

Capítulo

03



Diseño de **Algoritmos**

Ania Cravero Leal

Jorge Hochstteter Diez

Departamento de Ingeniería de Sistemas
Facultad de Ingeniería, Ciencias y Administración

“Proyecto financiado por el Fondo de Desarrollo Educativo de
la Facultad de Ingeniería, Ciencias y Administración de la
Universidad de La Frontera”

Versión

0.9

TEMARIO

- 3.1 Conceptos Básicos
- 3.2 La Resolución de Problemas
- 3.3 Análisis del Problema
- 3.4 Diseño del Algoritmo
- 3.5 Representación Gráfica de un Algoritmo
- 3.6 Herramientas para la representación y ejecución de Algoritmos
- 3.7 Ejercicios Resueltos
- 3.8 Ejercicios Propuestos
- 3.9 Comentarios Finales
- 3.10 Referencias

Diseño de Algoritmos

El objetivo principal de este capítulo es el aprendizaje y diseño de algoritmos. Este capítulo introduce al lector en el concepto de algoritmo, así como a las herramientas que permiten dialogar al usuario con el computador. La principal razón para que las personas aprendan a diseñar algoritmos (para luego crear programas) es utilizar el computador como una herramienta para resolver problemas. Para resolver un problema, revisaremos los pasos básicos que nos permiten crear algoritmos adecuados, que son: definición o análisis del problema, y diseño del algoritmo. El análisis y el diseño del algoritmo requieren la descripción del problema en subproblemas a base de “refinamientos sucesivos” y una herramienta de representación, diagramas de flujo y pseudocódigo.

3.1 Conceptos Básicos

En el capítulo 1 ya te presentamos una pequeña definición de algoritmo. Un algoritmo es un método para resolver un problema. En cuanto a su historia, **algoritmo** proviene de Mohammed al-Khowârizmî, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales; la traducción

al latín del apellido en la palabra **algorismus** derivó posteriormente en **algoritmo**.

3.2 La Resolución de Problemas

La principal razón para que las personas aprendan a programar, es utilizar el computador como una herramienta para la resolución de problemas. Ayudado por un computador, la resolución de problemas se puede dividir en tres etapas importantes:

- Análisis del problema
- Diseño o desarrollo de algoritmos
- Resolución del algoritmo en el computador

El primer paso –análisis del problema– requiere que el problema sea definido y comprendido claramente para que pueda ser analizado con todo detalle. Una vez analizado el problema, se debe desarrollar el algoritmo –procedimiento paso a paso para solucionar el problema dado–. Por último, para resolver el algoritmo por medio del computador se necesita codificar el algoritmo en un lenguaje de programación como C, C++ y Java, entre otros.

3.3 Análisis del Problema

El propósito del análisis de un problema es ayudar al programador para llegar a una cierta comprensión de la naturaleza del problema. El problema debe estar bien definido si deseamos llegar a una solución satisfactoria.

Para poder definir con precisión el problema se requiere que las especificaciones de entrada y salida sean descritas con detalle. Una buena definición del problema, junto con una descripción detallada de las especificaciones de entrada y salida, son los requisitos más importantes para llegar a una solución eficaz (Luis Joyanes, 2000).

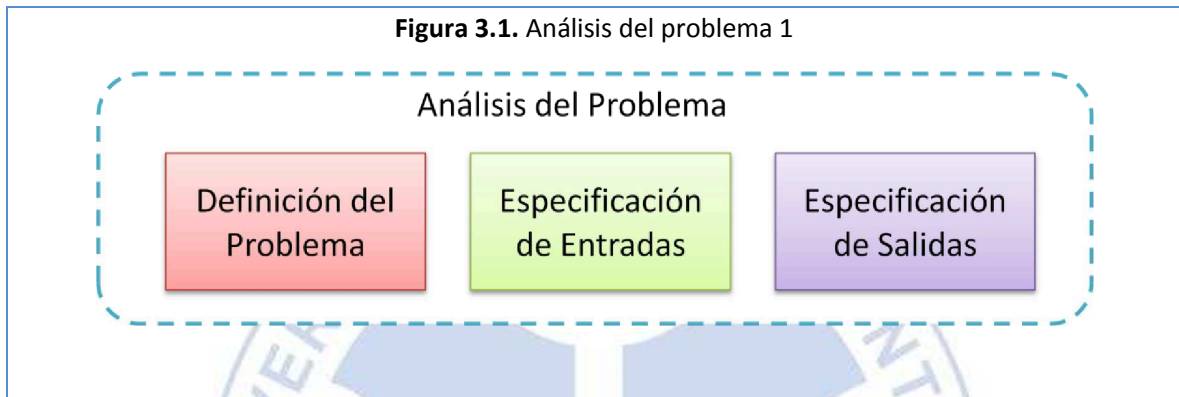
El análisis del problema exige una lectura previa del problema a fin de obtener una idea general de lo que se solicita. La segunda lectura deberá servir para responder a las preguntas:

¿Qué información debe proporcionar la resolución del problema? (salidas)

¿Qué datos se necesitan para resolver el problema? (entradas)

La respuesta a la primera pregunta indicará los resultados deseados a las **salidas del problema**. La segunda pregunta indicará qué datos debe ingresar el usuario o las **entradas del problema**.

Figura 3.1. Análisis del problema 1



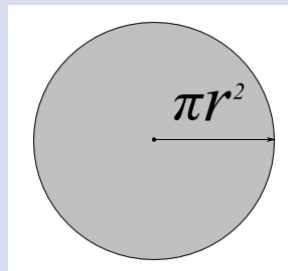
Ejemplo 3.1: El problema a resolver es calcular el área y perímetro de un círculo.

Análisis

Para calcular el área y perímetro de un círculo debemos conocer las fórmulas matemáticas que lo permiten. Así tenemos las siguientes fórmulas:

$$(1) \text{ areaCirculo} = \pi r^2$$

$$(2) \text{ perimetroCirculo} = 2 \pi r$$



Por lo tanto, podemos determinar que las entradas en este problema se concentran en el radio del círculo, ya que π es un valor constante determinado.

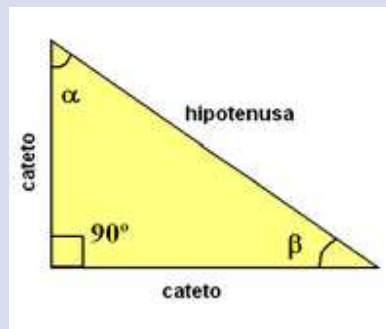
Entradas: radio del círculo (variable RADIO)

Salidas: área y perímetro del círculo (variables AREA y PERIMETRO)

Ejemplo 3.2: El siguiente problema a resolver es calcular el perímetro de un triángulo rectángulo

Análisis

Nuevamente para poder resolver el problema necesitamos conocer las características de un triángulo rectángulo. La figura siguiente muestra una representación.



Para calcular el perímetro del triángulo de la figura 3.3 se sabe que la fórmula es:

$$(1) \text{perimetroT} = \text{catetoBase} + \text{catetoAltura} + \text{hipotenusa}$$

Se sabe además que los catetos y la hipotenusa no pueden tener cualquier valor, ya que se debe cumplir la siguiente relación.

$$(2) \text{catetoBase}^2 + \text{catetoAltura}^2 = \text{hipotenusa}^2$$

Con esta descripción del problema podemos determinar las entradas y salidas de la solución. En primera instancia podríamos pensar que las entradas deben ser tres, catetoBase, catetoAltura y la hipotenusa, ya que la ecuación (1) necesita estos tres datos para calcular el perímetro. Sin embargo, el hecho de ingresar estos tres datos, no asegura que los valores cumplan la relación de la ecuación (2). Por lo tanto, lo correcto será ingresar sólo dos de los tres datos, que puede ser los dos catetos, o un cateto y la hipotenusa.

En cuanto a la salida, lo que se requiere es el área del triángulo como resultado.

Entradas: ambos catetos (variables catetoBase, catetoAltura)

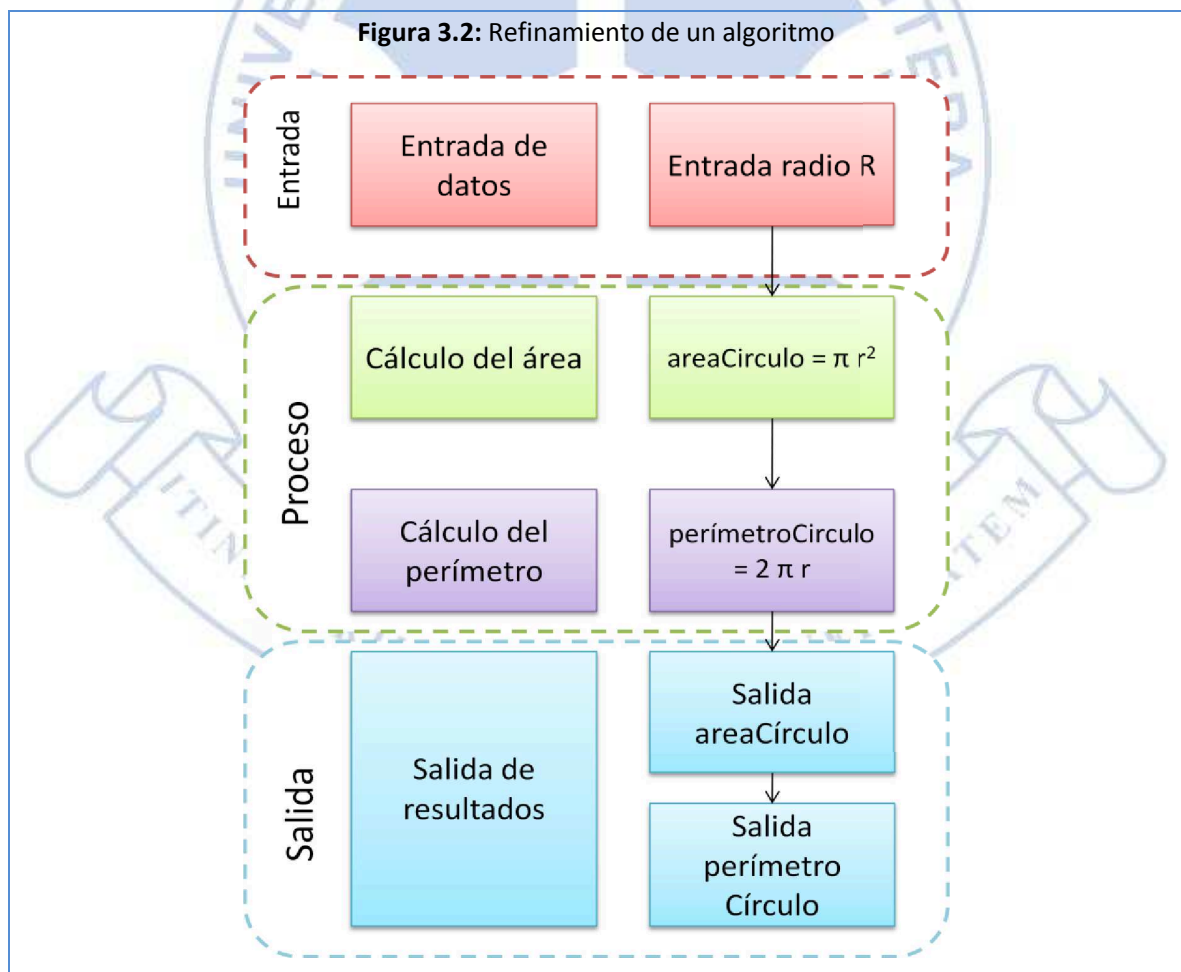
Salidas: perímetro del triángulo rectángulo (variable Perímetro)

3.4 Diseño del Algoritmo

Un algoritmo consiste de una serie de pasos sucesivos (o instrucciones) que permiten resolver un problema. En el capítulo 1 ya explicamos cómo ejecuta las instrucciones un computador, desde las entradas hacia las salidas. Ahora nos concentraremos en el diseño de esta secuencia de pasos para obtener algoritmos eficientes.

Los problemas complejos se pueden resolver más eficazmente con el computador cuando se dividen en subprogramas que sean más fáciles de solucionar que el original. Este método se suele denominar “divide y vencerás”, y consiste en dividir un problema complejo en otros más simples. Así, por ejemplo, el problema de calcular el área y perímetro de un círculo se puede dividir en cuatro problemas más simples o subproblemas (ver figura 3.2)

Figura 3.2: Refinamiento de un algoritmo



La descomposición del problema original en subproblemas más simples y a continuación dividir estos en subproblemas en otros más simples que pueden ser implementados para la solución del problema presentado mediante un computador se denomina “diseño descendente” (diseño top-down). Normalmente los pasos diseñados en un primer esbozo del algoritmo son incompletos e indicarán sólo unos pocos pasos de la solución, por lo que será necesario seguir refinando hasta llegar a una solución completa. Este proceso se llama “refinamiento del algoritmo o refinamiento por pasos”.

Como observamos en la figura del ejemplo 3.2, el problema de cálculo del área y perímetro de un círculo se puede descomponer en cuatro subproblemas más simples: (1) leer datos de entrada, (2) calcular área del círculo, (3) calcular perímetro del círculo, (4) escribir resultados (datos de salida).

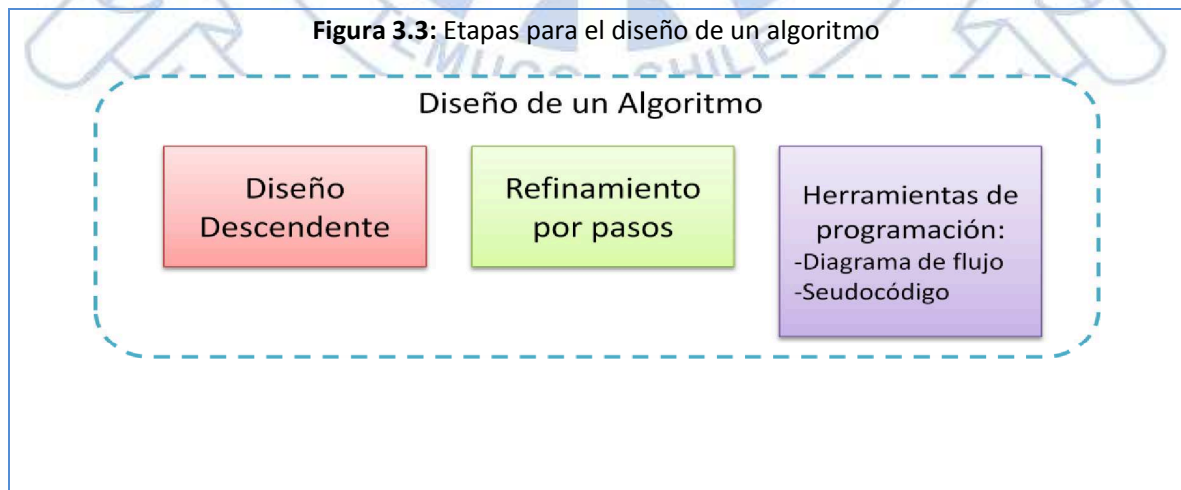
Las ventajas más importantes del diseño descendente son:

- Es más fácil comprender un problema al dividirlo en partes más simples (subproblemas).
- La modificación de un subproblema es más fácil que en todo el problema.
- Se puede verificar si la solución es correcta de manera más fácil.

Tras los pasos anteriores, (diseño descendente y refinamiento por pasos) se requerirá representar el algoritmo mediante una determinada herramienta de programación: diagrama de flujos y pseudocódigo.

De esta manera, el diseño de un algoritmo consiste en llevar a cabo tres etapas (ver figura 3.3).

Figura 3.3: Etapas para el diseño de un algoritmo



3.4.1 Escritura de un Algoritmo

Como ya se ha mencionado anteriormente, un algoritmo consiste en una secuencia de pasos descrita en lenguaje natural. Específicamente, un algoritmo es un método o conjunto de reglas para solucionar un problema. Estas reglas tienen las siguientes características:

Deben estar seguidas de alguna secuencia definida de pasos hasta que se obtenga un resultado coherente,

Sólo puede ejecutarse un paso (instrucción) a la vez.

El flujo de control comúnmente es secuencial.

Ejemplo 3.3:

La siguiente secuencia de pasos describe el proceso para comprar productos en un supermercado.

Ir al supermercado

Seleccionar los productos

Pagar en la caja

Regresar a la casa

El algoritmo consiste en cuatro acciones básicas, cada una debe ser ejecutada antes de realizar la siguiente. Para el computador, cada una de estas acciones corresponderá a un conjunto de instrucciones (vario pasos) que resuelven una tarea en particular.

El algoritmo descrito es muy sencillo, sin embargo, como ya se ha indicado en párrafos anteriores, éste algoritmo general debe ser descompuesto en otras acciones simples (refinamiento por pasos) con el fin de entregar una solución al problema de manera más eficaz.

Así, por ejemplo, un primer refinamiento del algoritmo anterior se puede describir de la siguiente forma:

Ir al supermercado en auto y estacionarse

Entrar al supermercado

Sacar un carro para compras

Repetir

Recorrer uno de los pasillos

Seleccionar productos

Ingresar productos al carro

Fin repetir

Ir hacia la caja

Hacer cola

Poner productos en la cinta

Pagar boleta

Colocar productos en carro

Llevar productos al auto

Regresar a la casa

Sacar productos del auto

Guardar productos

A continuación podemos seguir refinando el algoritmo anterior agregando pasos a las acciones que aún no quedan totalmente resueltas. Por ejemplo, para la acción número 2 podemos hacer lo siguiente:

Repetir

Escoger uno de los pasillos para recorrerlo

Repetir

Seleccionar un producto

Ingresar producto al carro

Seguir caminando en el pasillo

Fin Repetir

Fin repetir

Esto ocurre, ya que por cada producto que seleccionamos en un determinado pasillo, normalmente lo colocamos en el carro antes de seleccionar otro producto.

3.5 Representación Gráfica de un Algoritmo

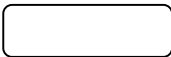


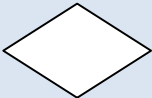

El diseño de un algoritmo se debe representar en una herramienta de programación como: el diagrama de flujo y el pseudocódigo, con el fin de independizar dicho algoritmo del lenguaje de programación que a continuación se debe seleccionar para obtener un programa. Ello permitirá que un algoritmo pueda ser codificado indistintamente en cualquier lenguaje.

3.5.1 Diagramas de Flujo

Un **diagrama de flujo** es una de las técnicas de representación de algoritmos más conocida y utilizada, aunque su empleo ha disminuido considerablemente, sobre todo desde la aparición de lenguajes de programación estructurados (Luis Joyanes, 2000).

La representación de un diagrama de flujo se realiza utilizando los símbolos de la tabla 3.1. Cada símbolo representa un paso del algoritmo por lo que deben estar conectados mediante flechas (líneas de flujo) con el fin de guiar la secuencia de los pasos.

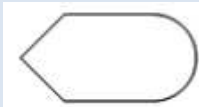
Tabla 3.1: Símbolos más utilizados del diagrama de flujo

Símbolos Principales	Función que desempeña
	Terminal (representa el “inicio” y “fin” del algoritmo)
	Entrada/Salida (cualquier dato de “entrada” al algoritmo o “salida” hacia un periférico)
	Proceso (cualquier tipo de operación que pueda originar un cambio de valor en las variables, formato o posición de la información almacenada en memoria, operaciones aritméticas, etc.)
	Decisión (indica operaciones lógicas o de comparación entre datos)
	Conector (sirve para enlazar dos partes cualquiera en un algoritmo)

Símbolos Secundarios **Función que desempeña**



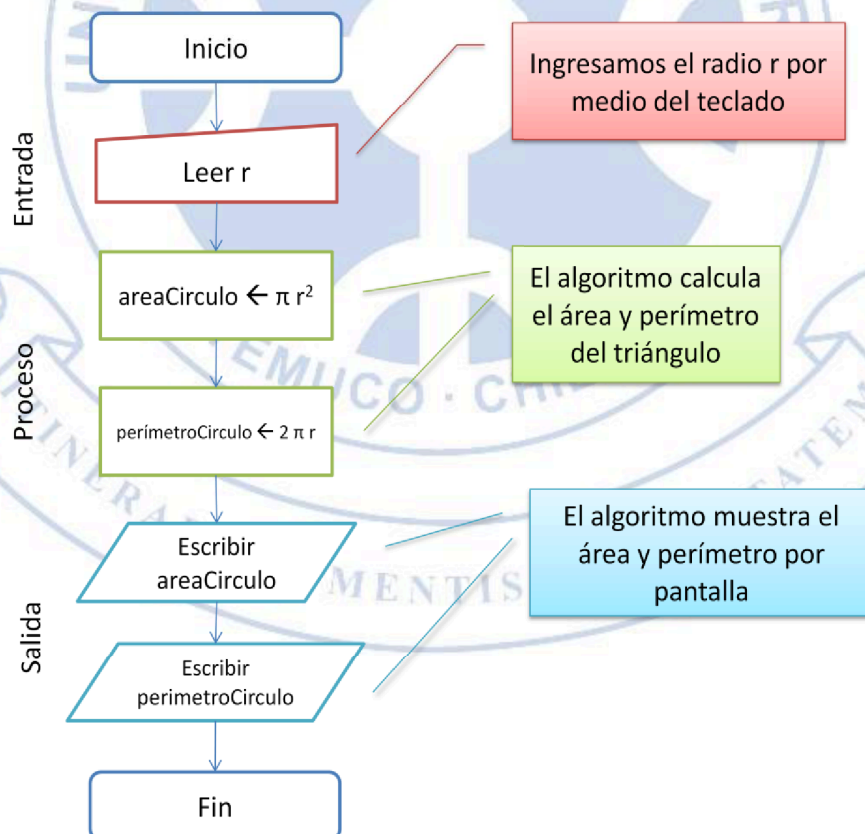
Teclado (se utiliza en ocasiones para las entradas de datos por teclado)



Pantalla (se utiliza en ocasiones para las salidas en pantalla)


El diagrama de flujo para el ejemplo 3.1 se muestra en la figura 3.4.


Figura 3.4: Ejemplo de diagrama de flujo





Observamos que el algoritmo se ejecuta desde arriba hacia abajo, y paso a paso, instrucción por instrucción. También observamos que para crear un algoritmo siempre debemos seguir la secuencia Entrada → Proceso → Salida.

Cada símbolo visto anteriormente indica el tipo de operación a ejecutar y el diagrama ilustra gráficamente la secuencia en la que se ejecutan las operaciones.

Las líneas de flujo (→) representan el flujo secuencial de la lógica del programa. Un rectángulo () significa algún tipo de proceso que se debe ejecutar de manera interna en el computador, es decir, acciones a realizar (ejemplo, sumar dos números, calcular la exponencial, etc.).

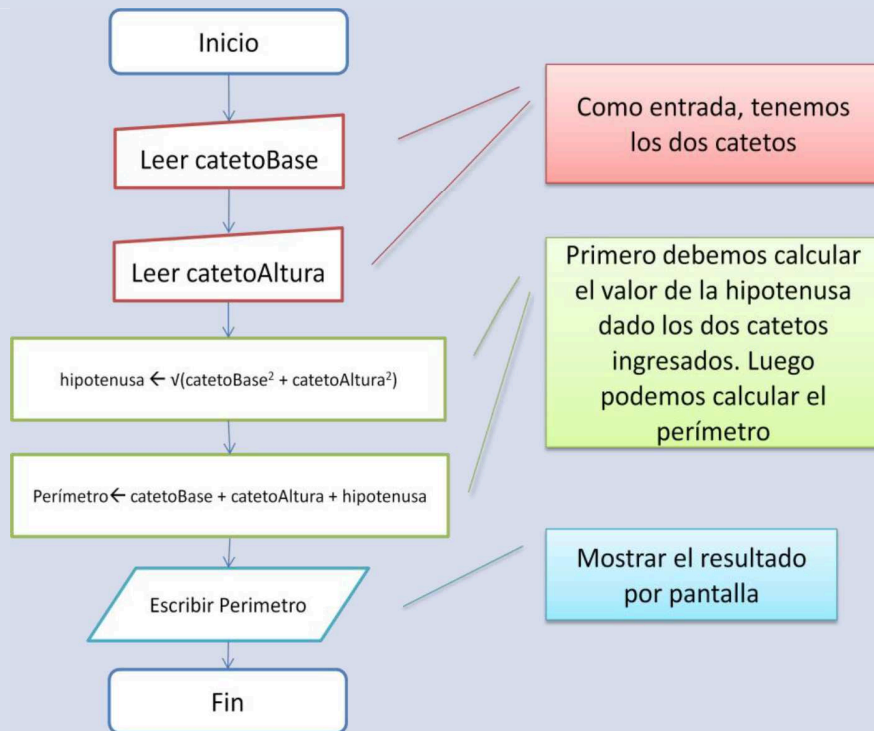
El símbolo de salida de datos () o paralelogramo, es un símbolo de entrada/salida de datos que representa cualquier tipo de entrada o salida desde el programa o sistema (ejemplo, entradas desde el teclado, salidas por la pantalla).

Cabe notar que en la figura 3.4 el símbolo se ha utilizado para representar sólo la salida de los datos. El símbolo rombo () es una caja de decisión que representa respuestas del tipo si/no a preguntas realizadas en base a una condición lógica.

Cada diagrama de flujo se inicia y finaliza con el símbolo terminal ()

Ejemplo 3.4: Para el problema presentado en el ejemplo 3.2 se requiere diseñar un diagrama de flujo que solucione el problema.

Figura 3.5: Diagrama de flujo para calcular el perímetro de un triángulo

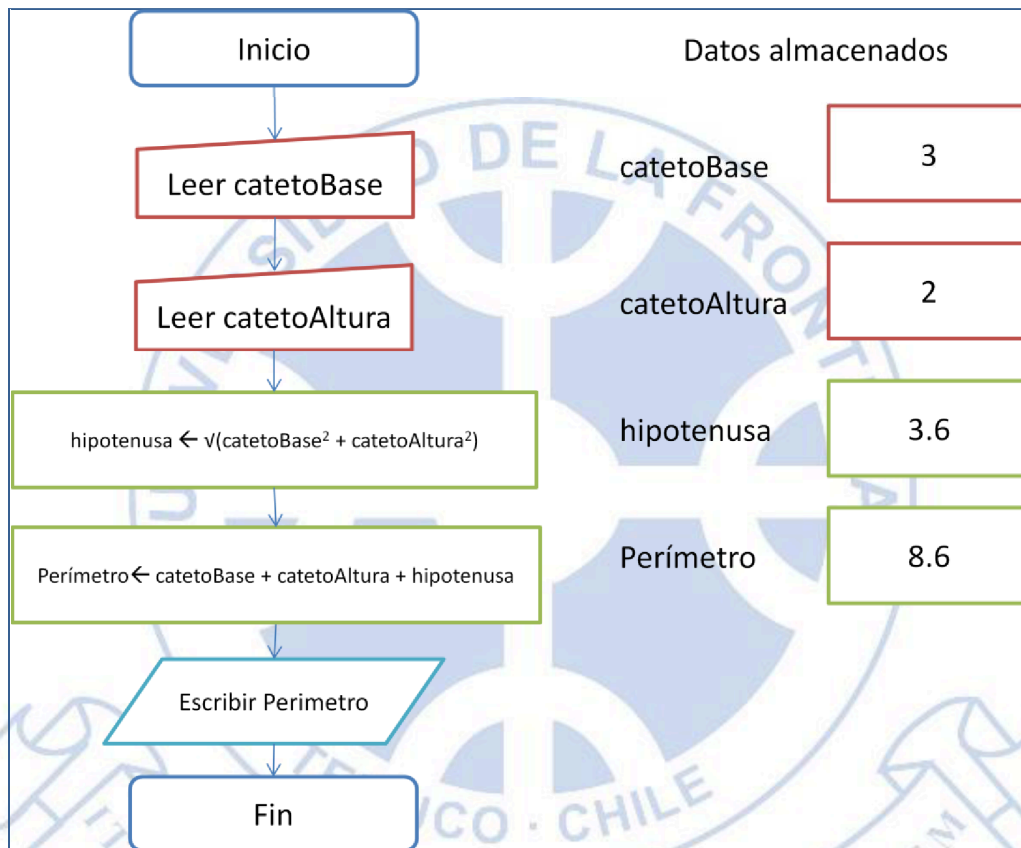


La figura 3.5 muestra la solución del problema con un diagrama de flujo. Observamos que lo primero es ingresar los datos de entrada, que en este caso serán los dos catetos. Luego, para calcular el perímetro del triángulo, necesitamos los valores de los catetos y la hipotenusa, por lo que será necesario calcular primeramente la hipotenusa. En esta caso, asignamos (con el símbolo \leftarrow) $\sqrt{\text{catetoBase}^2 + \text{catetoAltura}^2}$ a la variable hipotenusa, y asignamos $\text{catetoBase} + \text{catetoAltura} + \text{hipotenusa}$ a la variable Perímetro. Finalmente, podemos mostrar el resultado por pantalla (el resultado está almacenado en la variable Perímetro).

La figura 3.6 muestra un ejemplo de ejecución del algoritmo. En este caso, el usuario del programa ingresa los valores 3 y 2 para los catetos (catetoBase y luego catetoAltura). Una vez ingresados estos datos, podemos calcular la hipotenusa; para ello, el programa asignará $\sqrt{3^2 + 2^2}$, lo que equivale a 3.6 que es

asignado en la variable hipotenusa. Una vez obtenida la hipotenusa, el algoritmo calcula un valor para el perímetro asignando $3+2+3.6$ a la variable Perímetro. Finalmente, el resultado es mostrado en pantalla.

Figura 3.6: Ejemplo de ejecución del algoritmo cuando el usuario ingresa 3 y 2 para cada cateto.



3.5.2 Seudocódigo

El pseudocódigo es un lenguaje de especificación (descripción) de algoritmos. La finalidad de este lenguaje es hacer más fácil la traducción del diseño del algoritmo en un lenguaje de programación específico.

La ventaja del pseudocódigo es que el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje de programación específico. Otra ventaja, es que es fácil modificar un algoritmo en caso de encontrar errores.

Para diseñar un algoritmo mediante pseudocódigo, debemos conocer las palabras reservadas para cada acción, si son de entradas de datos, proceso o de salida. Estas suelen ser las descritas en la tabla 3.2.

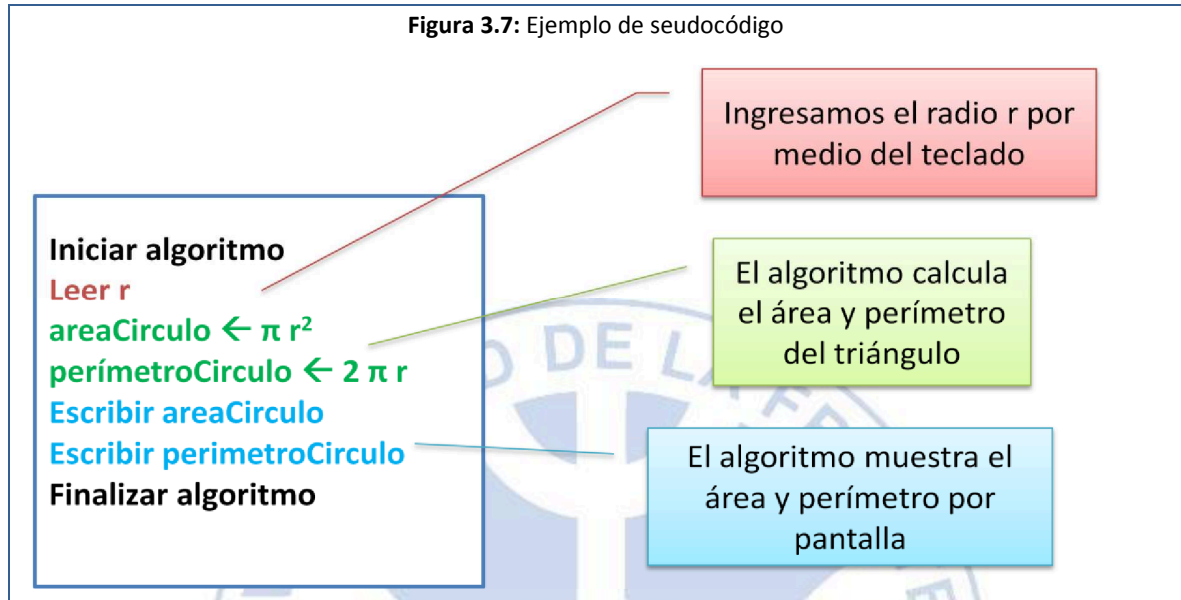
Tabla 3.2: Palabras reservadas utilizadas en pseudocódigo

Palabra Reservada	Función que desempeña	Ejemplo
Leer, Ingresar, Introducir	Estos son algunos ejemplos de palabras reservadas para ingresar datos en una variable	Leer numero
Si- Sino	Se utilizan para especificar una condición lógica	Si (numero >10)
Mientras, Hacer mientras	Se utilizan para indicar que hay instrucciones que se deben ejecutar más de una vez	Mientras (numero >0)
←	La flecha se utiliza para asignar un dato a una variable	Numero ← 20
Escribir, Mostrar, Imprimir	Estos son algunos ejemplos de palabras reservadas para mostrar un dato o una frase por pantalla	Escribir numero

Cabe destacar que las palabras reservadas de la tabla 3.2 no son las únicas, ya que es posible aplicar otras palabras que indiquen la misma acción, por ejemplo, podemos utilizar la palabra *Repetir* en vez de *Mientras* o *Hacer Mientras*.

La figura 3.7 muestra un ejemplo de pseudocódigo que resuelve el problema presentado en el ejemplo 3.1.

Figura 3.7: Ejemplo de pseudocódigo



3.6 Herramientas para la Representación y Ejecución de Algoritmos

Existen varias herramientas que son muy útiles para diseñar algoritmos y ejecutarlos como si fueran un programa. Sin embargo, en este curso sólo utilizaremos dos de ellas. La herramienta FreeDFD.exe que nos servirá para diseñar y ejecutar algoritmos mediante diagrama de flujo, y PSeInt que nos servirá para diseñar y ejecutar pseudocódigo.

3.6.1 Herramienta FreeDFD

"FreeDFD es un editor e intérprete de diagramas de flujo. Permite editar, ejecutar y depurar algoritmos representados como diagramas de flujo. Fue pensado para la enseñanza de algoritmos básicos, pero se puede usar para construir algoritmos complejos usando recursión y arreglos de varias dimensiones (<http://wiki.freaks-unidos.net/freedfd>).

La versión inicial de FreeDfd fue escrita en 1996-1997 por: Fabián Cárdenas Varela, Eduardo Daza Castillo, Nelson Castillo Izquierdo.

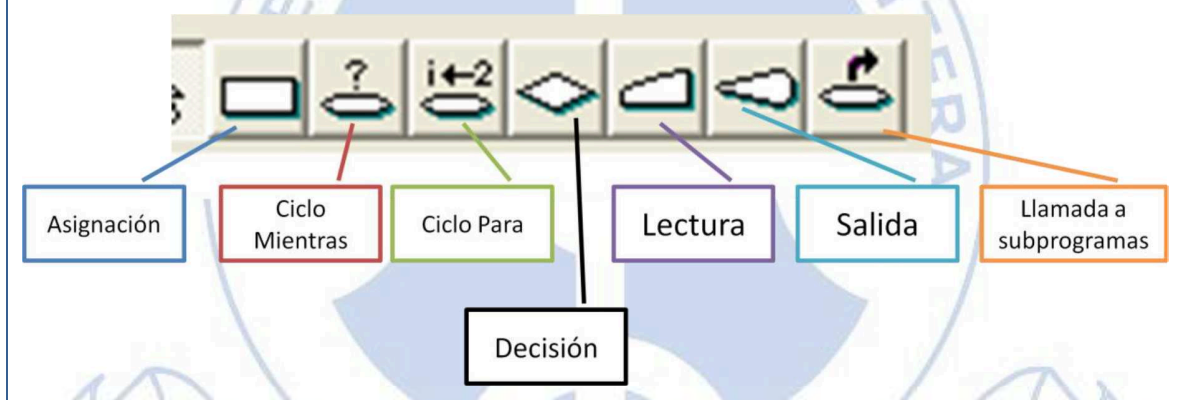
Una vista general de la barra de herramientas se presenta en la figura 3.8.

Figura 3.8: Barra de herramientas de FreeDFD



Y en la figura 3.9 se presenta una vista ampliada de los “Botones de objetos”.

Figura 3.9: Botones de objetos de FreeDFD

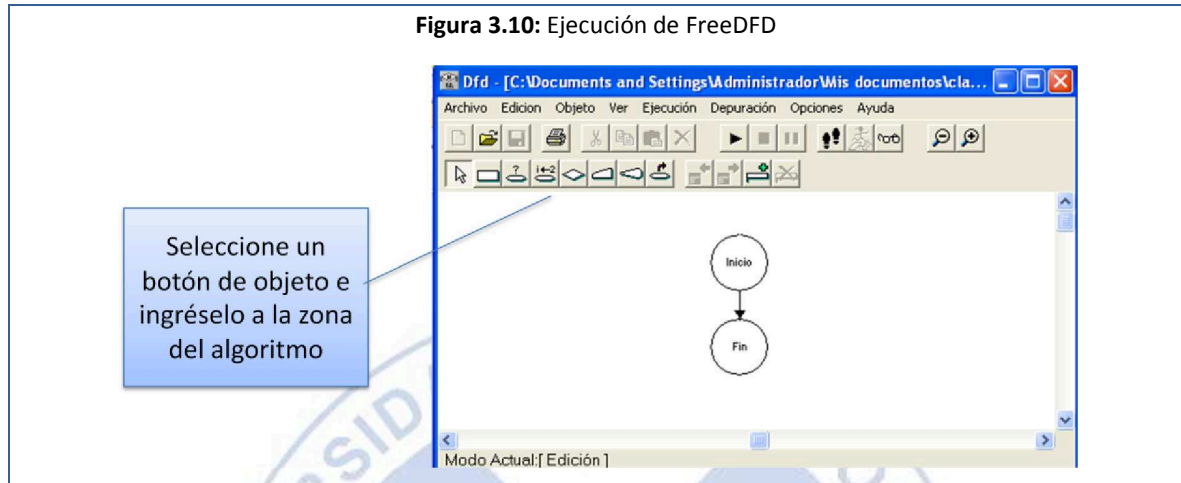


Para ejecutar FreeDFD, primeramente debe descargar la herramienta desde la página web de los creadores. Luego debe descomprimir el archivo en formato zip. Para ejecutar FreeDFD sólo debe hacer doble clic en el siguiente ícono que se encuentra dentro de la carpeta descomprimida:



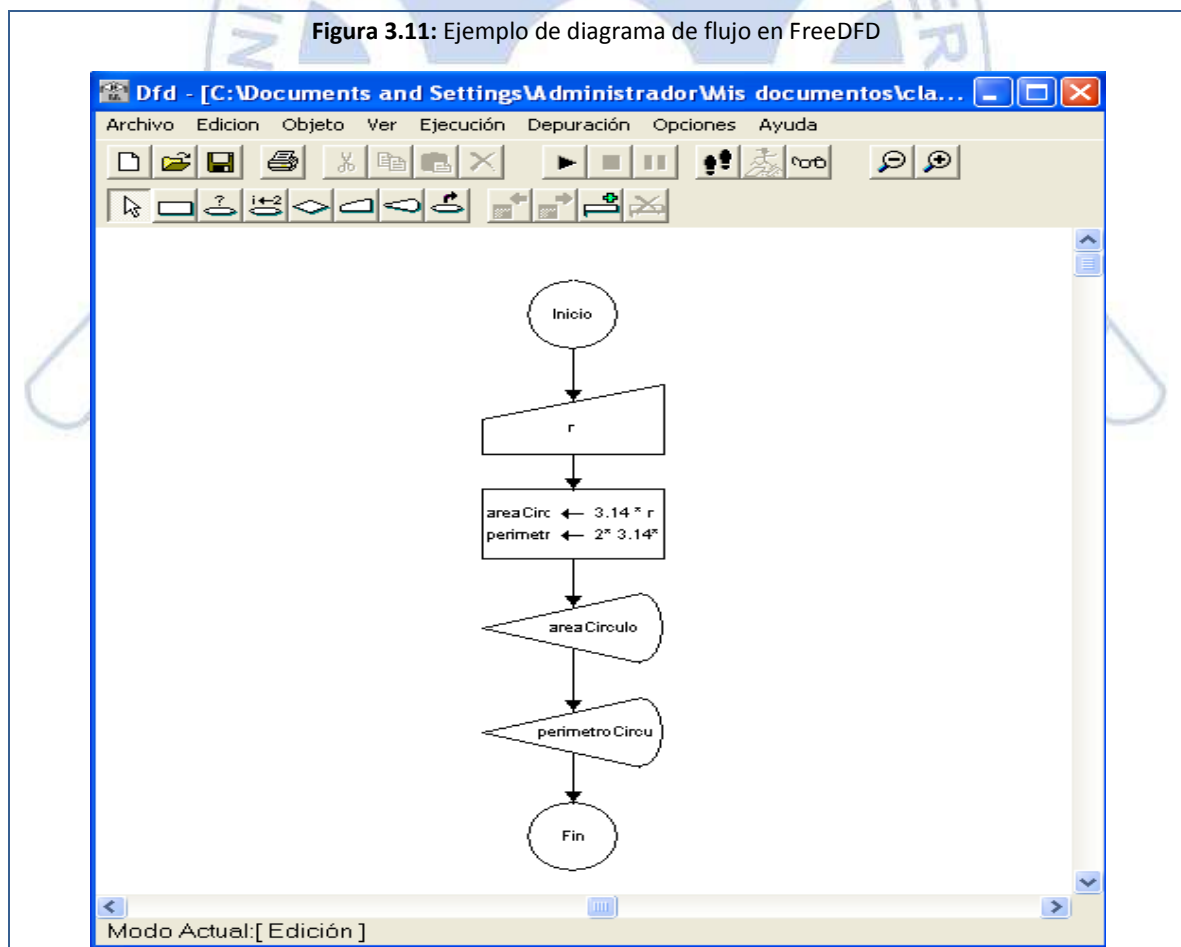
A continuación se iniciará FreeDFD tal y como se muestra en la figura 3.10.

Figura 3.10: Ejecución de FreeDFD



La figura 3.11 muestra un ejemplo de diseño de diagrama de flujo para el problema presentado en el ejemplo 3.1.

Figura 3.11: Ejemplo de diagrama de flujo en FreeDFD



Una vez creado el diagrama de flujo, podemos ejecutarlo haciendo clic en el botón “ejecución” de la barra de herramientas.

3.6.2 Herramienta PSeInt

PSeInt es una herramienta para aprender la lógica de programación, orientada a estudiantes sin experiencia en dicha área. Mediante la utilización de un simple y limitado pseudo-lenguaje intuitivo y en español, permite comenzar a comprender conceptos básicos y fundamentales de un algoritmo computacional. Nacido originalmente como proyecto final para la materia *Programación I* de la carrera *Ingeniería en Informática* de la *Facultad de Ingeniería y Ciencias Hídricas* de la *Universidad Nacional del Litoral*, es en realidad un intérprete de pseudocódigo basado en los contenidos de la cátedra de *Fundamentos de Programación* de dicha carrera. Pueden encontrar mayores detalles de la Historia y su documentación en <http://pseint.sourceforge.net/>

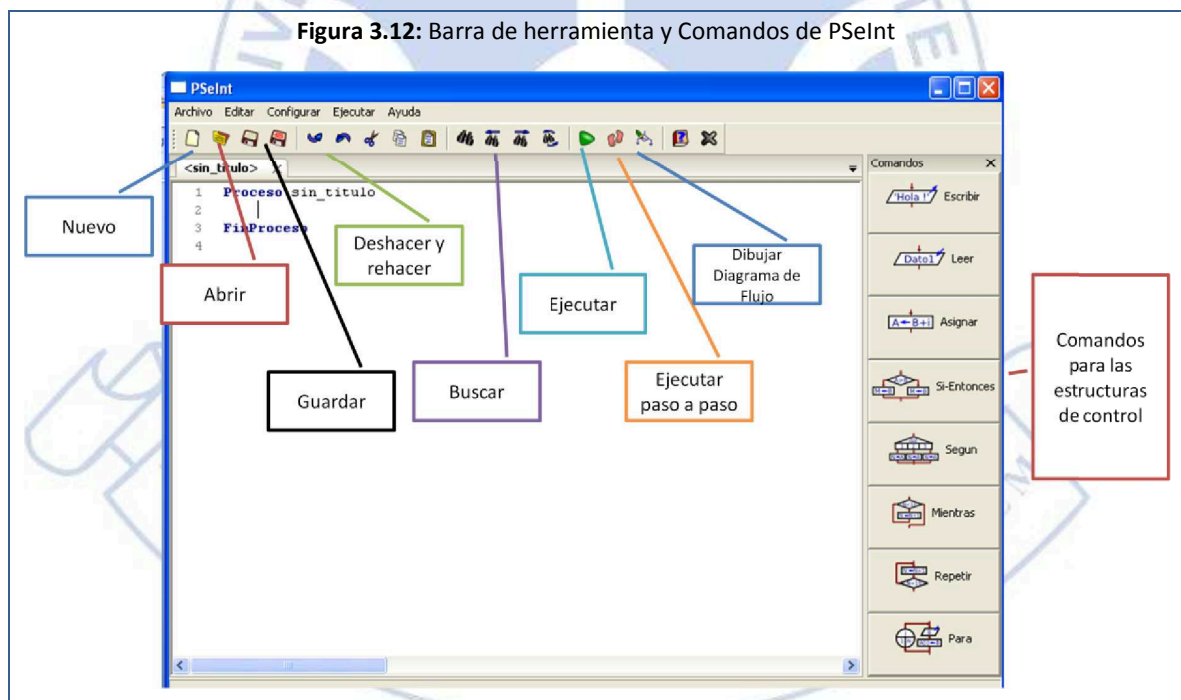
Este software pretende facilitarle al principiante la tarea de escribir algoritmos en este pseudolenguaje presentando un conjunto de ayudas y asistencias, y brindarle además algunas herramientas adicionales que le ayuden a encontrar errores y comprender la lógica de los algoritmos.

Algunas características y funcionalidades de PseInt son:

- Presenta herramientas de edición básicas para escribir algoritmos en pseudocódigo en español
- Permite la edición simultánea de múltiple algoritmos
- Presenta ayudas para la escritura
- Autocompletado
- Ayudas Emergentes
- Plantillas de Comandos
- Coloreado de Sintaxis
- Indentado Inteligente
- Puede ejecutar los algoritmos escritos
- Permite ejecutar el algoritmo paso a paso controlando la velocidad e inspeccionando expresiones
- Puede confeccionar automáticamente la tabla de prueba de escritorio

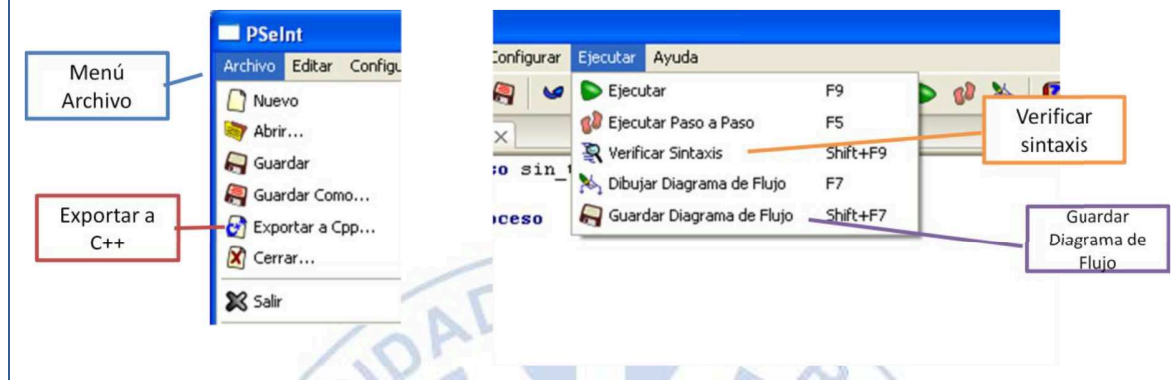
- Determina y marca los errores de sintaxis y en tiempo de ejecución
- Genera diagramas de flujo a partir del algoritmo escrito
- Convierte el algoritmo de pseudocódigo a código C++
- Ofrece un sistema de ayuda integrado acerca del pseudocódigo y el uso del programa (esta última, aún en construcción)
- Incluye un conjunto de ejemplos de diferentes niveles de dificultad
- Es multiplataforma (probado en Microsoft Windows y GNU/Linux)
- Es totalmente libre y gratuito (licencia GPL)

La figura 3.12 muestra una vista general de la barra de herramientas y sus comandos.



En cuanto al menú, la figura 3.13 presenta algunas de las acciones posibles.

Figura 3.13: Algunas de las acciones que permite el menú de PSeInt

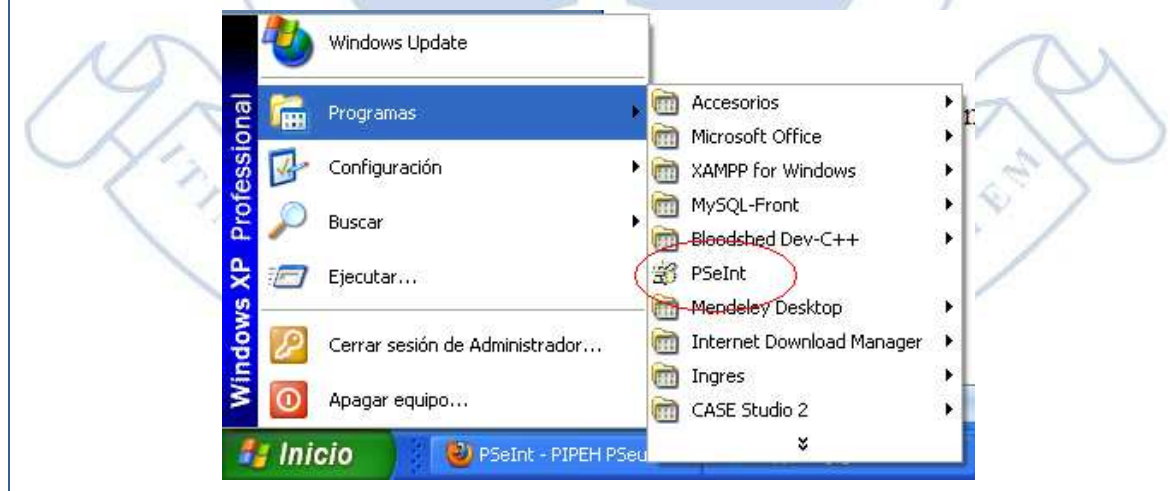


Para instalar PSeInt basta con descargarlo desde la página <http://pseint.sourceforge.net/> y luego hacer doble clic en el siguiente ícono:



Una vez instalado PSeInt puedes ejecutar la herramienta desde el menú Inicio-Programas-PSeInt como se muestra en la figura 3.14.

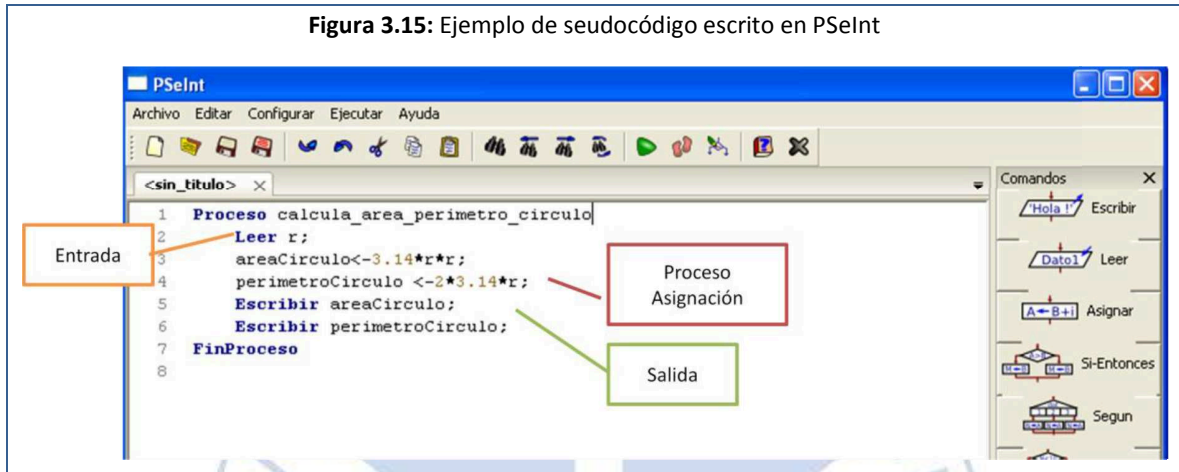
Figura 3.14: Menú para ejecutar la herramienta PSeInt



Para escribir el pseudocódigo diseñado en PSeInt debe hacer clic en los comandos disponibles a la derecha de la ventana de la herramienta, tal y como se muestra en la figura 3.12.

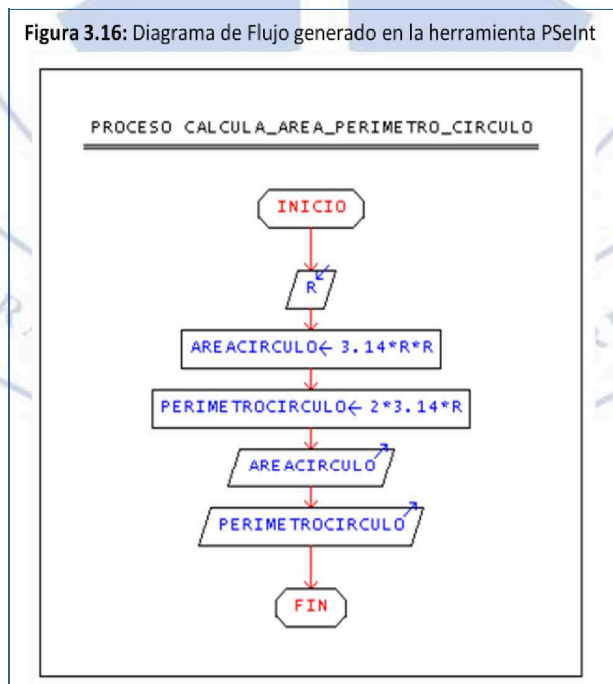
La figura 3.15 muestra un ejemplo de diseño de pseudocódigo para el problema presentado en el ejemplo 3.1.

Figura 3.15: Ejemplo de pseudocódigo escrito en PSeInt



Una vez escrito el pseudocódigo, podemos ejecutarlo haciendo clic en el botón “ejecutar” de la barra de herramientas. También podemos visualizar el diagrama de flujo haciendo clic en el botón “dibujar diagrama de flujo”. La figura 3.16 muestra el diagrama de flujo que resuelve el problema presentado en el ejemplo 3.1

Figura 3.16: Diagrama de Flujo generado en la herramienta PSeInt



3.7 Ejercicios Resueltos

Ejemplo 3.5:

Diseñe un algoritmo que permita calcular la remuneración mensual de un empleado de la empresa “La casa linda”. Se sabe que el sueldo mensual se calcula en base al sueldo base, las horas extras trabajadas (cuando trabaja más de 44 horas a la semana se consideran horas extras) y la cantidad de hijos de dicho empleado. Además se sabe que la empresa paga \$5.000 por hora extra trabajada y \$3.000 por cada hijo que tenga.

Solución

Ya dijimos que el primer paso para diseñar un algoritmo es describir con precisión el problema. Por lo que se requiere que las especificaciones de entrada y salida sean descritas con detalle. Una buena definición del problema, junto con una descripción detallada de las especificaciones de entrada y salida, son los requisitos más importantes para llegar a una solución eficaz.

También dijimos que el análisis del problema exige una lectura previa del problema a fin de obtener una idea general de lo que se solicita. La segunda lectura deberá servir para responder a las preguntas:

¿Qué información debe proporcionar la resolución del problema? (salidas)

¿Qué datos se necesitan para resolver el problema? (entradas)

Análisis

Al responder la primera pregunta tenemos que el problema requiere obtener el sueldo mensual de un empleado de la empresa.

En cuanto a los datos que se requieren para resolver el problema se observa que son tres: sueldo base, horas trabajadas en el mes, y cantidad de hijos.

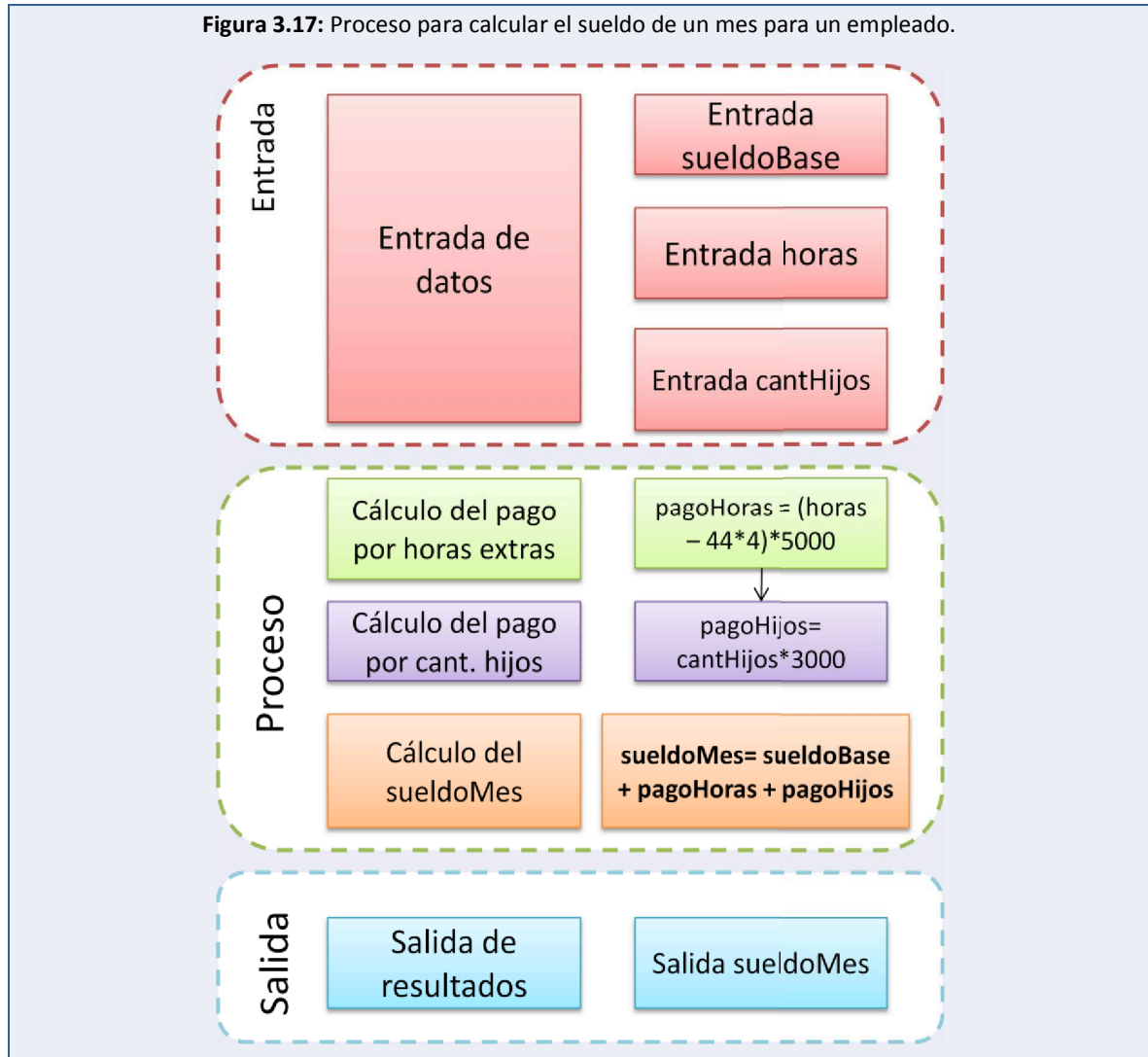
Entradas: sueldo base, horas trabajadas en el mes, y cantidad de hijos (variable sueldoBase, horas, cantHijos)

Salidas: sueldo del mes (variables sueldoMes)

Diseño

El diseño del algoritmo debemos crearlo aplicando la técnica “divides y vencerás”. En este caso para poder calcular el sueldo del mes, debemos calcular por separado el pago por hora extra y el pago por cantidad de hijos. La figura 3.17 muestra por separado cada acción a realizar en las entradas – proceso – salida.

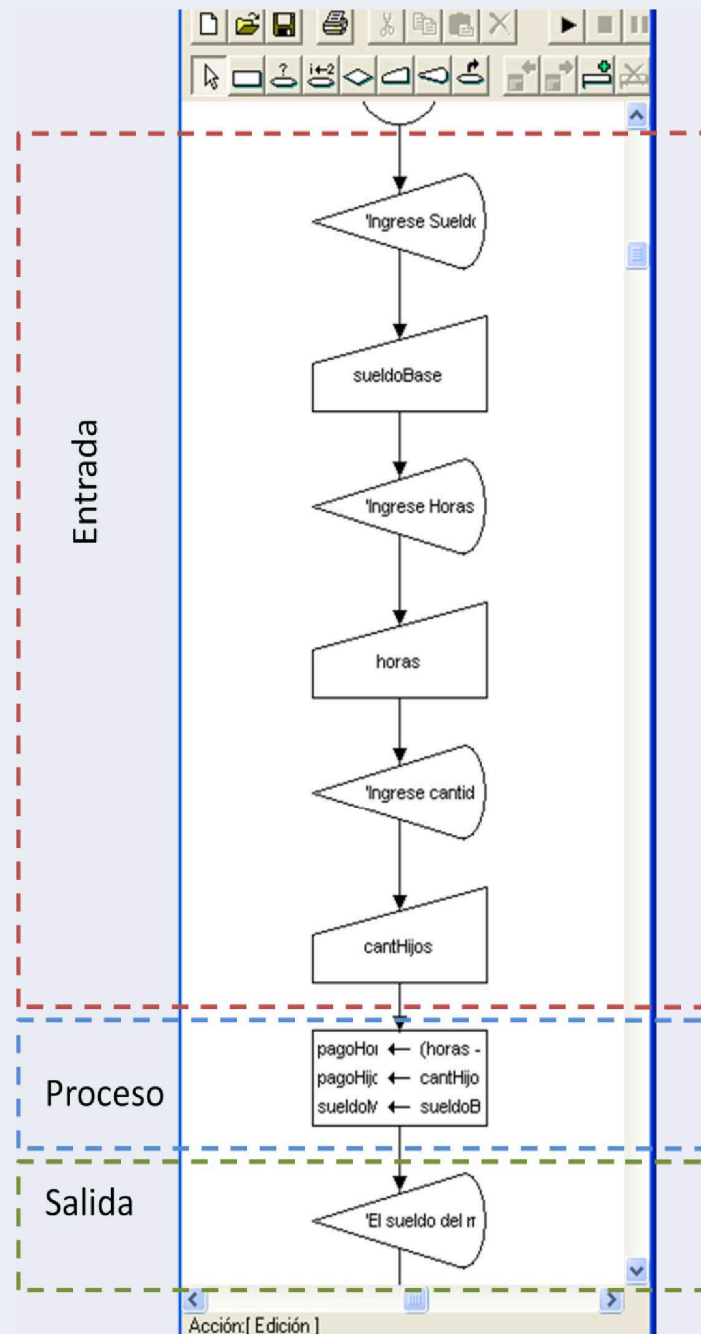
Figura 3.17: Proceso para calcular el sueldo de un mes para un empleado.



Observe que para calcular el pago por horas extras se debe restar a la variable "horas" la cantidad de horas normales trabajadas en la semana, que son 44 por semana considerando 4 semanas de trabajo para el mes. Y el sueldo del mes se calcula sumando el sueldo base con el pago por horas extras y el pago por hijo.

La figura 3.18 muestra el diseño del algoritmo en diagrama de flujo. En la figura resaltamos con cuadros de colores las entradas, proceso y salida.

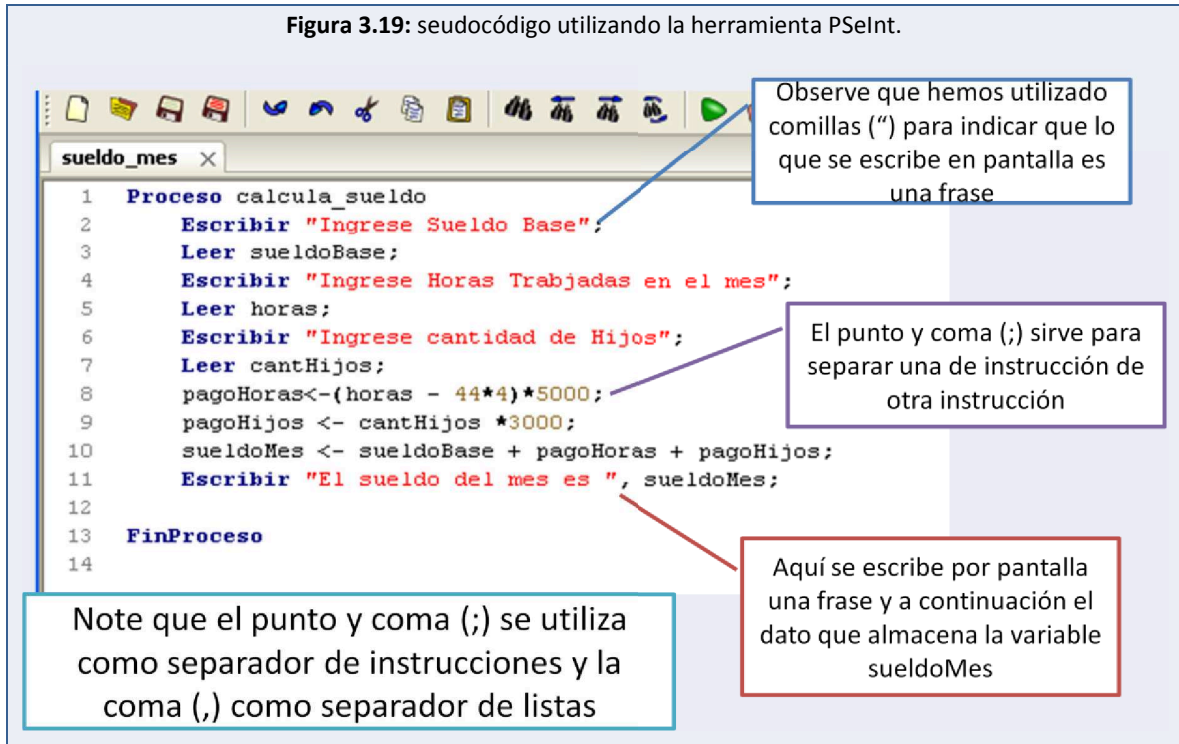
Figura 3.18: diagrama de flujo en la herramienta FreeDFD.



Observe que se han incorporado algunas salidas en la sección “entrada”. Esto se debe a que necesitamos que el usuario del algoritmo comprenda qué valores debe ingresar primero y qué después.

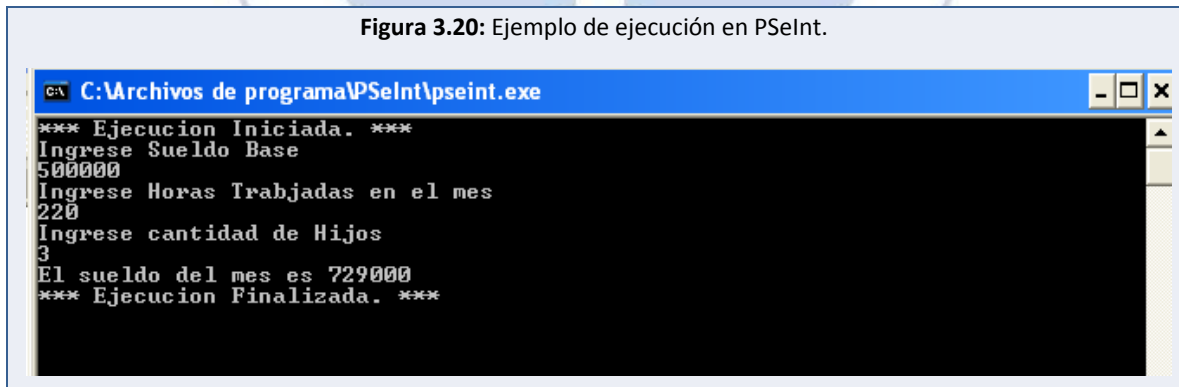
En la figura 3.19 mostramos el diseño del algoritmo utilizando pseudocódigo.

Figura 3.19: pseudocódigo utilizando la herramienta PSeInt.



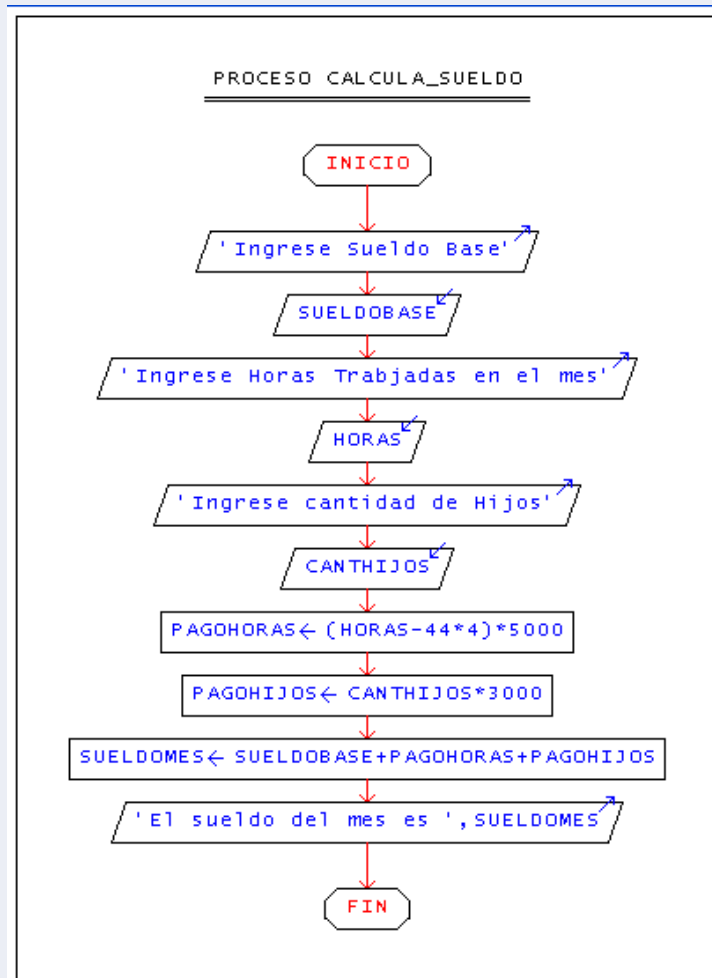
La figura 3.20 muestra un ejemplo de ejecución del algoritmo en PSeInt. Recuerde que para ejecutar un algoritmo debemos hacer clic en el botón "ejecutar" de la barra de herramientas de PSeInt.

Figura 3.20: Ejemplo de ejecución en PSeInt.



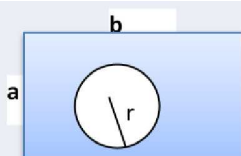
La figura 3.21 muestra el diagrama de flujo que genera PSeInt. Compárelo con el diagrama de la figura 3.18.

Figura 3.21: Diagrama de flujo generado desde PSeInt.



Ejemplo 3.6:

Escriba un algoritmo que permita calcular el área de un rectángulo, extrayéndole el área de un círculo completamente contenido dentro del rectángulo. El área achurada de la figura representa el área que se debe obtener. Los datos a ingresar son el largo de los lados del rectángulo (a y b) y el radio del círculo (r).



SOLUCIÓN

Análisis

En nuestro caso, la resolución del problema debe proporcionar información sobre el valor del área achurada de la figura anterior, por lo que necesitamos calcular el área del rectángulo y el área del círculo. La resta de ambos será el área achurada que se requiere.

En cuanto a las entradas, necesitaremos ingresar los valores de los lados del rectángulo y el radio del círculo ya que π es una constante que está determinada.

Nos faltaría determinar cómo calcular el área del rectángulo y el área del círculo. En este caso sólo debemos recurrir a las ecuaciones 1 y 2 que se presentan a continuación.

(1) $\text{areaRectángulo} = a * b$ (2) $\text{areaCírculo} = \pi * r^2$

Entradas: lados del rectángulo, radio del círculo (variable a, b, y r)

Salidas: área achurada (variables areaAchurada)

Diseño

El diseño del algoritmo debemos crearlo aplicando la técnica “divides y vencerás”. En nuestro caso, el problema del área achurada debemos separarla en tres pequeños problemas:

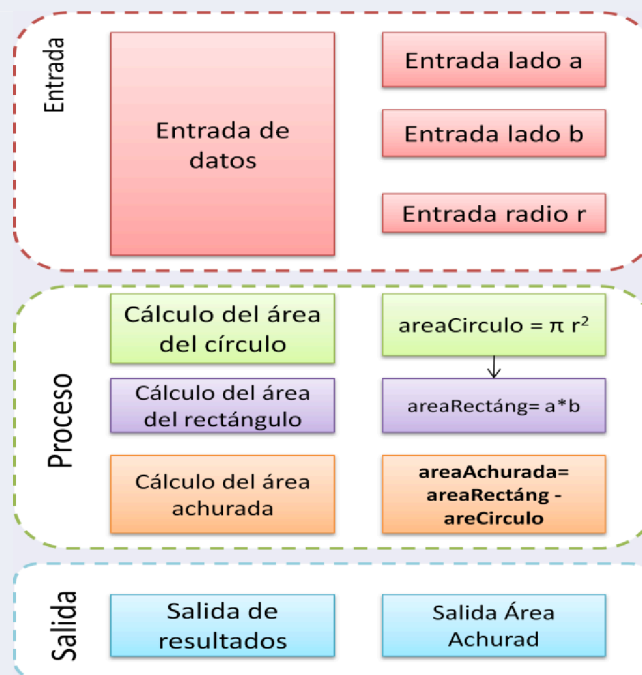
Calcular área del rectángulo

Calcular área del círculo

Calcular área achurada

La figura 3.22 muestra cómo sería el proceso completo para las entradas – proceso – salida

Figura 3.22: Proceso para calcular el área Achurada.



Ahora nos queda representar la solución mediante un diagrama de flujo y pseudocódigo (ver figuras 3.23 y 3.24 respectivamente).

Figura 3.23: Solución utilizando pseudocódigo en la herramienta PSeInt.

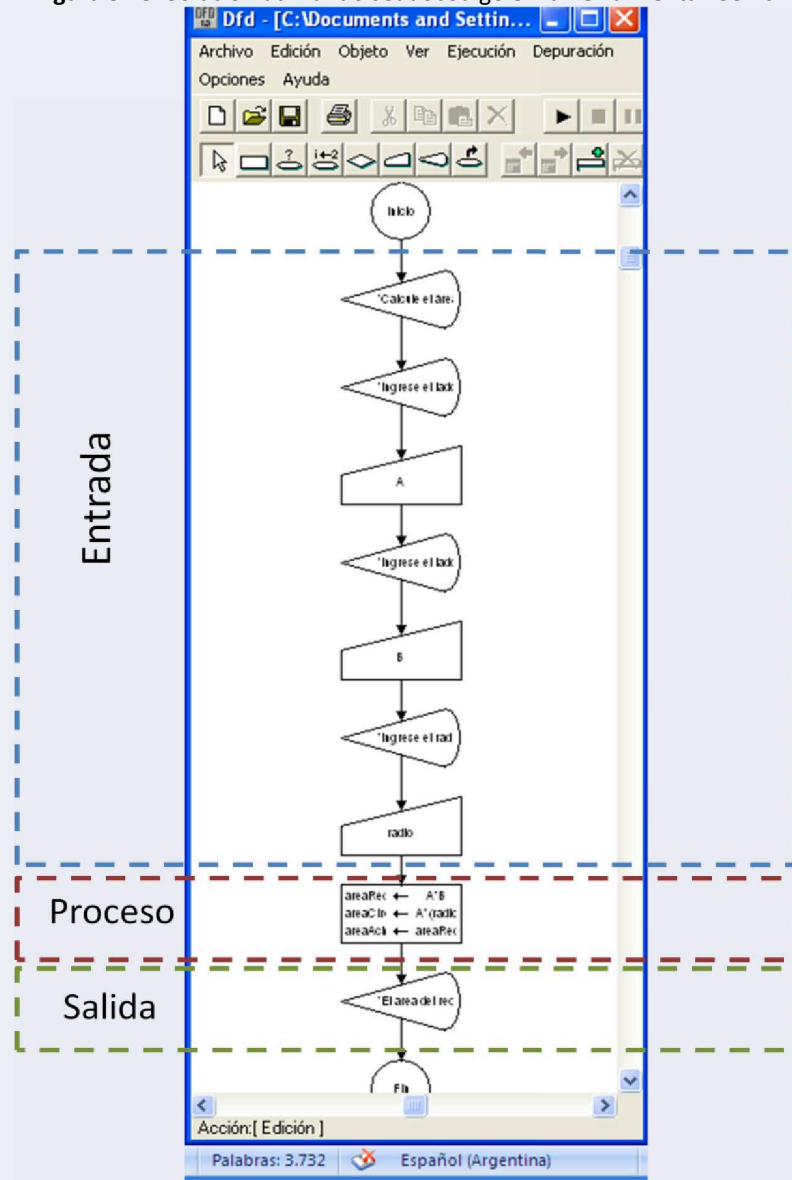
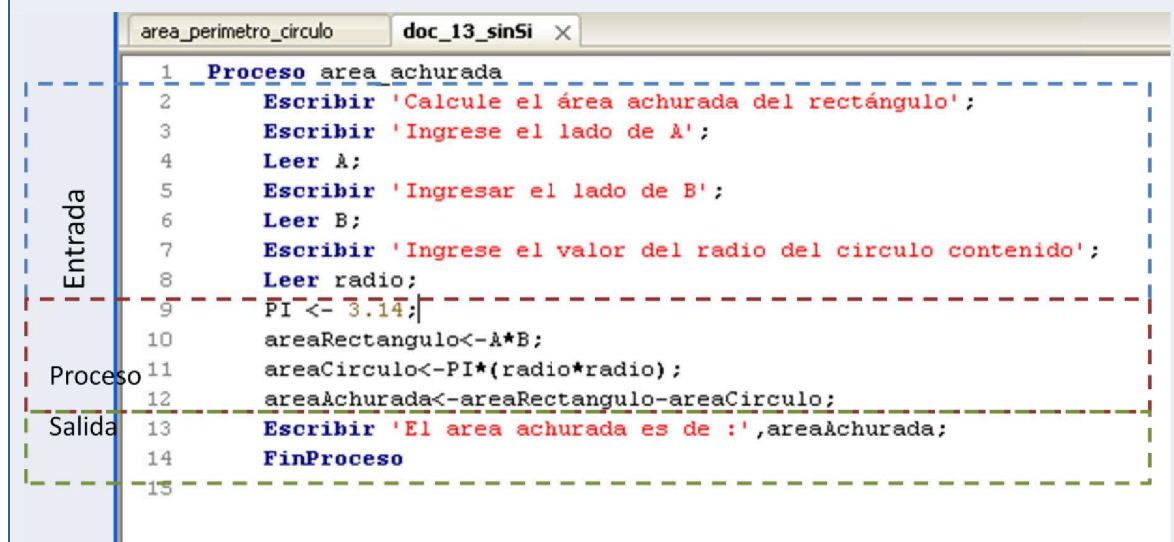


Figura 3.24: Solución utilizando pseudocódigo en la herramienta PSeInt.



Observamos que PI es una constante y no debe ser ingresada por el usuario del algoritmo, sino que debe ser proporcionada por el algoritmo almacenando 3.14 en una zona de memoria.



3.8 Ejercicios Propuestos

Ejercicio 3.1: Diseñe un algoritmo que permita determinar el área y volumen de un cilindro dado su radio(R) y altura (H).

Ejercicio 3.2: Diseñe un algoritmo que permita ingresar dos números y muestre como resultado, la suma y multiplicación de estos.

Ejercicio 3.3: Se pide crear un algoritmo que calcule los días vividos de una persona sabiendo edad.

Ejercicio 3.4: Se solicita diseñar un algoritmo que permita Introducir tres números por teclado, para hallar el promedio de los tres números, la suma y desplegar del primer número el doble, del segundo número el triple y del tercer número el cuadrado.

Ejercicio 3.5: Introducir el sueldo básico de un empleado por teclado, hallar el total ganado considerando que debe introducir el monto por horas extras, considerar un descuento del 20%. Desplegar el sueldo básico, el total del descuento y el total ganado.

Ejercicio 3.6: Un restaurante de comida rápida necesita un sistema que permita leer el número de unidades consumidas de cada alimento ordenado y calcular la cuenta de una orden más IVA.

Lista del menú	Precio (\$)
Completo italiano	690
Completo a la chilena	890
Hamburguesa – queso	990
Churrasco	1100
Cerveza	700
Bebida	500

Ejercicio 3.7: Una persona necesita calcular el número de pulsaciones que debe tener por cada 10 segundos de ejercicio.

La formula es: $\text{num_pulsaciones} = (220 - \text{edad})/10$

Ejercicio 3.8: Se pide desarrollar un algoritmo que permita calcular el nuevo salario de un trabajador si obtuvo un incremento del 10% sobre su salario anterior.

Ejercicio 3.9: Un atleta de lunes a sábados corre la misma ruta y cronometra los tiempos obtenidos, desarrolle un algoritmo que determine el tiempo promedio que el atleta se demora en recorrer la ruta en una semana cualquiera.

Ejercicio 3.10: Se requiere conocer los datos estadísticos de la asignatura de algoritmos, por lo tanto, se necesita un algoritmo que lea el número de reprobados de la signatura, el número de aprobados, notables y sobresalientes y de cómo resultado lo siguiente:

- El porcentaje de alumnos que han aprobado la asignatura.
- El porcentaje de alumnos que han reprobado la asignatura.
- El porcentaje alumnos notables de la asignatura.
- El porcentaje de alumnos sobresalientes de la asignatura.

Ejercicio 3.11: Desarrolle un algoritmo para que un alumno calcule su calificación final en la materia de Algoritmos. La cual se compone de los siguientes porcentajes:

Item	%
Calificaciones parciales	50
Promedio de talleres	25
Promedio de tareas	15
Exposicion del tema de investigación	10

Ejercicio 3.12: En una clínica existen 4 áreas: Kinesiología, Ginecología, Pediatría, y Traumatología. El presupuesto anual de la clínica se reparte según se especifica en la siguiente tabla:

Area	%
<i>Kinesiología</i>	20
<i>Ginecología</i>	25
<i>Traumatología</i>	30
<i>Pediatría</i>	25

Desarrolle un algoritmo que permita obtener la cantidad de dinero que recibirá cada área, para cualquier monto presupuestado.

Ejercicio 3.13: Una compraventa de automóviles necesita un sistema que permita informar a sus clientes el detalle del costo de un vehículo al comprarlo, es decir un comprobante que indique el costo total del vehículo y costo adicional que debe pagar el comprador por distintos conceptos.

Para ello se debe tener en cuenta que el costo de un vehículo nuevo para un comprador es la suma total del costo del vehículo, del porcentaje de la ganancia del vendedor, de los impuestos locales, e impuestos estatales aplicables (sobre el precio de venta).

- Suponer una ganancia del vendedor del 8% para cada vehículo
- Impuesto local del 3%
- Impuesto estatal del 3%

Ejercicio 3.14: Tres amigos reúnen sus ahorros para comprar un vehículo. Cada una de ellos aporta una cantidad distinta de dinero. Desarrolle el algoritmo que determine el porcentaje que cada amigo aporta con respecto a la cantidad total reunida.

3.9 Comentarios Finales

Inicialmente se introduce al lector el concepto de algoritmo, y como abordar esquemáticamente la resolución de un problema describiendo las tres etapas:

(1) *Análisis del problema*

(2) *Diseño o desarrollo de algoritmos*

(3) *Resolución del algoritmo en el computador (tema abordado en los siguientes capítulos)*

Con respecto al *Análisis del problema*, se recuerda que se debe tener presente siempre la:

- (a) *Definición del problema*
- (b) *Especificación de entrada*
- (c) *Especificación de salida*

Estos como los requisitos más importantes para llegar a una solución eficaz.

Con respecto al *Diseño o desarrollo de algoritmos* se destaca que el diseño de un algoritmo consiste en llevar a cabo tres etapas:

- (a) *Diseño descendente*
- (b) *Refinamiento por pasos*
- (c) *Herramientas de programación (diagrama de flujos y pseudocódigo).*

En esta sección también se describe cómo debe realizarse el diseño de un Algoritmo, la cual consiste en una secuencia de pasos descrita en lenguaje natural y siempre considerando las siguientes características:

- Deben estar seguidas de alguna *secuencia definida* de pasos hasta que se obtenga un resultado coherente.
- Sólo puede ejecutarse un paso (instrucción) a la vez.

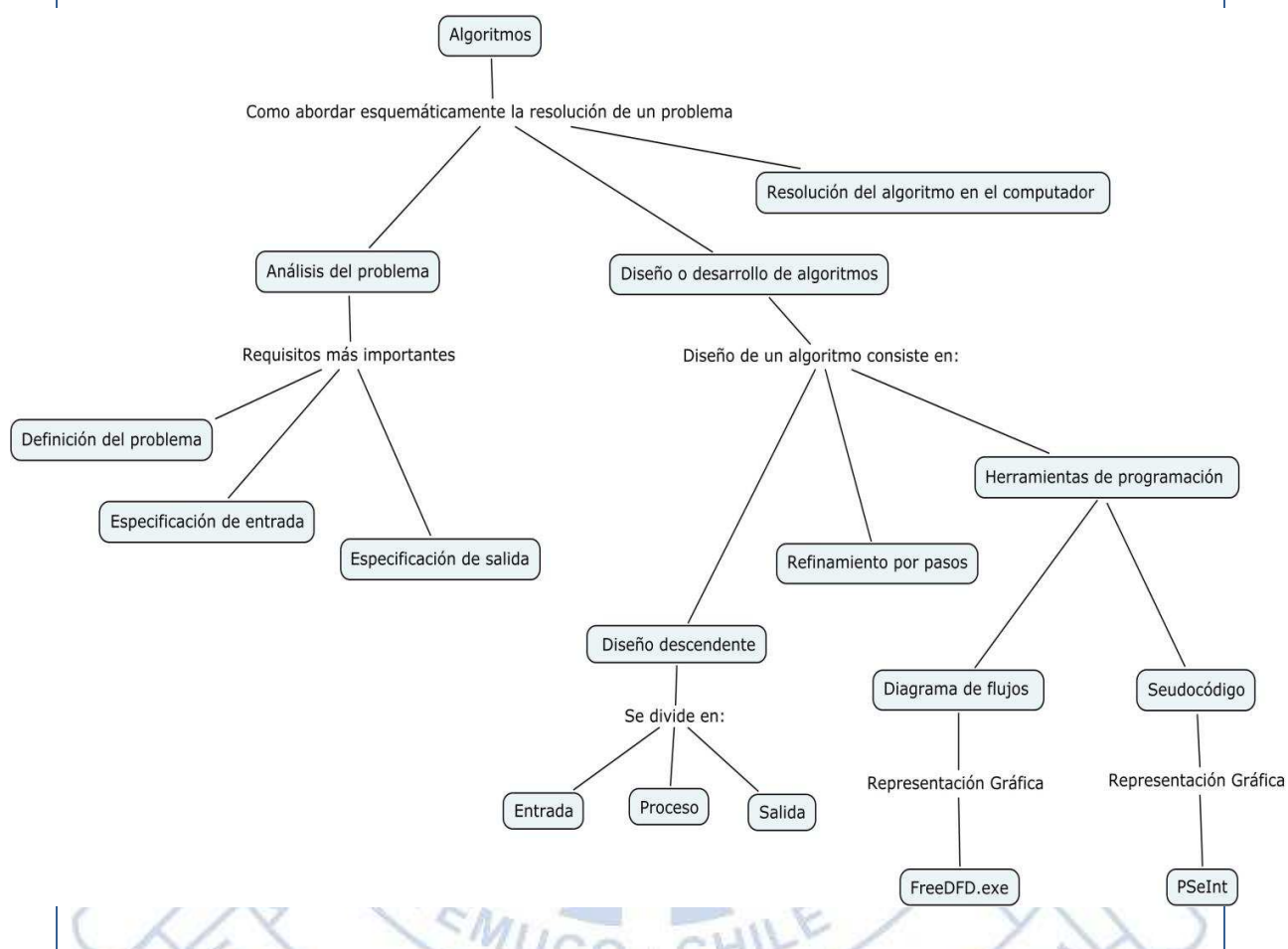
La *Representación de un Algoritmo*, la realizamos por medio de: *el diagrama de flujo y el pseudocódigo*.

Las *Herramientas para la Representación y Ejecución de Algoritmos*, que revisamos son:

- FreeDFD.exe (se utiliza para diseñar y ejecutar algoritmos mediante diagrama de flujo)
- PSeInt (se utiliza para diseñar y ejecutar pseudocódigo)

La siguiente figura muestra un resumen utilizando un mapa conceptual.

Figura 3.25: Mapa conceptual de resumen



3.10 Referencias

Luis Joyanes (2000): **“Fundamentos de Programación, algoritmos y estructuras de datos”**. Mc-Graw Hill.
ISBN 84-481-0603-2