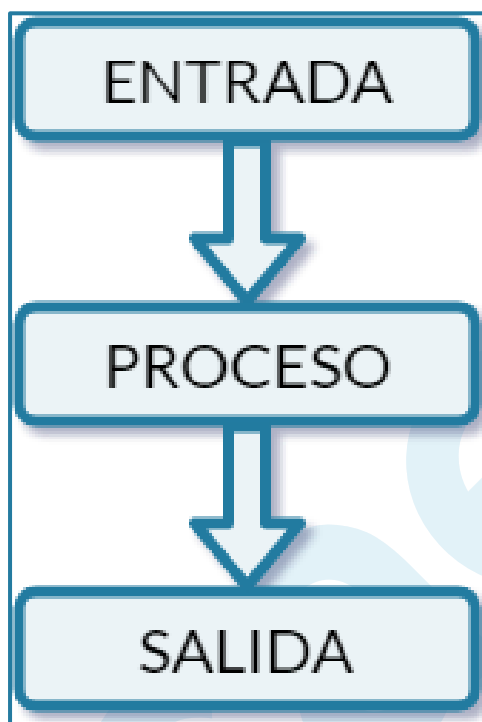


Ilustración 1: Componentes de un algoritmo.

- Un manual de instrucciones para colgar una televisión en la pared.
- Una receta de cocina para preparar un postre.
- Las directivas de un jefe a su empleado.
- Las indicaciones de un GPS para llegar a destino.
- Los pasos para calcular el cociente entre dos números enteros.

La importancia de los algoritmos en la informática es trascendental, dado que la programación tiene como objetivo la implementación de ellos en una computadora, para que sea ésta quien los ejecute y resuelva determinado problema, sin embargo, éstos trascienden la disciplina informática, pudiendo encontrarlos en la matemática o la lógica.

Características

Para que un algoritmo pueda ser considerado como tal, debe cumplir con las siguientes claves:

- Debe ser **preciso**, indicando el orden de realización de cada paso.

- Debe estar **definido**, obteniéndose el mismo resultado si se repite el proceso con los mismos datos de entrada.
- Debe ser **finito**, teniendo un número de pasos que permita llegar a un final.

Representación

Volviendo a los ejemplos de la página 12, podemos expresar los algoritmos en un lenguaje natural cuya intuición y comprensión resultan convenientes, pero con la desventaja de que son imprecisos.

La manera de representar algoritmos de manera precisa y sin ambigüedades es mediante dos maneras formales: como **diagrama de flujo** o como **pseudocódigo**.

Un **diagrama de flujo** es una representación gráfica de un algoritmo. Seguramente te resulta familiar, pues es usado también para describir procesos industriales o de negocio. Son bastante convenientes al principio, dado que son bastante fáciles de entender, gracias a que presentan las instrucciones de una manera gráfica.

La clave para entenderlos es comprender que existen diferentes componentes, cada uno con una forma distinta. La unión de ellos permite formalizar una solución:

Forma	Descripción
	Delimitador. Representa el comienzo o la finalización de la secuencia de instrucciones.
	Entrada. Representa la adquisición de datos por medio de un periférico, como el teclado.
	Proceso. Representa una operación o una tarea.
	Salida. Representa la visualización de los resultados por medio de un periférico, como la pantalla.
	Condición. Representa un interrogante cuya evaluación debe dar únicamente VERDADERO o FALSO .
	Línea de flujo. Representa la dirección de la secuencia de instrucciones.

A continuación, te presento un diagrama de flujo que muestra cómo preparar café:

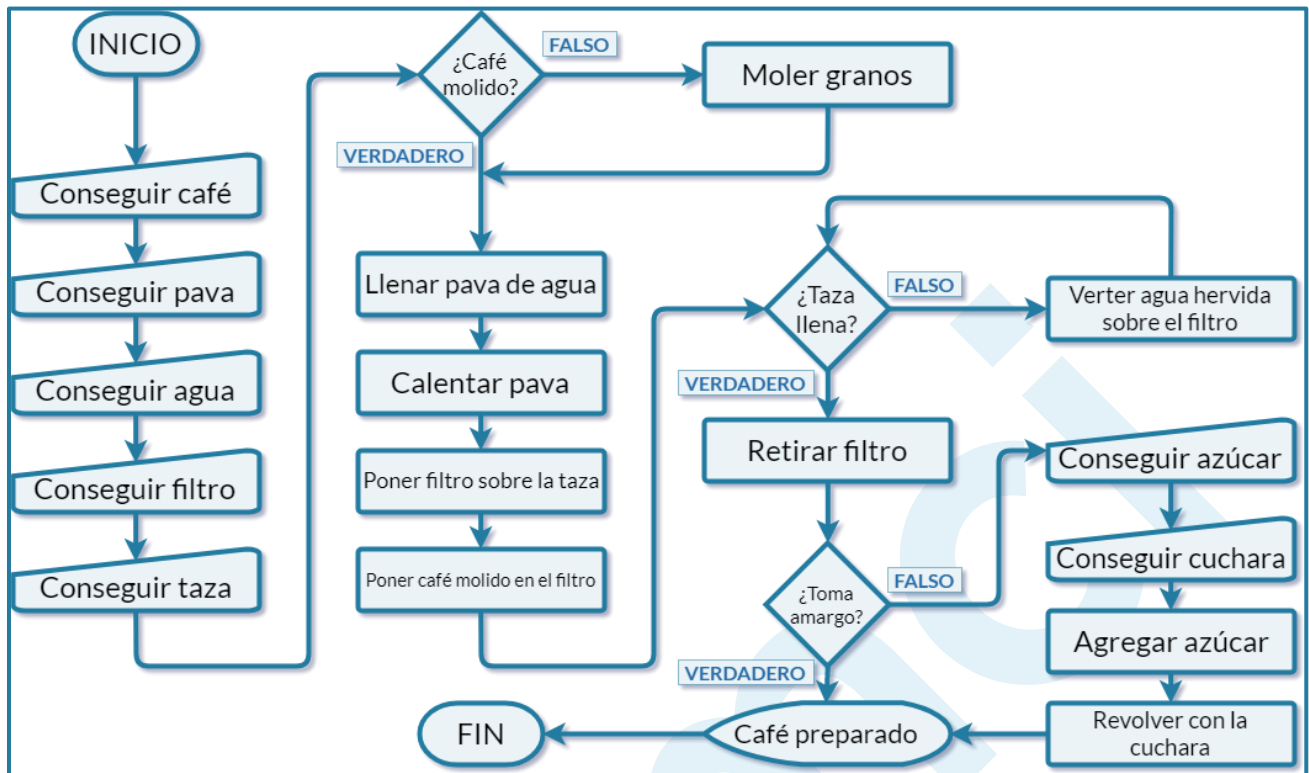


Ilustración 2: Diagrama de flujo de cómo preparar café

La desventaja de los diagramas de flujo es que una computadora no puede interpretarlos directamente. Son simples representaciones gráficas para los seres humanos.

La manera más cercana a la máquina de representar un algoritmo, pero todavía interpretable fácilmente por una persona, es mediante **pseudocódigo**.

Un **pseudocódigo** es una descripción de las instrucciones de manera tal de ser muy similar al formato que se obtiene al utilizar un lenguaje de programación. El punto es que el pseudocódigo no tiene un estándar de reglas sintácticas a seguir, sino que es constituido por convención por uno o más programadores para tener una solución abstracta del problema, algo así como una base para luego transcribir ese algoritmo en un lenguaje de programación real.

A continuación, te presento un posible pseudocódigo correspondiente al problema de preparar café, cuyo diagrama de flujo ya se detalló en la ilustración anterior:

1	INICIO
2	CONSEGUIR café
3	CONSEGUIR pava
4	CONSEGUIR agua
5	CONSEGUIR filtro

6	CONSEGUIR taza
7	SI el café no está molido
8	MOLER café
9	LLENAR pava CON agua
10	CALENTAR pava
11	PONER filtro EN taza
12	PONER café EN filtro
13	MIENTRAS la taza no esté llena
14	VERTER agua EN filtro
15	RETIRAR filtro
16	SI no toma amargo
17	CONSEGUIR azúcar
18	CONSEGUIR cuchara
19	AGREGAR azúcar
20	REVOLVER CON cuchara
21	SERVIR café
22	FIN

Código 1: Cómo preparar café.

Preparando el ambiente

Tal como mostré anteriormente, los algoritmos son independientes de cualquier medio y lenguaje, por eso, mi 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**), el cual utilizaré a lo largo de este libro.

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

Para proceder a la instalación de PSeInt, descargá el instalador desde la siguiente web: <http://pseint.sourceforge.net/index.php?page=descargas.php>

El software está disponible para varias plataformas, aunque mis indicaciones serán para **Windows**, dado que es el sistema operativo más utilizado. Una vez descargado el instalador, ejecutalo para que aparezca la siguiente ventana:

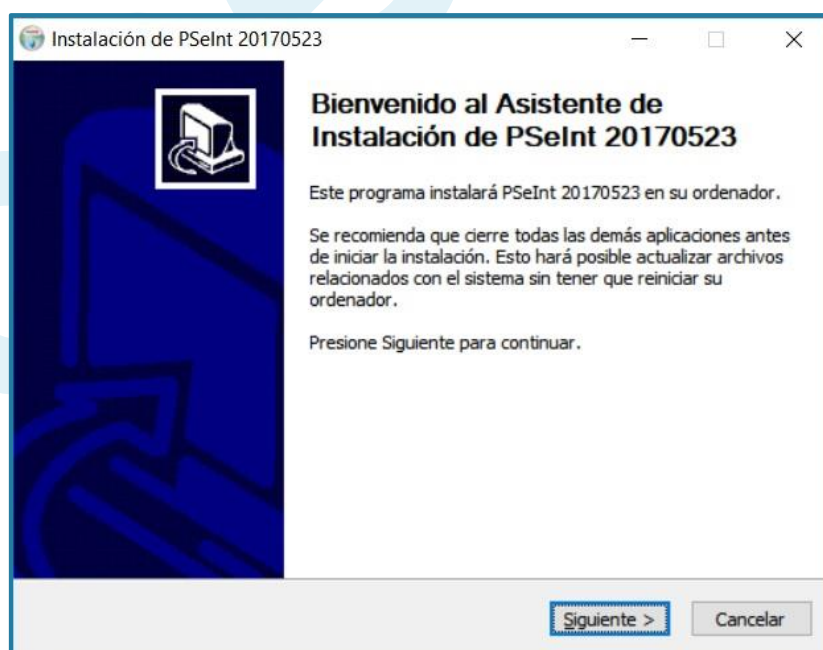


Ilustración 3: Ventana inicial del instalador.

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

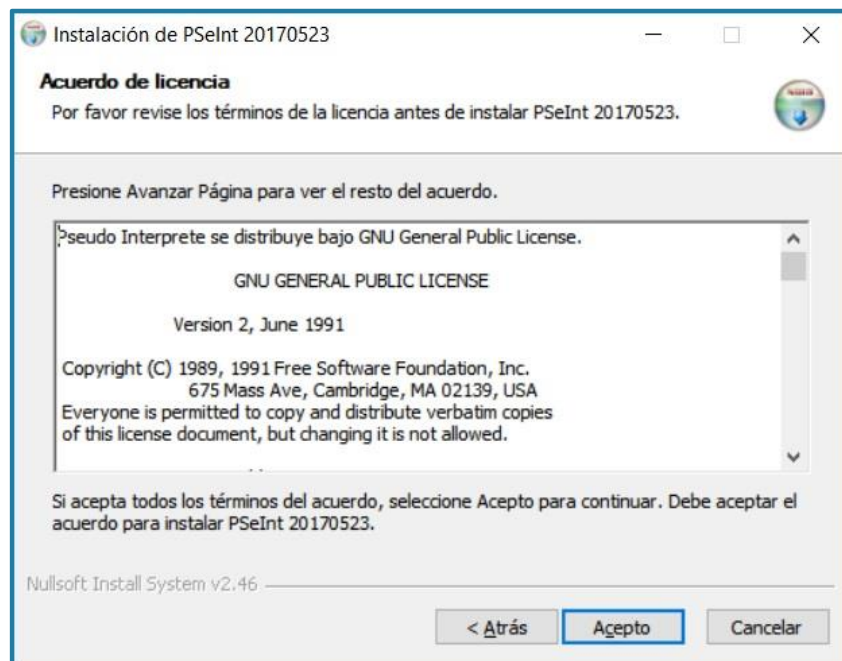


Ilustración 4: Acuerdo de licencia de PSeInt.

Si estás de acuerdo, presioná **Acepto**. A continuación, aparecerá la selección del directorio de instalación, el cual por defecto es **C:\Program Files (x86)\PSeInt**. Si deseás modificarlo, debés hacer click en **Examinar**.

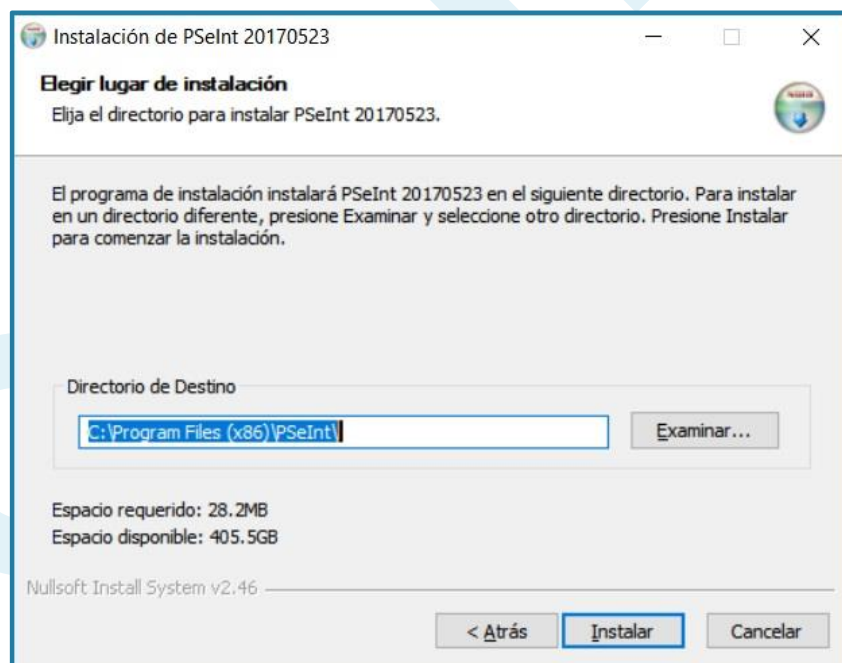


Ilustración 5: Directorio de instalación de PSeInt.

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

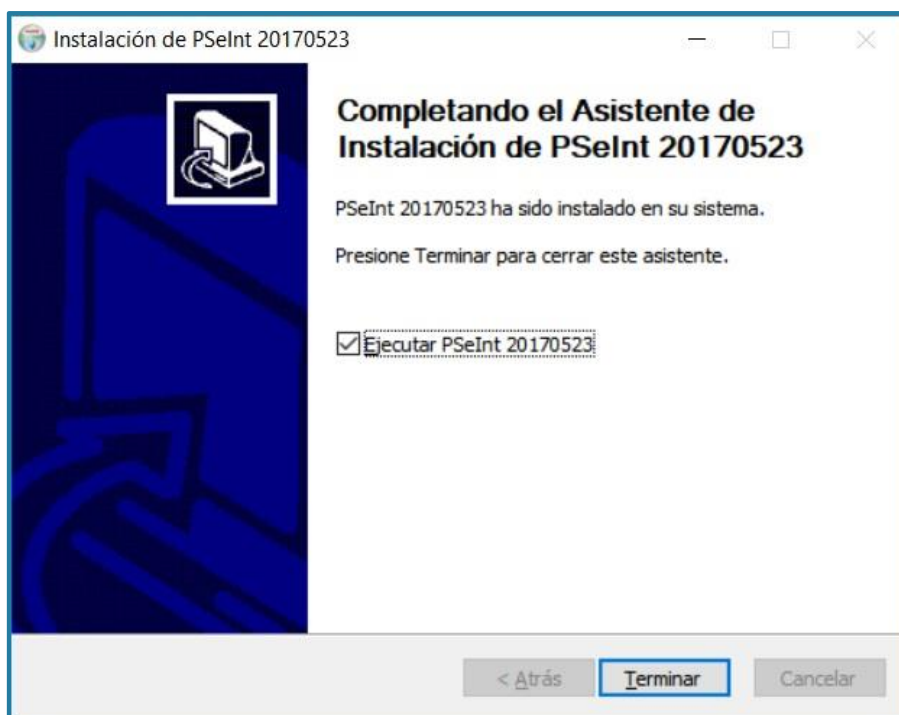


Ilustración 6: Ventana final del instalador de PSeInt.

Dejá tildada la opción de **Ejecutar** y presioná **Terminar** para que se abra el programa.

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 uso en este libro para poder llevar a cabo las prácticas de manera satisfactoria.

Para configurar la sintaxis del pseudolenguaje, debés ir al menú **Configurar** y elegir **Opciones del Lenguaje (perfiles)...**

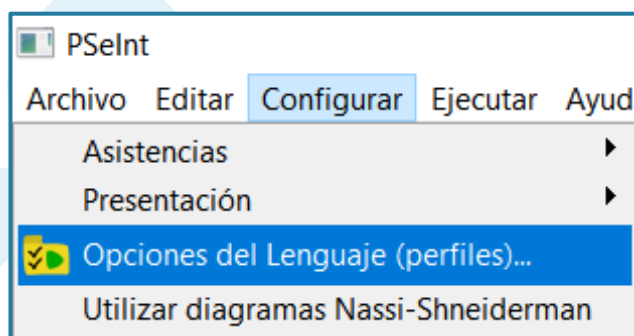



Ilustración 7: Menú Configurar.

Allí aparecen listados distintos centros educativos, cada uno con su sintaxis predefinida. Como no está listado el perfil que usaré en este libro, hacé click en el

botón  **Personalizar...** que se encuentra debajo a la izquierda y asegurate que los

parámetros del lenguaje estén marcados exactamente igual que en la siguiente ilustración:

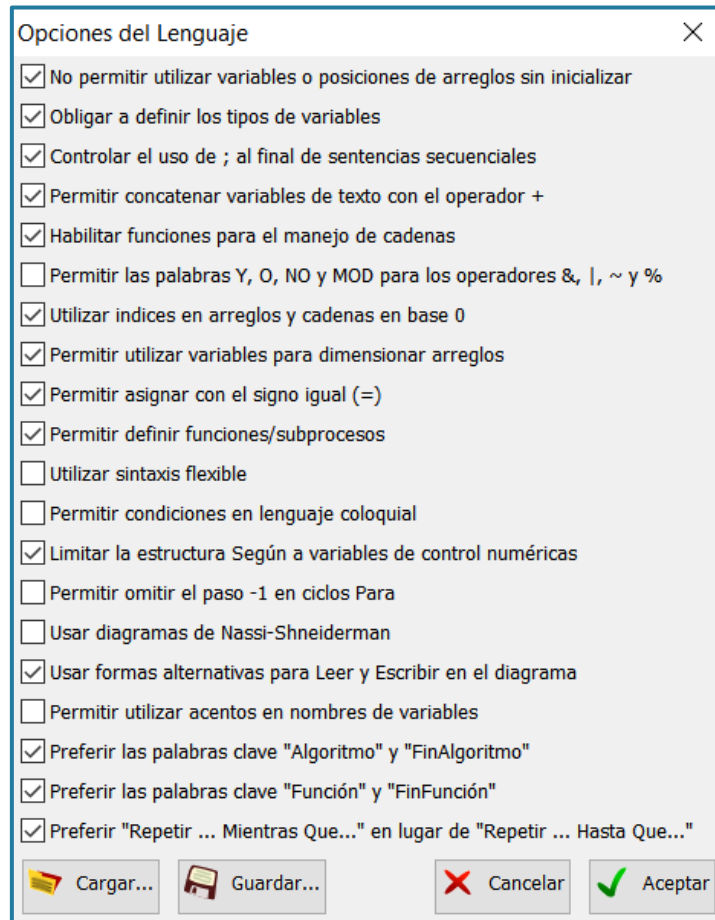


Ilustración 8: Parámetros del lenguaje usado en este libro.

Una vez terminado, **aceptá 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, vas a realizar tu 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**, escribí literalmente la siguiente instrucción:

Escribir "Hola Mundo!";

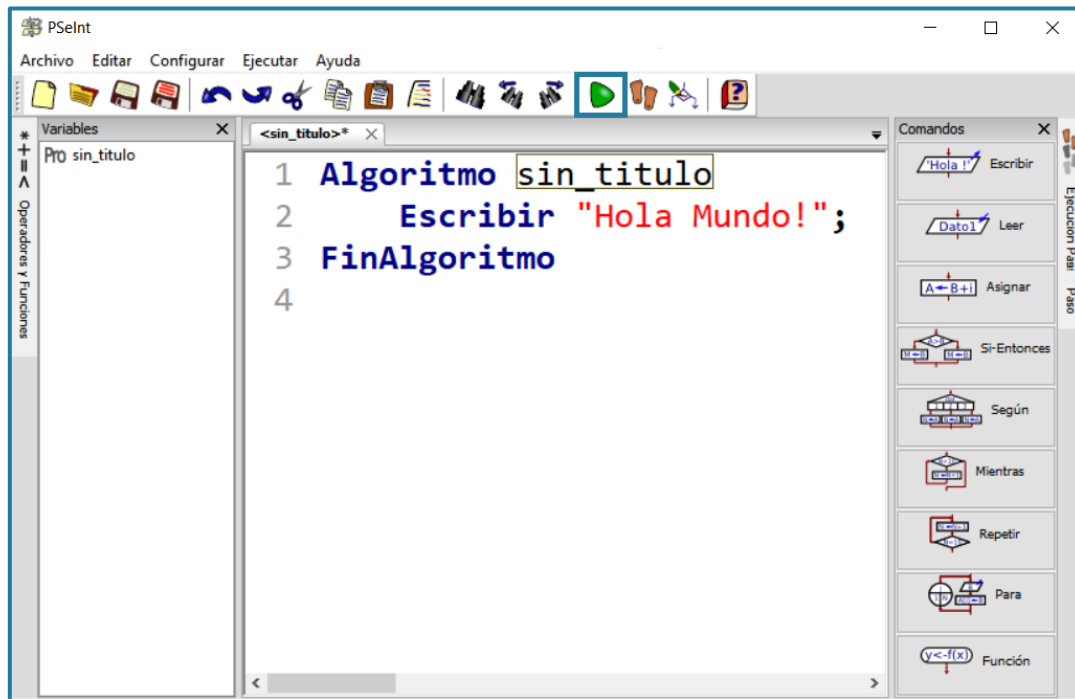



Ilustración 9: Editando el primer programa.

Si todo va bien (PSeInt no marca errores), estás en condiciones para ejecutar tu programa. Hasta acá, lo que ves 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. Presioná el botón  o la tecla **F9**.

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

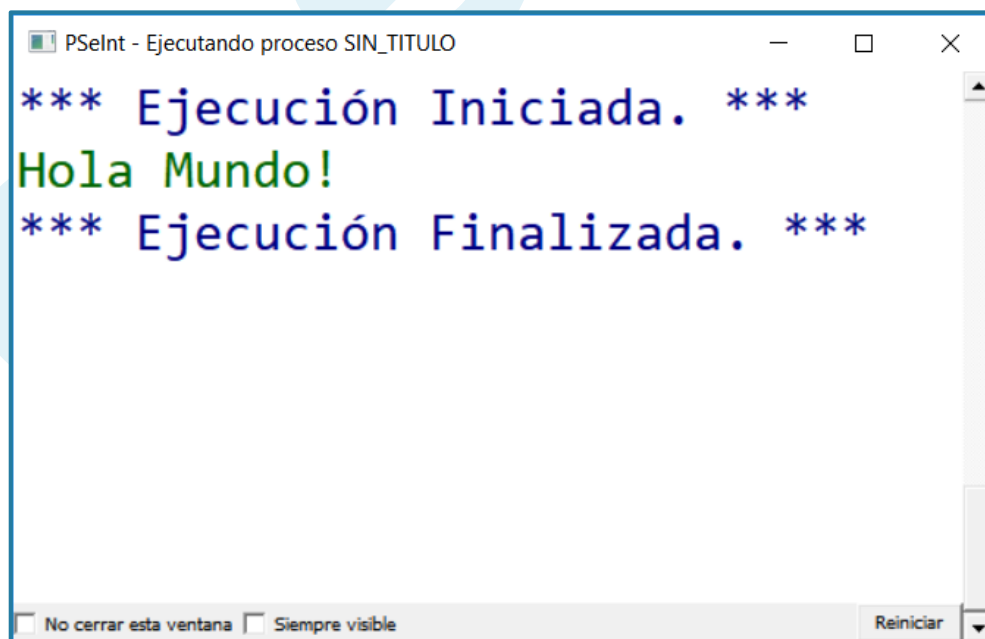


Ilustración 10: Ejecución del primer programa.


Entiendo tu posible frustración. Quizá esperabas que tu primer programa fuera un juego de guerra o un software de facturación. Te pido paciencia.


Hoy en día el software tiene como componente fundamental la **GUI** (siglas en inglés para **Interfaz Gráfica de Usuario**). Es difícil imaginar un programa sin botones, menús, ventanas emergentes, barras deslizables, colores, etc.

Lo que realizarás en este libro son **programas de consola**, es decir, las entradas y salidas no serán mediante componentes gráficos, sino a través de la consola o terminal del sistema.

No es poco lo que puede hacerse en la consola al principio. Lo importante es sentar las bases y fundamentos de la programación para que luego puedas encarar otros paradigmas y conceptos que te permitan construir software con ventanas gráficas.

Todos alguna vez comenzamos por acá, así que, **¡ánimo, ya irás avanzando!**

Antes de continuar, sería bueno que vayas guardando tus pseudocódigos para repasarlos o ejecutarlos en otro momento. A través del botón , podés 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, debés presionar el botón  y buscarlo entre tus archivos.

Errores

Antes de continuar, quiero hacerte una aclaración sobre algo que a menudo ocurrirá: **el error**. No te preocupes, es parte del proceso y hasta los programadores más experimentados no son ajenos a ellos. ¡Somos humanos!

Los errores que se pueden cometer se dividen en dos tipos: **errores en tiempo de compilación** y **errores en tiempo de ejecución**.

Los **errores en tiempo de compilación** son aquellos en los que, cuando el compilador de código detecta que algo no está bien, acusa un mensaje donde describe tal error, haciendo que el programa no pueda si quiera ejecutarse. PSeInt no es compilado, sino interpretado, por lo que la definición anterior no es del todo cierta para este tipo de lenguaje, por ello es que en este caso usaré para el término **errores de sintaxis**.

Para resumir, los **errores de sintaxis**, como, por ejemplo, que a la instrucción:

Escribir "Hola Mundo!";

le quites el punto y coma, harán que no puedas siquiera ejecutar el programa. Probalo y verás.

Los **errores en tiempo de ejecución** son aquellos que se producen cuando el programa ya ha sido ejecutado sin errores sintácticos. En determinado momento, el programa detectará un error y no podrá continuar, ocasionando que finalice de forma abrupta.

Hay errores que no impiden que el programa se ejecute, pero provocan que los resultados quizá sean inesperados. Son **errores lógicos**, y son los más difíciles de detectar, dado que se requiere volver a analizar y probar el código en busca de la falla.

A lo largo de los temas que se irán desarrollando, te mostraré los errores más típicos que se pueden cometer. Recordá que del error es de donde más y mejor se aprende.

caemci