# ALGORITMOS RESUELTOS CON PYTHON

PSEDOCÓDIGOS
DIAGRAMAS DE FLUJOS

python

ENRIQUE EDGARDO CONDOR TINOCO
UNIVERSIDAD NACIONAL JOSÉ MARÍA ARGUEDAS

MARCO ANTONIO DE LA CRUZ ROCCA
UNIVERSIDAD NACIONAL DANIEL ALCIDES CARRIÓN

Eidec
EDITORIAL

# Algoritmos resueltos con Python

## Colección:
## Tecnologías de la información y la comunicación (TIC)

**Autores:**

**Enrique Edgardo Condor Tinoco**
Universidad Nacional José María Arguedas
enricoti@unajma.edu.pe

**Marco Antonio De la cruz Rocca**
Universidad Nacional Daniel Alcides Carrión
mcruzr@undac.edu.pe

**Volumen No. 1**

Primera Edición 2020

**Editorial EIDEC**

**NIT**: 900583173-1

**Sello editorial:** 978-958-53018

**ISBN:** 978-958-53018-2-5

**DOI:** https://doi.org/10.34893/6kbn-5a63

**Fecha Publicación:** 2020-10-28

comiteeditorial@editorialeidec.com
**www.editorialeidec.com**

Colombia

# Prefacio

El ser humano todos los días enfrenta distintas situaciones en las actividades que hace, estas las tiene que resolverlos basándose en sus conocimientos y experiencias adquiridas, pero, los realiza usando alguna metodología o una serie de pasos con la finalidad de obtener una solución que le convenga. Los hombres en su quehacer diario tienen acciones rutinarias como prepararse para ir a estudiar o a trabajar, tomar el taxi o el bus, atender las tareas diarias del trabajo, preparar los alimentos del día, llevar a los niños a la escuela, responder los mensajes de los correos electrónicos; todas ellas siguen una secuencia y un propósito. Al conjunto de actividades ordenadas con un objetivo claro se le llama algoritmos.

El presente texto ha sido diseñado para tratar los temas relacionados a los algoritmos y provee conocimiento teórico y práctico de la metodología para la programación de forma estructurada mediante el empleo de diagramas de flujo y pseudocódigos.

Debido a que la programación involucra un grado de complejidad, aquí se introduce la herramienta algorítmica para

resolver un problema ingenieril en varios subsistemas, aplicando la técnica divide y vencerás.

El libro incluye cinco capítulos, distribuidos en orden de aprendizaje y dificultad, que permitirá al lector sumergirse con facilidad y captar de manera simple la resolución de problemas mediante algoritmos que se puedan traducir a un lenguaje de programación.

Los ejercicios se encuentran en pseudocódigos, diagramas de flujo y traducidos al lenguaje de programación Python. Se utilizó este lenguaje de programación porque permite expresar algoritmos de forma casi directa y es adecuado para la enseñanza de la programación.

En el libro se tiene ejercicios prácticos, en el que podrán percibir que programar no es difícil, solo se requiere de paciencia y un poco de esfuerzo ya que con mucha practica obtendrás la habilidad de programar. Esperamos que esta obra cumpla sus objetivos y satisfaga sus inquietudes querido lector.

Enrique Edgardo Condor Tinoco

Marco Antonio De la Cruz Rocca

# CONTENIDO

Prefacio .....5
Índice de Tablas .....11
Índice de Figuras .....12
Introducción .....13
1. Lenguaje de Programación Python .....15
1.1 Python ..... 15
1.2 Características del lenguaje ..... 15
1.3 Instalación de Python: ..... 18
1.4 ¿Por qué Python? ..... 20
1.5 Visual Studio Code ..... 21
1.6 Integración de Python con VS Code ..... 27
1.7 Ejecutar código Python ..... 28
1.8 Ejecutar código Python en VS Code ..... 30
1.9 ¿Por qué VS Code? ..... 33
2. Lenguaje y Algoritmos .....34
2.1 ¿Qué es un algoritmo? ..... 35
a. Pseudocódigos ..... 36
b. Diagramas ..... 36
a. Datos numéricos ..... 37
b. Datos alfanuméricos ..... 38
c. Datos lógicos ..... 38

a. Identificadores ..... 39

b. Constantes ..... 39

c. Variables ..... 39

a. Operaciones aritméticas ..... 39

b. Operaciones relacionales ..... 42

c. Operaciones lógicas ..... 43

3. Instrucciones Básicas en un Algoritmo ..... 49

3.1 Leer ..... 49

3.2 Asignar ..... 51

3.3 Escribir ..... 51

3.4 Ejercicios del tema ..... 53

3.5 Pseudocódigo y sus partes ..... 67

3.6 Diagramas de flujo símbolos usados ..... 69

3.7 Ejercicios complementarios ..... 80

4. Estructuras Selectivas ..... 102

5. Estructuras Repetitivas ..... 148

5.1 Desde o Para (For): ..... 148

5.2 Mientras (While) ..... 149

5.3 Repetir (Repit) ..... 150

5.4 Ejercicios del Tema ..... 153

5.5 Ejercicios Complementarios ..... 166

6. Vectores y Matrices Funcionales ..... 203

6.1 Vectores y Matrices .....203

6.2 Arrays Unidimensionales: Vectores.....203

6.3 Arrays Multidimensionales o Matrices .....204

6.4 Ejercicios del Tema.....208

6.5 Ejercicios Complementarios.....226

7. Glosario .....251

8. Bibliografía .....256

# Índice de Tablas

Tabla 1 Atajos de edición básica ..... 23
Tabla 2 Atajos de búsqueda y remplazo ..... 23
Tabla 3 Atajos de mutiCursor y Selección..... 23
Tabla 4 Atajos de administración de archivos..... 24
Tabla 5 Prioridad de operadores aritméticos..... 40
Tabla 6 Operaciones Aritméticos en Python..... 42
Tabla 7 Prioridad de operadores relacionales..... 42
Tabla 8 Operaciones relacionales en Python..... 43
Tabla 9 Prioridad de operadores lógicos. ..... 44
Tabla 10 Operadores lógicos en Python..... 44
Tabla 11 Símbolos usados frecuentemente en diagramas de flujo ..... 69

# Índice de Figuras

Figura 1. Pruebas en el intérprete por la línea de comando de Python. ..... 16
Figura 2. Accediendo a la línea de comando de Python mediante CMD ... 17
Figura 3. Características opcionales para instalar en Python. ..... 19
Figura 4. Opciones avanzadas de la instalación de Python. ..... 20
Figura 5. Comprobar la instalación de Python en el Sistema Operativo. ... 20
Figura 6. Abriendo la terminal integrada de VS Code..... 25
Figura 7. Configuración del atajo de teclas de la terminal integrada. ..... 26
Figura 8. Presentación de Marketplace. ..... 27
Figura 9. Instalación del plugin Python. ..... 27
Figura 10. Primer programa en Python. ..... 28
Figura 11. Guardando el archivo con el nombre: saludo.py ..... 29
Figura 12. Pasos para ejecutar un programa en Python usando CMD ..... 29
Figura 13. Para abrir una terminal en la carpeta donde está ubicado ..... 30
Figura 14. Abrir un directorio en VS Code, usando Open Folder… ..... 31
Figura 15. Vista posterior al abrir un directorio con: Open Folder… ..... 31
Figura 16. Ejecutando un programa de Python con el botón Ejecutar..... 32
Figura 17. Ejecutando un Python de la segunda manera. ..... 33
Figura 18. Pensando en una idea ..... 35
Figura 19. Secciones de un algoritmo..... 35
Figura 20. Algoritmo mediante Pseudocódigo ..... 36
Figura 21. Diagrama de flujo (izq.) y Carta N-S (der.)..... 37
Figura 22. Diagrama de flujo de una estructura condicional simple .....103
Figura 23. Diagrama de flujo de una estructura condicional doble .....107
Figura 24. Diagrama de flujo de una estructura condicional múltiple.....111

# Introducción

El presente libro está dirigido a estudiantes y personas que están relacionados en el proceso de aprendizaje con las especialidades de ingeniería de sistemas, informática, computación y pretende aportar las herramientas básicas y luego es aplicado a la programación para la resolución de problemas.

En el libro se tiene una secuencia de temas que permite abordar la programación desde un punto de vista más creativo. Se aborda distintos ejercicios según el capitulo en estudio, todos los ejercicios se encuentran resueltos por un pseudocodigo, diagrama de flujo y programado en Python.

El libro se encuetra dividido en los siguientes seis capítulos: 1). Lenguaje de programación Python, en el que se indica de que se trata este lenguaje, sus características, ventajas y desventajas, su proceso de instalación. Se ha considerado mencionar al inicio del libro porqué se usa este lenguaje en el desarrollo de todos los ejercicios que se encuentran planteados. En esta parte, también se detalla sobre el entorno de desarrollo integrado (IDE) Visual Studio Code o VS, lugar donde se han escrito todos los programas hechos en Python, se aborda desde lo que es VS hasta su integración con Python. 2). Lenguaje y algoritmos, en este capítulo se inicia con la secuencia del aprendizaje al tema de algoritmos en el que se trata los temas de lo que es un algoritmo, representación de un algoritmo, que es un dato, identificadores y operadores, todos estos sirven para iniciar a realizar algoritmos

mediante un pseudocodigo y un diagrama de flujo. 3). Instrucciones básicas de un algoritmo, en esta parte se complementa al capítulo 2 con las instrucciones que realiza todo algoritmo, de leer datos y después de un proceso obtener un resultado. 4). Estructuras selectivas, se analiza las estructuras simples, dobles y multiples para el desarrollo de algoritmos. 5). Estructuras repetitivas, en este capitulo se aborda las instrucciones desde, mientras, repetir, los que nos permiten realizar bucles en nuestros programas. 6). Vectores y matrices funcionales, se estudia los temas de arreglos unidimensionales y multidimensionales.

Con el desarrollo de todos los capítulos se habrá logrado comprender la utilidad de los algoritmos y su uso correcto de los pseudocodigos y diagramas de flujos, además se habrá vizualizado la facilidad con que se program en Python. En cada capítulo se realiza ejercicios, los que se denomina ejercicios complementarios, estos con la finalidad de reforzar cada tema.

De todos los temas tratados, a los estudiantes he de indicarles que es preciso adoptar una actitud crítica en la solución de los problemas propuestos y hasta de plantear otras soluciones al problema.

# 1

# Lenguaje de Programación Python

## 1.1 Python

Python es un lenguaje de programación de alto nivel, es decir las sintaxis que se suele usar es fácil de leer para un ser humano, a comparación de otros lenguajes como java y c++, ya que la filosofía del lenguaje es proporcionar una sintaxis muy limpia y que beneficie con código leíble. (Challenger-Pérez et al., 2014)

Python es una de las herramientas tecnológicas que en los últimos años se ha hecho muy popular, gracias a varias razones como:

- La cantidad de librerías que contiene.
- La rapidez con la que se crean los programas.
- Es multiplataforma (se puede desarrollar y ejecutar programas en, Linux, Windows, Mac, Android y otros)
- Python es gratuito, incluso para propósitos empresariales.

## 1.2 Características del lenguaje

a) **Propósito general**

Con Python se pueden construir todo tipo de programas. Se podrían crear aplicaciones de escritorio o web, usarlo para automatizar tareas en servidores, o quizá usarlo como una gran calculadora. Lo más interesante de su propósito se ha visto con aplicación a la Inteligencia artificial, por su capacidad para procesamiento de dato**.**

b) **Multiplataforma**

Otra de sus ventajas es, que una aplicación creada en Python puede ejecutarse en distintas plataformas como Unix, Windows, Mac o incluso Android.

c) **Interpretado**

Al ser una ventaja la capacidad de crear programas rápidamente, una desventaja que posee Python es que el un programa debe ser interpretado, lo que hace que sea más lento que un programa compilado (traducido a binario, 1's y 0's) que se comunica directamente con microprocesador, ya que es el intérprete quién se comunica con el microprocesador.

d) **Interactivo**

Posterior a su instalación Python dispone de un intérprete por la línea de comandos en el que se pueden introducir sentencias. La ventaja de tener esta terminal es que uno puede ir probando comandos de Python que no ha entendido bien y hacer pruebas con él. O probar que hacen algunos trozos de código del programa que se esté desarrollando.

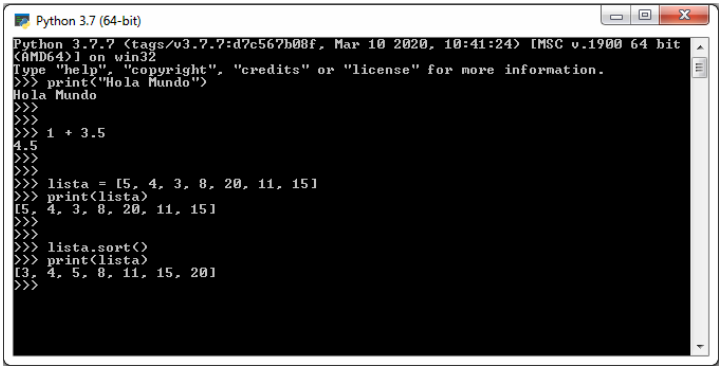



**Figura 1.** Pruebas en el intérprete por la línea de comando de Python.

Otro modo de acceder a la línea de comandos es utilizando cualquier terminal (en caso de Windows usando el CMD) y digitar el comando Python seguido de la tecla enter.

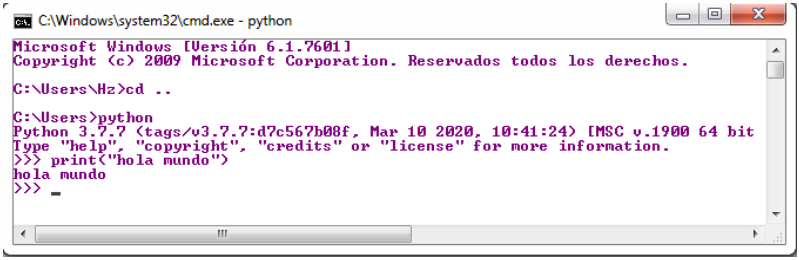



**Figura 2.** Accediendo a la línea de comando de Python mediante CMD

e) **Orientado a Objetos**

Python está diseñado para soportar el paradigma de la programación orientado a objetos (POO), donde todos los módulos se tratan como entidades que tienen atributos, y comportamiento. Lo cual, en muchos casos, ayuda a crear programas de una manera sencilla, además de que los componentes son reutilizables.

f) **Funciones y librerías**

El lenguaje trae consigo diversas funciones incluidas en la carpeta de instalación; muchas de ellas para la interacción con Cadenas de texto, números, listas, ficheros, sistema operativo, etc.

Además, en el mercado, existen muchas librerías de terceros que se pueden importar los cuales podrían facilitar a la creación de interfaces gráficas como puede ser Tkinter. Si se desea crear videojuegos, se podría importar alguna otra librería como Pygame. O si se trata de hacer operaciones con grandes cantidades de datos se podría importar NumPy. Así se pueden encontrar infinidad de librerías orientadas a diferentes campos**.**

g) **Sintaxis clara**

Python tiene una sintaxis muy visual (fácil de leer), gracias a su notación indentada (Agregar márgenes a la izquierda cuando estos los requieran para identificar estructura de datos)

los cuales deben implementarse obligatoriamente si se quiere evitar errores y que el programa ejecute satisfactoriamente.

En varios lenguajes, para separar trozos de código, se usan caracteres como las llaves o las palabras reservadas begin y end. En ocasiones muchos programadores novicios obvian la indentación generando así caos en el programa que están escribiendo, lo cual dificulta la lectura para los humanos. Otra desventaja de no indentar el código es que si ocurre un error en el programa será mucho más difícil encontrarlo, por no decir que en algunos casos será imposible. La buena indentación de código hace que podamos entender y mantener el código en futuros porvenires.

Para realizar la indentación se debe hacer uso de la tecla tabulador, colocando un margen a la izquierda del trozo de código que iría dentro de una función o un bucle.

Otra de las ventajas es que los programadores tengan una estandarización de reglas para escribir un código claro y leíble**.**

## 1.3 Instalación de Python:

Python se puede descargar de manera gratuita de su página oficial https://www.python.org/downloads/, para los ejercicios de este libro se está usando Python 3.

Sobre el instalador haga doble clic. En la ventana que salga es recomendable dejar las configuraciones por defecto que sugiere Python:

- Documentation: permite instalar la documentación al cual se puede acceder sin necesidad de internet.
- Pip: Es una herramienta que permite integrar paquetes y librería de terceros.
- Tcl/tk: una librería que permite crear interfaces gráficas para Python.

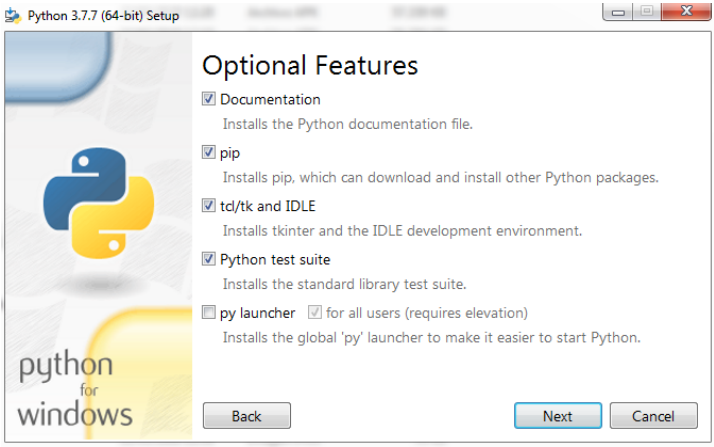



***Figura 3.*** Características opcionales para instalar en Python.

Al momento de instalar Python, uno de los factores más importantes para que se puede ejecutar en cualquier lugar del Sistema Operativo es la configuración de variables de entorno: Afortunadamente Python trae una opción para realizar esta configuración desde la instalación. Por ello es imperativo marcar la casilla: Add Python to environment variables (Agrega a Python a las variables de entorno). Teniendo todo ello en cuenta solo hace falta darle clic a Instalar.

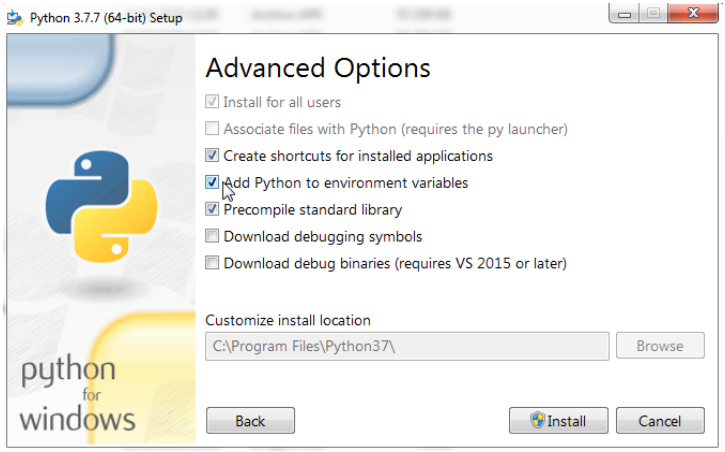



***Figura 4.*** Opciones avanzadas de la instalación de Python.

Para comprobar que todo ha ido correcto en una terminal escriba: Python - -version.

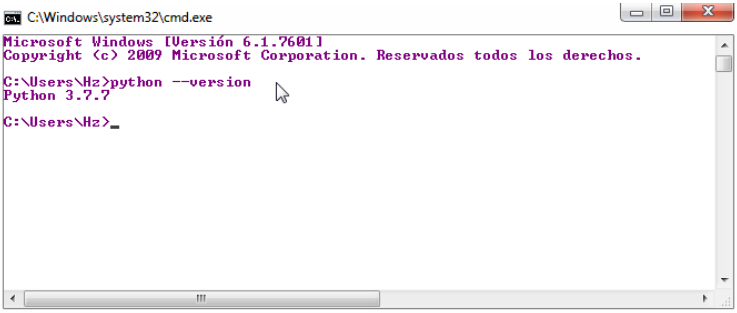



***Figura 5.*** Comprobar la instalación de Python en el Sistema Operativo.

## 1.4 ¿Por qué Python?

Python está en movimiento y en continuo desarrollo apoyando en distinticos campos de la ciencia, matemática, estadística, inteligencia artificial (Chazallet, 2016)

Es multiplataforma, lo que significa que no se debe reescribir todo el código para migrar de una plataforma a otra.

Si uno tiene dudas acerca de los comandos o instrucciones de Python tiene a su disposición una infinidad de blog en internet, muchos de ellos en español.

Existen muchas librerías a disposición para todo tipo de campos de estudios, Python es una gran herramienta de apoyo para la ciencia.

## 1.5 Visual Studio Code

Visual Studio Code o VS Code (como se le conoce) es un proyecto de Microsoft de código abierto que además es multiplataforma por lo que se podrá ejecutar en Windows, Linux, o macOs.

VS Code es un editor de código moderno y muy poderoso gracias a sus muchas funcionalidades prácticas al momento de trabajar con código. Algunas de las ventajas que tiene son:

**Soporte para depuración de errores**, significa que este entorno de desarrollo proporciona herramientas para ir probando el código buscar errores y si fuera necesario se podría hacer seguimiento de la ejecución línea a línea.

**a) Resaltado de sintaxis**.

Como en todo editor de texto, no podría faltar, el resaltado de sintaxis que es una excelente ayudada a la hora de identificar elementos del código; como pueden ser las palabras claves del lenguaje en el que se esté trabajando, los comentarios, las funciones y clases.

Tiene soporte de resaltado de sintaxis para casi todos los lenguajes como puede ser: Bash, C, C++, Closure, C#, Go, Java, Lua, Makefile, Markdown, Objective-C, Perl, PHP, PowerShell, Python, R, Ruby, SQL, Visual Basic, XML, HTML, CSS, Javascript, JSON, Less, Sass, Typescript, etc.

En caso de que algún resaltado de sintaxis aún no se encuentre instalado puede hacer uso del Marketplace (que también ya viene integrado), para instalarlo.

**b) Control integrado de Git.**

Una de las herramientas que no podría faltar, y que es una gran ayuda a nivel profesional, es el control de versiones Git. Git es otra herramienta potente que todo programador debe aprender para le gestión, orden de las versiones de los proyectos que se van creando. Otra de sus ventajas es que Git ayuda a organizar el trabajo en equipos y multi-equipos gracias a su filosofía de ramas, por lo que cada equipo estaría trabando en una rama aislada; una vez hayan finalizado su respectiva tarea puede unir cada avance a la rama principal.

**c) Autocompletado inteligente de código.**

Otra de las ayudas que no debe faltar en todo editor de texto es: el autocompletado inteligente, que sugiere lo que quizá se quiere escribir; esto significa que puede que no escribirse toda la palabra siempre, eso ahorrar tiempo y da la posibilidad de terminar el programa mucho más rápido.

**d) Atajos de teclado.**

Otro de los aspectos a tener en cuenta tener atajos de teclado, lo que permite hacer tareas muy rápido y sin necesidad de usar el mouse. VS Code tiene una infinidad de atajos de teclado, ellos pueden ser agrupados en distintos grupos, algunos de los más importantes son:

**Atajos de edición básica**. Son aquellos que permiten agilizar tareas al momento de escribiendo código.

**Atajos de navegación.** Aquellos facilitan el desplazarse por toda la página; su potencial resalta en documentos muy largos, donde moverse de una a otra función es tediosa.

**Atajos de búsqueda y remplazo**. Estos atajos permiten encontrar palabras claves que se quiere buscar, y puede moverse a través de ellas; en algunas ocasiones quizá la palabra haya sido mal escrita o se desea cambiar por una mejor nomenclatura, para ellos se dispone de la opción de remplazar y remplazar todo.

**Atajos de administrador de archivos.** Son aquellos atajos de teclado permiten abrir, crear, cerrar archivos, así como crear nuevas carpetas. Prácticamente permiten moverse por un árbol de directorios dentro del mismo editor, evitando la necesidad de abrir el Administrador de archivos del sistema operativo.

**Tabla 1**
*Atajos de edición básica*

| **Atajos de edición básica** | |
|---|---|
| Ctrl+X | Cortar line (sin seleccionar) |
| Ctrl+C | Copiar line (sin seleccionar) |
| Alt+Up Arrow/Down Arrow | Mover línea Arriba/abajo |
| Shift+Alt+Down Arrow/Up Arrow | Copia línea Arriba/abajo |
| Ctrl+Shift+K | Eliminar línea |
| Ctrl+Enter | Insertar línea abajo |
| Ctrl+Shift+Enter | Insertar línea arriba |
| Home | Ir al inicio línea |
| End | Ir al final de la línea |
| Ctrl+Home | Ir al inicio del archive |
| Ctrl+End | Ir al final del archivo |
| Ctrl+KthenCtrl+C | Comentar línea |
| Ctrl+KthenCtrl+U | Remover comentario de línea |

**Tabla 2**
*Atajos de búsqueda y remplazo*

| **Búsqueda y remplazo** | |
|---|---|
| Ctrl+F | Buscar |
| Ctrl+H | Remplazar |
| F3/Shift+F3 | Buscar siguiente/anterior |

**Tabla 3**
*Atajos de mutiCursor y Selección*

| **MutiCursor y Selección** |
|---|

| | |
|---|---|
| Alt+Click | Insertar cursor |
| Ctrl+Shift+L | Seleccionar todas las ocurrencias de la selección actual |
| Ctrl+F2 | Select all occurrences of current word |
| Ctrl+L | Seleccionar línea |

**Tabla 4**
*Atajos de administración de archivos*

| **Atajos de administración de archivos** | |
|---|---|
| Ctrl+N | Nuevo archivo |
| Ctrl+O | Abrir archivo |
| Ctrl+S | Guardar |
| Ctrl+Shift+S | Guardar como |
| Ctrl+KthenS | Guardar todos los archivos |
| Ctrl+F4 / Ctrl + W | Cerrar archivo |
| Ctrl+KthenCtrl+W | Cerrar todos los archivos |
| Ctrl+Shift+T | Reabrir archivo |

**e) Soporte de multicursor:**

Esta característica es resaltante porque permite editar varias partes del documento al mismo tiempo.

**f) Terminal Integrada:**

En muchos lenguajes, y en especial en Python, se inicia haciendo aplicaciones de consola lo que significa que para ejecutarlos se necesita de una terminal. El proceso que normalmente se seguiría sería: salir del editor, ir a la ubicación del archivo, abrir una terminal y ejecutar el programa.

La ventaja de tener una terminal integrada es que no hace falta salir del editor para ejecutar el programa en construcción; además de que se puede invocar con una combinación de teclado, lo cual puede varías dependiendo de la configuración

del idioma del teclado; si la configuración está en español la combinación será **Ctrl + Ñ.**

**Nota**: la terminal se abrirá en la path (dirección) de la carpeta abierta en el editor de texto.

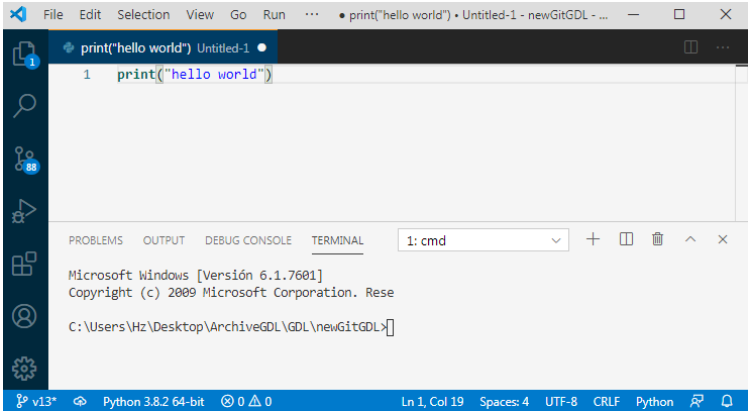



***Figura 6.*** Abriendo la terminal integrada de VS Code

En caso de que no se mostrará habrá que ver cuál es la configuración de la terminal integrada, para ello debe presionar: **Ctrl + K + S**, que abrirá la configuración de atajos de teclado.

En su buscador digitar la oración: toggle integrated terminal, el cual mostrará la combinación de teclas que le corresponde. En caso de que no tenga asignado alguna combinación o que no es atractivo el tajo de teclado, se puede reconfigurarse haciendo doble clic sobre el nombre, y posteriormente presionar la combinación que se desee. Se debe tener presente que la asignación de teclado no interfiera con otras operaciones, ya que esta última dejarían de funcionar.

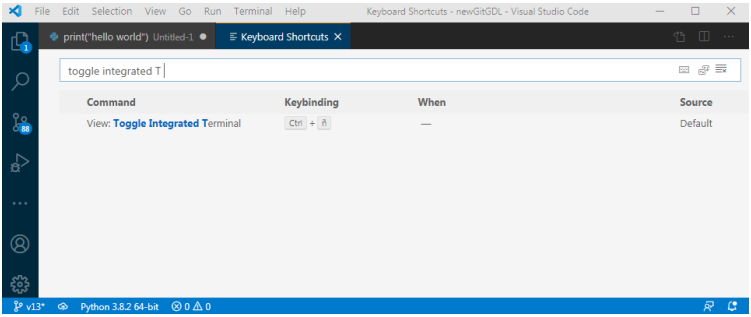



***Figura 7.*** Configuración del atajo de teclas de la terminal integrada.

### g) Personalizable

También es personalizable lo que significa que el editor se adapta a la exigencia del usuario y no de forma contraria, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias.

### h) Marketplace

Es una tienda de extensiones para VS Code el cual esta soportado por una comunidad de desarrolladores que suben sus plugin.

Gracias a su buscador se pueden encontrar infinidad de nueva funcionalidad para convertir el editor en un gran entorno de desarrollo. En el Marketplace se pueden conseguir temas nuevos, por si los que ya vienen instalados no son suficientes, así como soportes de resaltado para otros lenguajes y ayudas de autocompletado.

Para acceder al Marketplace basta presionar en el botón **Extensiones** que está en la parte izquierda del editor o presionar la combinación de teclas: **Ctrl + Shift + X**. Una vez en la interfaz de Marketplace solo debe escribir el nombre de algún plugin que se quiera integrar a VS Code.

Si se tiene dudas de lo que una extensión realiza, puede hacer clic sobre él, posteriormente se abrirá una ventana detallando toda la información acerca de este.

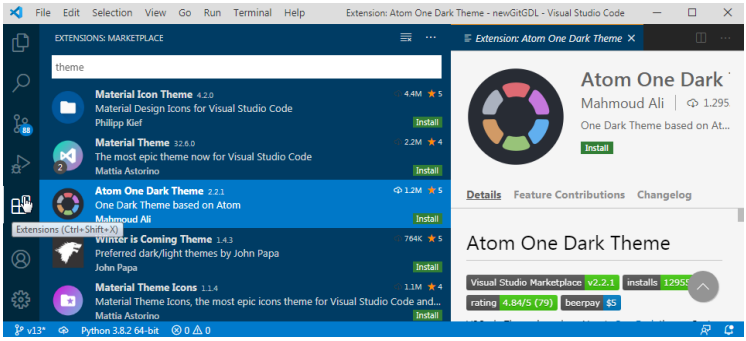



***Figura 8.*** Presentación de Marketplace.

## 1.6 Integración de Python con VS Code

Para tener VS Code funcional y lista para trabajar con Python se necesita de dos pasos:

1. Que Python ya esté instalado en el sistema operativo.
2. Integrar un plugin que permitirá trabajar cómodamente con Python y brinde la ayuda de autompletado.

Para ello acceder a Marketplace y en su buscador escribir: **python**, debe seleccionar la opción que se muestra en la imagen adjunta, posteriormente clicar en instalar.

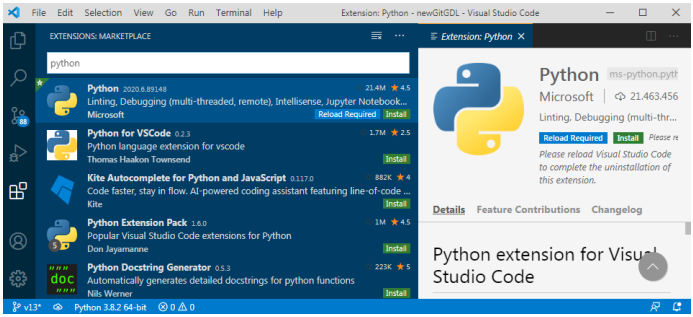



***Figura 9.*** Instalación del plugin Python.

Ahora ya está listo para hacer programas en Python.

## 1.7 Ejecutar código Python

En esta primera prueba se creará un programa que imprima por consola el clásico hola mundo. Se recomienda copiar el codigo a su editor de texto y guardarlo con el nombre: **saludo.py**,

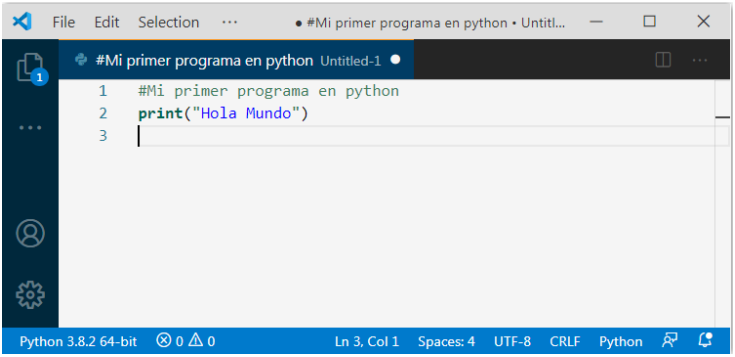



***Figura 10.*** Primer programa en Python.

Para guardar el archivo puede hacer uso del menú File seguido de la opción Save o usar el atajo de teclas Ctrl + S.

**Nota**: no olvidar el ".py" al final del nombre del archivo, esto es importante para decirle al sistema operativo que el archivo es un programa de Python.

El fichero puede ser almacenado en cualquier lugar de su disco duro, de preferencia en un espacio que sea fácil de hallar para su futura interacción con el archivo.

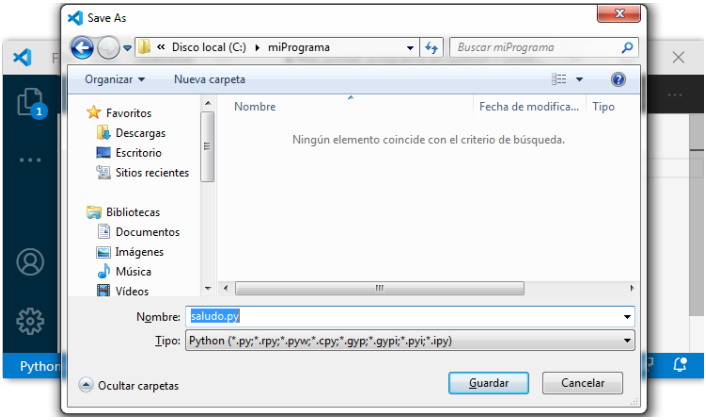



***Figura 11.*** Guardando el archivo con el nombre: saludo.py

Después de haber guardado el archivo, debe abrir una terminal y moverse a la carpeta del archivo (si está usando CMD debe hacer uso del comando CD para moverse entre directorios usando la terminal).

Una vez en directorio bastará escribir el nombre del programa: saludo.py seguido de la tecla **Enter.** En caso de que no funcione pruebe la alternativa: **python saludo.py.**

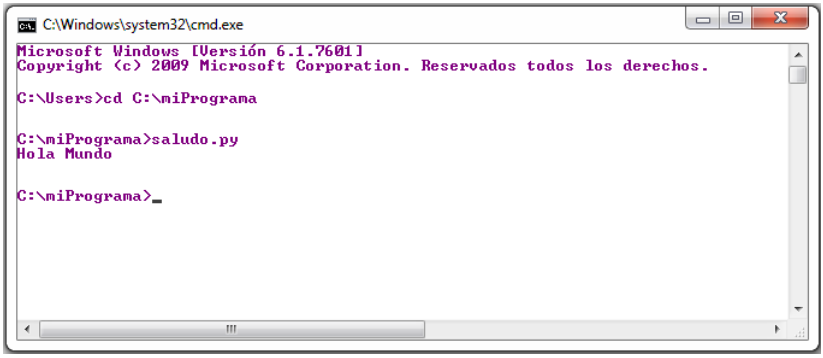



***Figura 12.*** Pasos para ejecutar un programa en Python usando CMD

**Nota:** para trabajar con Python es necesario conocer algunos comandos básicos de la terminal, que utilice, como puede ser:

moverse entre directorios, listar elementos del directorio, crear, mover, eliminar ficheros.

Otro atajo interesante que traen los Administradores de archivos es la posibilidad de abrir terminales en el directorio donde te encuentres.

En Windows existe el atajo **Shift + Anti-Clic (clic derecho)**, el cual desplegará la opción **Abrir ventana de comandos aquí,** con este paso lo que queda es ejecutar el archivo Python.

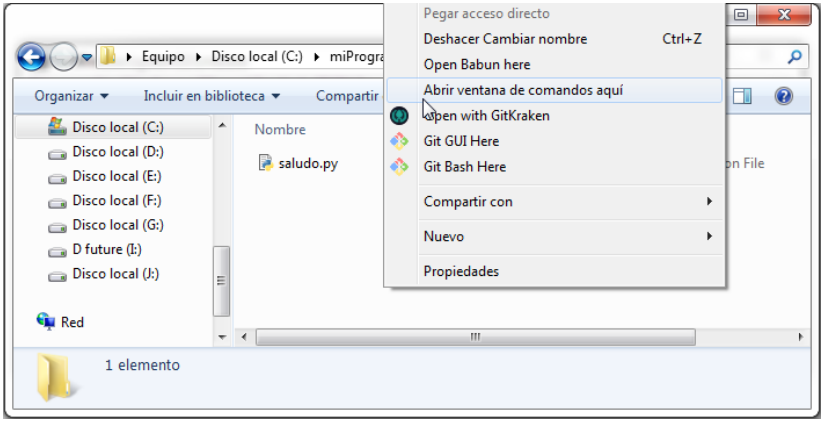



***Figura 13.*** Para abrir una terminal en la carpeta donde está ubicado

## 1.8 Ejecutar código Python en VS Code

Ejecutar el código desde VS Code es más sencillo. Debe abrir el directorio donde está almacenado su programa haciendo uso del menú **File** seguido de **Open folder…**

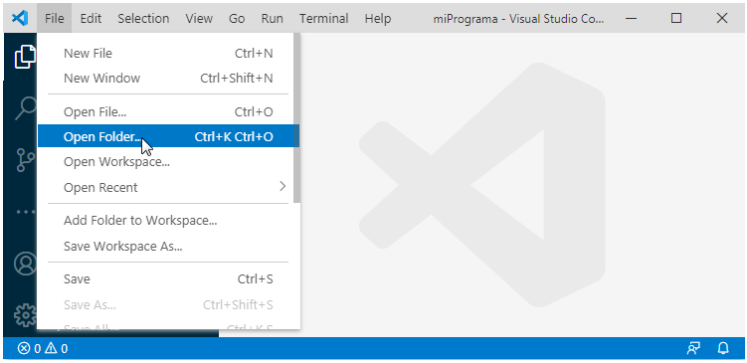



***Figura 14.*** Abrir un directorio en VS Code, usando Open Folder…

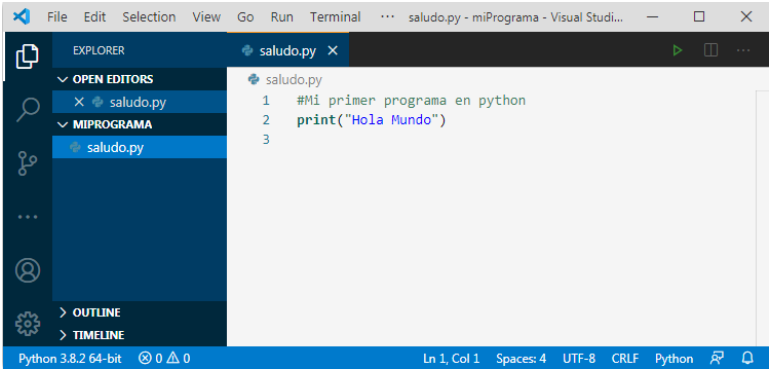



***Figura 15.*** Vista posterior al abrir un directorio con: Open Folder…

Abra el código fuente de su programa, en caso de no visualizarlo, usando el panel **Explorer** y haga clic sobre el nombre del archivo.

Existen dos maneras de ejecutar un programa de Python en VS Code:

1. La forma más sencilla es usando el botón **Ejecutar**; que es un triángulo verde que se encuentra en la parte superior derecha del editor.

La desventaja de usar este acceso directo es que funciona excelente con programas que no necesitan mucha interacción con el usuario.

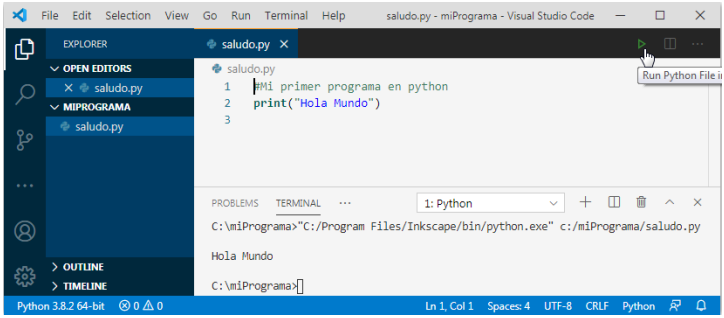



***Figura 16.*** Ejecutando un programa de Python con el botón Ejecutar.

2. La otra manera de ejecutar un código Python es haciendo uso del atajo del atajo de teclas de la terminal integrada **Ctrl + Ñ** que desplegará la terminal por defecto del sistema operativo, en este caso CMD.

   Posteriormente a presionar las teclas se abrirá una terminal cuya path (dirección) se ha posicionado en la carpeta que se pre-abrió con **Open Folder…**, Por lo que solo queda escribir el nombre del archivo: **saludo.py** o **python saludo.py.**

   La ventaja de ejecutar el código de esta forma es que el programa interactúa mejor con el usuario además de desplegar los errores en caso de que los hubiera.

   Su desventaja que necesita más pasos que al usar el botón ejecutar.

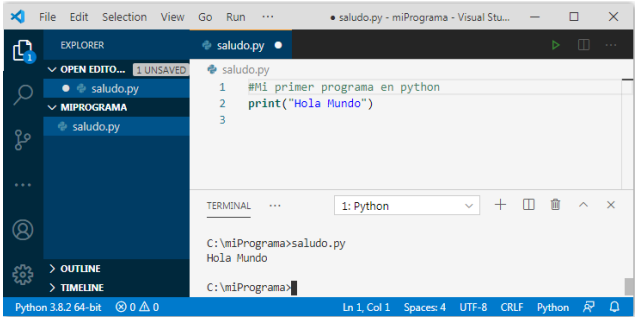



***Figura 17.*** Ejecutando un Python de la segunda manera.

## 1.9 ¿Por qué VS Code?

VS Code es una herramienta potente que permite editar código rápidamente y de forma inteligente gracias a sus atajos de teclado y extensión, además de que se puede hacer muchas tareas sin salir del editor, se puede administrar archivos o ejecutar el programa que se esté creando.

En caso de que algunas funcionalidades no se encontraran se puede hacer uso del Marketplace para buscar la extensión que se desea.

# 2

## Lenguaje y Algoritmos

El ser humano pensante por naturaleza ha hecho uso del lenguaje desde épocas remotas para comunicarse con los demás, a medida que evolucionó, desarrolló herramientas tecnológicas que facilitaron este proceso, pasando por señales de humo, palomas mensajeras, cartas, telégrafo, teléfono hasta hoy en día donde encontramos computadoras, tablets, smartphones, la red de redes (internet), los satélites, la nube, entre muchas otras que seguirán apareciendo. Particularmente son las herramientas denominadas computadoras las que nos llaman la atención para este particular, el ser humano se comunica con ellas, le da instrucciones que deben ejecutar retornando un resultado que nos servirá para tomar decisiones. (Aguilar, 2008)

Estas instrucciones que reciben las computadoras se hacen a través de *programas* para computadoras que se crean mediante un lenguaje de programación como el C++, Visual Studio, Delphi, Java, etc. Estos programas deben seguir una lógica y orden independientemente del lenguaje de programación en el que fueron hechos.

La forma estandarizada de crear esta lógica es mediante **algoritmos,** seguramente en algún momento de nuestras vidas hemos oído de esta palabra, o hemos hecho uso del mismo, por ejemplo cuando preparamos un café, cuando hacemos la tarea, en cada cosa que realizamos aplicamos algoritmos, y de manera especializada aquellos que se encuentran en el mundo de la ingeniería y las ciencias computacionales; el término algoritmo históricamente hablando se remonta a la antigüedad, el nombre se tomó en honor al matemático árabe “Abu al-Khwwarizmi” que vivió entre los siglos VIII y IX, quien se encargó de difundir el conocimiento de la antigua Grecia e India. Su aporte se centró no en dar nuevos teoremas o postulados matemáticos sino en simplificar la matemática a un nivel que pueda ser fácilmente entendido por el hombre común. Aunque él no creo el primer algoritmo su aporte ya mencionado le dio este honor.

## 2.1 ¿Qué es un algoritmo?

Un algoritmo es visto como un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.

***Figura 18.*** Pensando en una idea

Todo algoritmo por lo general consta de tres secciones o módulos principales. En la figura podemos observar las secciones que constituyen un algoritmo.

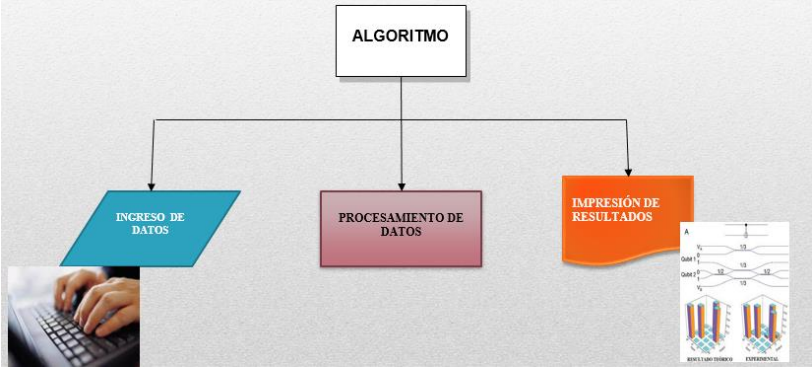



***Figura 19.*** Secciones de un algoritmo

Todo algoritmo requiere datos de entrada para poder realizar sus procesos de transformación de datos y poder finalmente emitir o imprimir resultados.

Así mismo un algoritmo debe tener ciertas características básicas a todos: (Cairó et al., 1993)

- **Determinismo**: El algoritmo, dado los mismos datos de entrada en diferentes ejecuciones del algoritmo, siempre debe arrojar los mismos resultados.
- **Precisión**: Los pasos que se siguen en el algoritmo deben ser precisados con claridad para evitar dudas.
- **Finitud**: Es decir debe tener un inicio y un fin. El algoritmo, independientemente de la complejidad, siempre debe ser de longitud finita.

## 2.2 Representación de algoritmos

Los algoritmos se representan empleando *Pseudocódigos* y *Diagramas.* (De la cruz Roca & Condor Caballero, 2017)

### a. Pseudocódigos

Los pseudocódigos son conjunto de instrucciones del lenguaje natural, como el castellano o el inglés, un ejemplo se muestra en la siguiente figura.

```
DATOS                                   Identificadores
    Salida
        Promedio                        P
    Entrada
        Primera Nota Parcial            N1
        Segunda Nota Parcial            N2
        Tercera Nota Parcial            N3
    Inicio
      Leer ( N1 )
      Leer ( N2 )
      Leer ( N3 )
      P ← (N1 + N2 + N3)/3
      Escribir ( P )
    Fin
```

***Figura 20.*** Algoritmo mediante Pseudocódigo

### b. Diagramas

Es una representación que usa símbolos predefinidos para diagramar un algoritmo, con el fin de que sea fácil de seguir

la lógica, indicando el inicio y el termino de los mismos. Las más conocidas son el *diagrama de flujo* y *Carta N-S*.

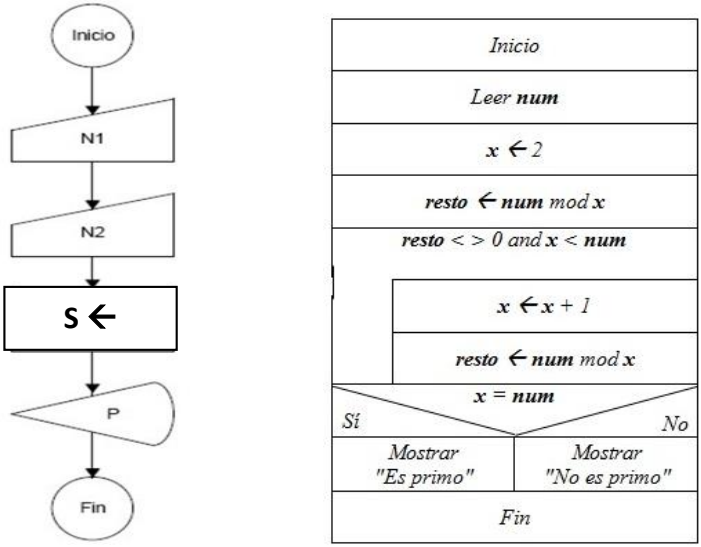



***Figura 21.*** Diagrama de flujo (izq.) y Carta N-S (der.)

## 2.3 Datos

En algoritmos un dato es un valor numérico o no numérico que se toma como entrada para poder ejecutar el algoritmo. Dentro de los tipos de datos se definen a los siguientes: (Cairó et al., 1993)

### a. Datos numéricos

Dentro de los tipos de datos numéricos encontramos valores enteros y valores reales. Los enteros son números que pueden estar precedidos del signo + o - y que no tienen parte decimal, a diferencia de los valores reales, algunos ejemplos

128 152S -714 8 530 16 235 -14 780

7.5 128.0 -37.865 129.7 16 000.50 -15.0

## b. Datos alfanuméricos

Dentro de este tipo de datos encontramos el tipo carácter (un solo carácter) y el tipo cadena (secuencia de caracteres). Son datos cuyo contenido pueden ser letras del abecedario (a, b, c .. z), dígitos (0, 1, 2 , .., 9) o sím- bolos especiales ( #, ?, $, \, *, etc.)

Un dato tipo **carácter** contiene un solo carácter. Por ejemplo:

'a' 'B' '$' '9' '-' '#* 'f'

Un dato tipo **cadena** contiene un conjunto de caracteres.

“abcde” “$9#7” “Carlos Gómez” “Rosario” “754-27-22”

## c. Datos lógicos

Dentro de este tipo de datos encontramos los booleanos. Son datos que sólo pueden tomar dos valores: verdadero (true) o falso (false).

```
#TIPOS DE VARIABLE EN PYTHON
#El tipo de variable estará determinado por el tipo de dato que almacena
#Enteros: vasta agregar un valor entero
a = 10
b = 11

#Decimales: debe agregarse un número con punto flotante
a = 10.0
b = 12.5
c = 3.1416
#Booleanos: Datos que pueden tomar verdadero o falso
#Podrías usar las palabras reservadas: True y False
verdad = True
falso = False
#Podrías usar los enteros
```

```
verdad = 1
falso = 0

#Texto o Cadena o String
saludo = "hola"
color = "Rojo"
#También se pueden usar comilla simple
saludo = 'hola'
color = 'Rojo'
```

## 2.4 Identificadores, constantes y variables

### a. Identificadores

Llamaremos identificador al nombre que se les da a las casillas de memoria en una computadora, lugar donde se guarda o almacena los valores que se le asigna a las constantes y variables. El primer carácter que forma un identificador debe ser una letra (a, b. c, .. , z).

### b. Constantes

Las constantes son objetos que no cambian de valor durante la ejecución de un algoritmo.

### c. Variables

Las variables son objetos que pueden cambiar su valor durante la ejecución del algoritmo.

## 2.5 Operaciones combinadas y lógica operacional

### a. Operaciones aritméticas

Para poder realizar operaciones aritméticas necesitamos de operadores aritméticos, una operación combinado siempre se evalúa comenzando por el lado izquierdo, detectando los operadores de mayor prioridad. El resultado de una operación

aritmética será un número. Si los operadores que se evalúan son del mismo nivel entonces se comienza por el operador que se encuentra al extremo izquierdo.

Si la operación combinada presenta paréntesis de agrupación entonces se debe evaluar comenzando por el paréntesis más interno.

Los operadores aritméticos son:

+ : suma

- : resta

* : multiplicación

/ : división

Div : división entera

Mod : resto de una división entera

^ : potencia ( también usada para raíz)

Para que el resultado de la operación combinada salga correcto, es importante tener en cuenta el orden de resolución de los operadores.

A continuación, en la tabla, presentamos la jerarquía de operaciones aritméticas. (Cairó et al., 1993)

**Tabla 5**

*Prioridad de operadores aritméticos*

| Operador | Jerarquía | Operación |
|---|---|---|
| ^ | Mayor | Potencia |
| * , /, mod, div | ↓ | Multiplicación, división, modulo, división entera |
| + , - | Menor | Suma, resta |

A continuación, algunos ejemplos demostrativos.

a) 6 * 5^3 / 5 div 3

6 * 125 / 5 div 3

750 / 5 div 3

150 / 3

**50**

b) 4*2*(160 mod 3^3) div 5*13-28

4*2*(160 mod 27) div 5*13- 28

4*2*25 div 5*13-28

8*25 div 5*13 – 28

200 div 5 *13 – 28

40 * 13 – 28

520 – 28

**492**

**Nota:** Se resalta la operación que se realiza a continuación.

c) 15/3*(7+(68-15*33+(45^2/5)/3)/2)+19

15/3*(7+(68-15*33+(2025/5)/3)/2)+19

15/3*(7+(68-15*33+ 405 / 3)/2)+19

15/3*(7+(68-495+ 405/ 3)/2)+19

15/3*(7+(68-495+ 135)/2)+19

15/3*(7+(-427+ 135)/2)+19

15/3*(7+ (-292)/2)+19

15/3*(7-146 ) +19

15/3*(-139) +19

5 *-139 +19

-695 + 19

**-676**

**Tabla 6**
*Operaciones Aritméticos en Python*

| Operaciones Aritméticos en Python | | |
|---|---|---|
| **Operador** | **Jerarquía** | **Operación** |
| ** | Mayor a Menor | Potencia |
| *, /, %, // | | Multiplicación, división, modulo, división entera |
| +, - | | Suma, resta |

### b. Operaciones relacionales

Constituidos por operadores relacionales. (Cairó et al., 1993)

**Tabla 7**
*Prioridad de operadores relacionales*

| Operador | Ejemplo | Resultado |
|---|---|---|
| = | "cola" = "sola" | FALSO |
| < > | 'a' < > 'x' | VERDADERO |
| < | 7 < 151 | VERDADERO |
| > | 22 > 10 | VERDADERO |
| < = | 14 < = 20 | VERDADERO |
| > = | 55 > = 21 | VERDADERO |

En prioridad los operadores relacionales son después de los operadores aritméticos. Una operación combinada de este tipo debe comenzar evaluando y simplificando las operaciones aritméticas, según lo indicado en esa sección. Vea el siguiente ejemplo.

a) ((1240 mod 6*2^4) > (7+8*3^4)) > ((15*2) = (30*2/4))

((1240 mod 6*16) > (7+8*3^4)) > ((15*2) = (30*2/4))

((4*16) > (7+8*3^4)) > ((15*2) = (30*2/4))

(64> (7+8*3^4)) > ((15*2) = (30*2/4))

(64 > (7+8*81)) > ((15*2) = (30*2/4))

(64 > (7+648)) > ((15*2) = (30*2/4))

(64> 655) > ((15*2) = (30*2/4))

FALSO > ((15*2) = (30*2/4))

FALSO > (30 = (30*2/4))

FALSO > (30 = (60/4))

FALSO > (30 = 15)

FALSO > FALSO

**FALSO**

**Tabla 8**

*Operaciones relacionales en Python*

| Operaciones relacionales en Python | | | |
|---|---|---|---|
| **Operador** | **nombre** | **Ejemplo** | **Resultado** |
| == | Igualdad | "Cola" == "Sola" | False |
| != | diferente | a' != 'x' | True |
| < | menor que | 7 < 151 | True |
| > | mayor que | 22 > 10 | True |
| <= | menor o igual que | 14 <= 20 | True |
| >= | mayor o igual que | 55 >= 21 | True |

### c. Operaciones lógicas

Las operaciones lógicas son las ultimas en realizarse, opera sobre valores booleanos VERDAERO (1) y FALSO (0), la siguiente tabla muestra su jerarquía y que operadores se considera. (Cairó et al., 1993)

**Tabla 9**
*Prioridad de operadores lógicos.*

| Operador | Jerarquía |
|---|---|
| NO | Mayor |
| Y | ↓ |
| O | Menor |

A continuación, se muestra el siguiente ejercicio

a) **NO** ( 18 >= 6^2) **O** (43 – 8 * 2 div 2 <> 3 * 2 div 3)

**NO** ( 18 >= 36) **O** (43 – 8 * 2 div 2 <> 3 * 2 div 3)

**NO** (FALSO) **O** (43 – 8 * 2 div 2 <> 3 * 2 div 3)

**NO** (FALSO) **O** (43 –16 div 2 <> 3 * 2 div 3)

**NO** (FALSO) **O** (43 – 8 <> 3 * 2 div 3)

**NO** (FALSO) **O** (43 – 8 <> 6 div 3)

**NO** (FALSO) **O** (43 – 8 <> 2)

**NO** (FALSO) **O** (35 <> 2)

**NO** (FALSO) **O** VERDADERO

VERDADERO **O** VERDADERO

**VERDADERO**

**Tabla 10**
*Operadores lógicos en Python*

| Operadores lógicos en Python | | |
|---|---|---|
| **Operador** | **nombre** | **Jerarquía** |
| **not** | Negación | Mayor a Menor |
| **and** | Y | |
| **or** | O | |

## 2.6 Ejercicios complementarios

Si A = 9, B = 8 y C = 2, evaluar las siguientes expresiones:

a) B * A - B * B / 4 * C

8 * 9 – 8 * 8 / 4 * 2

72 – 64 / 4 * 2

72 – 16 *2

72 – 32

**40**

b) (A * B) / 3 * 3

(9 * 8) / 3 * 3

72 / 3 * 3

24 * 3

**72**

c) (((B + C) / 2 * A + 10) * 3 * B) – 6

(((8 + 2) / 2 * 9 + 10) * 3 * 8) – 6

((10 / 2 * 9 + 10) * 3 * 8) – 6

((5 * 9 + 10) * 3 * 8) – 6

((45 + 10) * 3 * 8) – 6

(55 * 3 * 8) – 6

(165 * 8) – 6

1320 – 6

**1314**

Resolver:

d) 40/5*5+6/2*3+4-5*2/10

8*5+6/2*3+4-5*2/10

40+3*3+4-10/10

40+9+4-1

49+4-1

53-1

**52**

e) 500- ((6 -1)*8/4*3+16/(10-2))-5

500- (5*8/4*3+16/8)-5

500- (40/4*3+16/8)-5

500- (10*3+2)-5

500- (30+2)-5

500- 32-5

468-5

**463**

f) 4 / 2 * 3 / 6 + 6 / 2 / 1 / 5 mod 2 / 4 * 2

2 * 3 / 6 + 6 / 2 / 1 / 5 mod 2 / 4 * 2

6 / 6 + 6 / 2 / 1 / 5 mod 2 / 4 * 2

1 + 6 / 2 / 1 / 5 mod 2 / 4 * 2

1 + 3 / 1 / 5 mod 2 / 4 * 2

1 + 3 / 5 mod 2 / 4 * 2

1 + 0.6 mod 2 / 4 * 2

1 + 0.6 / 4 * 2

1 + 0.15 * 2

1 + 0.30

**1.30**

g) 14 – (7 + 4 * 3 - [(-2)* 2 * 2 - 6) ] ) + (22 + 6 - 5 * 3) + 3 - (5 - 23 div 2)

Como ya se conoce la forma de procesar, se toma y reduce todos los paréntesis internos.

14 – (7 + 4 * 3 - [(-4) * 2 - 6)]) + (28 - 15) + 3 - (5 - 11)

14 – (7 + 4 * 3 - [(-8) - 6)]) + 13 + 3 - (-6)

14 – (7 + 4 * 3 - [-14]) + 13 + 3 - (-6)

14 – (7 + 12 - [-14]) + 13 + 3 - (-6)

14 – (19 - [-14]) + 13 + 3 - (-6)

14 – 33 + 13 + 3 - (-6)

-19 + 13 + 3 - (-6)

-6 + 3 - (-6)

-3 - (-6)

**3**

h) ((1580 mod 6*2^7) > (7+8*3^4)) > ((15*2) = (60*2/4))

((1580 mod 6*128) > (7+8*81)) > ((30) = (120/4))

((2*128) > (7+648)) > (30 = (120/4))

((256) > (7+648)) > ((30) = (30))

((256) > (655)) > V

F > V

**F**

i) NO (15 >= 7^2) O (43 – 8 * 2 div 4 <> 3 * 2 div 2)

NO (15 >= 49) O (43 – 16 div 4 <> 6 div 2)

NO (V) O (43 – 4 <> 3)

F O (39 <> 3)

F O (V)

**V**

j) (15<=7*3^2 Y 8>3 Y 15<6) **O NO** (7*3 < 5+12*2 div 3^2)

(15<=7*9 **Y** V **Y** F) **O NO** (21 < 5+ 24 div 9)

(15<=63 **Y** V **Y** F) **O NO** (21 < 5+ 2)

(V **Y** V **Y** F) **O NO** (21 < 7)

(V **Y** V **Y** F) **O NO** (F)

(V **Y** V **Y** F) **O** V

Cuando existe una triple comparación lógica, se desglosa en dos grupos simples vinculados mediante una **Y** entonces se tiene (V **Y** V **Y** F) =>> (V **Y** V) **Y** (V **Y** F)

(V **Y** V) **Y** (V **Y** F) **O** V

(V **Y** F) **O** V

F **O** V

**V**

# 3

## Instrucciones Básicas en un Algoritmo

En este capítulo se busca afianzar el criterio de análisis y diseño de algoritmos, para ello nos valemos de herramientas como los Pseudocódigos, describimos sus partes y usos.

Así mismo emplearemos diagramas de flujo, como una representación equivalente a un Pseudocódigo, esta práctica nos llevara a desarrollar mejor el aspecto de análisis y diseño de un algoritmo.

A continuación, se describe las instrucciones básicas que se utilizaran para construir algoritmos. (Cairó et al., 1993)

### 3.1 Leer

La instrucción de *Lectura* viene a ser aquella mediante la cual se ingresa uno o más datos por medio del teclado que luego internamente será guarda en una variable, este valor se requerirá para realizar cálculos y hallar la solución a un algoritmo. Permite:

- Solicitar un dato inicial
- Requerir un dato de entrada

Su declaración dentro de un Pseudocódigo (Algoritmo en instrucciones) es:

**Leer** <Dato, dato2, dato3,…>

```
#LEER DATOS EN PYTHON
```

```
#Leer datos de teclado es facil gracias a la función: input() de python
#El cual nos devolverá una cadena de texto
#El cual debemos capturar en alguna variable

texto = input()
#Además input puede tener un parámetro indicando al usuario
#que debe ingresar

mes = input("Ingrese mes: ")
# CASTING (Conversiones de tipo de dato)
# como la función input() retorna un texto
# si nos otro queremos que nos devuelva un entero
# debemos hacer una covnersión usando la función int()

numero = input()    #numero es string
entero = int(numero)

#O la simplificada
entero = int( input())

# Si en otro caso queremos trabajar con decimales
# Se debe usar la función float()

numero = input()    #numero es string
decimal = int(numero)

#O de la forma directa
decimal = float( input())
```

## 3.2 Asignar

Esta instrucción asigna un valor a una variable, mediante un símbolo flecha "←", esta flecha reemplaza al símbolo igual para no confundir con el operador lógico "=", a diferencia de la instrucción leer esta no se ingresa por teclado sino mediante una instrucción en el mismo Pseudocódigo. Permite:

- Operar sobre el dato obteniendo nuevo valor
- Procesar los datos, obteniendo nuevo valor

Su empleo dentro de un Pseudocódigo (Algoritmo en instrucciones) es:

Variable ← <Valor>

```
#OPERADOR ASIGNACIÓN EN PYTHON
#En Python como en muchos otros lenguajes se usa el operador:
#  '='

numero = 5    #Asigna 5 a la variable
```

## 3.3 Escribir

La instrucción *Escribir* se emplea para poder visualizar resultados o valores que contiene una variable, también para mostrar mensajes, por lo general el resultado se aprecia en la pantalla de la computadora o impreso en papel. Permite:

- Mostrar el resultado
- Visualizar el resultado
- Imprimir el valor resultante

Su empleo dentro de un Pseudocódigo (Algoritmo en instrucciones) es:

**Escribir** <Valor Resultante>

```
#ESCRIBIR O MOSTRAR EN PYTHON
```

```
# Para mostrar resultados por la pantalla tenemos la función
# print()

numero = 5
print(numero)   #Muestra el valor de 'numero' por pantalla

dia = 10
mes = "Septiembre"
anio = "1990"   #Es recomendable usar solo caracteres Ingleses para evitar error

print(dia, mes, anio)   #Salida:   10 Septiembre 1990
#Note que los espacios fueron agregados por la       función print

#format()
#Otra manera interesante de imprimir es utilizando la función format()
# "cadena de texto".formta()

print("dia: {} mes: {} anio: {}".format(dia, mes, anio))
#Salida: dia: 10 mes: Septiembre anio: 1990

#Se sugiere al estudiante profundizar más.
# Nota:
# Para no tener problemas con los caracteres especiales
# del lenguaje Español como pueden ser las tildes y la eñe
# es recomendable poner la siguiente sentencia al inicio de
# cada programa como comentario.

# -*- coding: utf-8 -*-
```

A continuación, desarrollaremos nuestros primeros algoritmos, para ello emplearemos las tres instrucciones descritas: Leer, asignar y escribir.

## 3.4 Ejercicios del tema

**Ejercicio 1.**

Se desea calcular la distancia recorrida (m) por un móvil que tiene velocidad constante (m/s) durante un tiempo t (s), considerar que es un MRU (Movimiento Rectilíneo Uniforme).

**Solución:**

Primero se procede a determinar los datos de entrada y la salida a obtener, obsérvese.

| **Entrada** | **Identificador** |
|---|---|
| Velocidad Constante (m/s) | **V** |
| Tiempo (s) | **T** |
| **Salida** | |
| Distancia Recorrida (m) | **D** |

Ahora se establece el proceso requerido para obtener el resultado pedido:

Se sabe que el cálculo de la velocidad en MRU es por formula: $\boldsymbol{V = \frac{D}{T}}$

Despejando la distancia se tiene: $\boldsymbol{D = V\ x\ T}$

Una vez determinado los tres elementos de un algoritmo (entrada, proceso, salida) procedemos a construir la parte medular del Pseudocódigo.

**INICIO**

**Escribir** ("ingrese la velocidad y el tiempo de la unidad móvil") //muestra el mensaje en pantalla

**Leer** (V) // se lee y almacena en la variable V la velocidad del vehículo

**Leer** (T) // se lee y almacena en la variable T el tiempo que tarda el vehículo

D ← V * T // Se procede a calcular la distancia y asignar el valor a la variable D

**Escribir** (D) //Se imprime el valor que contiene la variable D

**FIN**

Este segmento de Pseudocódigo se puede llevar a cualquier lenguaje de programación para su implementación, solamente requiere reemplazar las instrucciones por los comandos que use el lenguaje.

**Nota:** Las dos barras inclinadas "//" indica que lo que sigue en esa línea es un comentario, no interviene en ninguna de las instrucciones.

El código en Python para resolver el Ejercicio1 es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio1: CALCULAR DISTANCIA.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese la velocidad y el tiempo de la unidad móvil")
#convirtiendo entrada a Entero
V = float( input("Velocidad: ") )
T = int( input("Tiempo: ") )

#Proceso
D = V*T

#Salida
print(D)
```

```
-------------------------------------------------------
Ejercicio1: CALCULAR DISTANCIA.
```

```
-----------------------------------------------------
Ingrese la velocidad y el tiempo de la unidad móvil
Velocidad: 10
Tiempo: 20
200.0
```

**Ejercicio 2.**

Se necesita obtener el promedio simple de un estudiante a partir de sus tres notas parciales N1, N2 y N3.

**Solución:**

Procederemos a determinar los datos de entrada y la salida a obtener, obsérvese.

| **Entrada** | **Identificador** |
|---|---|
| Promedio | **P** |
| **Salida** | |
| Primera Nota Parcial | **N1** |
| Segunda Nota Parcial | **N2** |
| Tercera Nota Parcial | **N3** |

Ahora se establece el proceso "Calcular Promedio" para obtener el resultado pedido, como ustedes saben el promedio simple de cualquier dato, se halla, sumando todos los datos y dividiendo entre el número de datos sumados, por ejemplo, les vamos a asignar valores a los identificadores.

N1 = 14

N2 = 13

N3 = 15

P = (14 + 13 + 15) / 3

Entonces P (Promedio) tomaría el valor de 14

Por lo tanto, el cálculo del promedio es: $\boldsymbol{P} = \frac{\mathbf{N1+N2+N3}}{\mathbf{3}}$

Una vez determinado los tres elementos de un algoritmo (entrada, proceso, salida) procedemos a construir la parte medular del Pseudocódigo. Siempre colocar la palabra INICIO para comenzar y FIN al terminar.

**INICIO**

**Escribir** ("Ingrese las 3 notas del alumno N1, N2, N3") //muestra el mensaje en pantalla

**Leer** (N1) // se lee y almacena en la variable N1 la nota número uno del alumno

**Leer** (N2) // se lee y almacena en la variable N2 la nota número dos del alumno

**Leer** (N3) // se lee y almacena en la variable N3 la nota número tres del alumno

P ← (N1 + N2 + N3) / 3 // Se procede a calcular el promedio y asignar el valor a la variable P

**Escribir** (P) //Se imprime el valor que contiene la variable P: promedio obtenido

**FIN**

El código del Ejercicio 2 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio2: PROMEDIO DE 3 NOTAS.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese las 3 notas del alumno N1 , N2, N3")
N1 = int( input("N1: "))
N2 = int( input("N2: "))
N3 = int( input("N3: "))

#Proceso
P = int( (N1+N2+N3)/3 )
```

```
#Salida
print("\nSalida: ")
print(P)
```

```
-----------------------------------------------------
Ejercicio2: PROMEDIO DE 3 NOTAS.
-----------------------------------------------------
Ingrese las 3 notas del alumno N1, N2, N3
N1: 15
N2: 10
N3: 14

Salida:
13
```

**Ejercicio 3.**

Se necesita elaborar un algoritmo que solicite el número de respuestas correctas, incorrectas y en blanco, correspondientes a postulantes, y muestre su puntaje final considerando que por cada respuesta correcta tendrá 3 puntos, respuestas incorrectas tendrá -1 y respuestas en blanco tendrá 0.

**Solución:**

Procederemos a determinar los datos de entrada y la salida a obtener, obsérvese.

| **Entrada** | **Identificador** |
|---|---|
| Puntaje Final | **PF** |
| **Salida** | |
| Número de Respuestas Correctas | **RC** |
| Número de Respuestas Incorrectas | **RI** |
| Número de Respuestas en Blanco | **RB** |

| Otras variables intermedias para usar: | |
|---|---|
| Puntaje de Respuestas Correctas | **PRC** |
| Puntaje de Respuestas Incorrectas | **PRI** |
| Puntaje de Respuestas en blanco | **PRB** |

Se ha considerado variables intermedias al momento de determinar las entradas y salidas, debido a que durante la ejecución del pseudocódigo habrá necesidad de almacenar información adicional para los cálculos, esta es una situación perfectamente normal, uno puede usar variables adicionales si así lo cree conveniente. (Introducción a la Programación en Java., 2007)

Ahora se establece el proceso de cálculo para obtener el resultado pedido.

Para calcular el **puntaje de respuestas correctas** (PRC) debemos multiplicar el número de respuestas correctas (RC) por el valor de cada respuesta correcta (3), entonces:

$$PRC = RC \times 3$$

Para el puntaje de respuestas incorrectas (PRI):

$$PRI = RI \times -1$$

El puntaje de respuestas en blanco es:

$$PRB = RB \times 0$$

Finalmente, la suma de estos tres será el puntaje final (PF).

$$PF = PRC + PRI + PRB$$

Determinado los tres elementos de un algoritmo (entrada, proceso, salida) procedemos a construir la parte medular del Pseudocódigo.

**INICIO**

**Escribir** ("Ingrese número de respuestas correctas")

**Leer** (RC)

**Escribir** (“Ingrese número de respuestas incorrectas”)

**Leer** (RI)

**Escribir** (“Ingrese número de respuestas en blanco”)

**Leer** (RB)

// Procedemos a realizar el cálculo de cada grupo de respuestas

PRC ← RC * 3 //No olvidar que el asterisco **“*”** es el símbolo de multiplicación

PRI ← RI * -1

PRB ← RB * 0

PF ← PRC + PRI + PRB

**Escribir** (PF)  //Se imprime el puntaje final

**FIN**

El código del Ejercicio 3 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio3: PUNTAJE FINAL.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese número de respuestas correctas: ")
RC = int( input())
print("Ingrese número de respuestas incorrectas: ")
RI = int( input())
print("Ingrese número de respuestas en blanco: ")
RB = int( input())

#Proceso
PCR = RC*3
PRI = RI*-1
PRB = RB*0
```

```
PF = PCR + PRI + PRB
#Salida
print("El puntaje total es:", PF)
```

```
-----------------------------------------------------
Ejercicio3: PUNTAJE FINAL.
-----------------------------------------------------
Ingrese número de respuestas correctas:
8
Ingrese número de respuestas incorrectas:
6
Ingrese número de respuestas en blanco:
4
El puntaje total es: 18
```

**Ejercicio 4.**

Elaborar un algoritmo que permita ingresar el número de partidos ganados, perdidos y empatados, por ABC club en el torneo apertura, se debe de mostrar su puntaje total, teniendo en cuenta que por cada partido ganado obtendrá 3 puntos, empatado 1 punto y perdido 0 puntos.

**Solución:**

Procederemos a determinar los datos de entrada, la salida y variables intermedias, obsérvese.

| **DATOS** | **Identificador** |
|---|---|
| **Salida** | |
| Puntaje Total | **PT** |
| **Entrada** | |
| Número de Partidos Ganados | **PG** |

| | |
|---|---|
| Número de Partidos Empatados | **PE** |
| Número de Partidos Perdidos | **PP** |
| **Intermedio** | |
| Puntaje de Partidos Ganados | **PPG** |
| Puntaje de Partidos Empatados | **PPE** |
| Puntaje de Partidos Empatados | **PPP** |

El proceso de cálculo para obtener el resultado pedido es similar al anterior (Ejercicio 3), procedemos a construir el pseudocódigo.

**INICIO**

**Escribir** ("Ingrese número de partidos ganados")

**Leer** (PG)

**Escribir** ("Ingrese número de partidos empatados")

**Leer** (PE)

**Escribir** ("Ingrese número de partidos perdidos")

**Leer** (PP)

// Procedemos a realizar el cálculo de cada grupo de partidos

PPG ← PG * 3

PPE ← PE * 1

PPP ← PP * 0

// Calculamos el puntaje total del equipo ABC club

PT ← PPG + PPE + PPP

**Escribir** (PT) //Se imprime el puntaje final

**FIN**

El código del Ejercicio 4 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
```

```
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio4: PUNTAJE TOTAL DE PARTIDOS.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese número de partidos ganados")
PG = int( input())
print("Ingrese número de partidos empatados")
PE = int( input())
print("Ingrese número de partidos perdidos")
PP = int( input())

#Proceso
PPG = PG*3
PPE = PE*1
PPP = PP*0

PF = PPG + PPE + PPP

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Puntaje Final: ", PF)
```

```
-------------------------------------------------------
Ejercicio4: PUNTAJE TOTAL DE PARTIDOS.
-------------------------------------------------------
Ingrese número de partidos ganados
5
Ingrese número de partidos empatados
3
Ingrese número de partidos perdidos
4

SALIDA:
-------------------------------------------------------
Puntaje Final: 18
```

## Ejercicio 5.

Elaborar un algoritmo que permita calcular el número de micro discos 3 .5 necesarios para hacer una copia de seguridad, de la información almacenada en un disco cuya capacidad se conoce. Hay que considerar que el disco duro está lleno de información, además expresado en gigabyte. Un micro disco tiene 1.44 megabyte y un gigabyte es igual a 1,024 megabyte.

**Solución:**

Procederemos a determinar los datos de entrada, la salida y variables intermedias, obsérvese.

| **Entrada** | **Identificador** |
|---|---|
| Número de Gigabyte del Disco Duro | **GB** |
| Número de Megabyte del Disco Duro | **MG** |
| **Salida** | |
| Número de Micro Disco 3.5 | **MD** |

El proceso de cálculo para determinar el número de Megabytes (**MG)** dado la cantidad de Gigabytes (**GB**) es MG = 1024 x GB, esto se puede determinar también si se aplica la regla de tres simple. Para calcular el número de Micro discos de 3.5 se procede a dividir el número de Megabytes (MD) calculados entre 1.44 que es la capacidad de un solo Micro disco, así:

$$MD = MG / 1.44$$

Determinado el proceso de conversión entonces construimos el pseudocódigo.

**INICIO**

**Leer** (GB)

MG ← GB * 1,024

MD ← MG / 1.44

**Escribir** (MD)

**FIN**

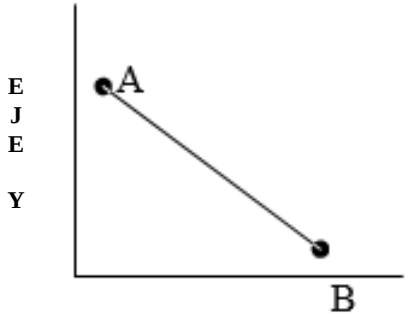
El código del Ejercicio 5 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
import math #librería necesaria para usar funciones Matemáticas
#en este caso math.ceil(), que redondea un numero al Entero      superior

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio5: NÚMERO DE MICRO DISCOS 3.5 NECESARIOS")
print("-------------------------------------------------------")

#Entradas
print("Ingrese GB: ")
GB = float( input())

#Proceso
MG = GB*1024
MD = MG/1.44

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(MD)
#En caso de Decimal Aproximar al siguiente entero
#Ya que la parte decimal debe ser almacenada en otro DISCO 3.5
print("Numero de Discos necesarios: ", math.ceil(MD))
```

```
-------------------------------------------------------
Ejercicio5: NÚMERO DE MICRO DISCOS 3.5 NECESARIOS
-------------------------------------------------------
Ingrese GB:
2

SALIDA:
-------------------------------------------------------
1422.2222222222222
Número de Discos necesarios: 1423
```

**Ejercicio 6.**

Se tiene los puntos A y B en el cuadrante positivo del plano cartesiano, elabore el algoritmo que permita obtener la distancia entre A y B.




**Solución:**

Un punto en el plano tiene dos coordenadas (X ,Y), entonces el punto A tendrá sus coordenadas (AX, AY) y el punto B de manera similar (BX, BY), luego se tiene los siguientes datos:

| **Entrada** | **Identificador** |
|---|---|
| Coordenada X de A | **AX** |
| Coordenada Y de A | **AY** |
| Coordenada X de B | **BX** |
| Coordenada Y de B | **BY** |
| **Salida** | |
| Distancia entre el punto A y B | **D** |

Para determinar la distancia entre dos puntos, empleamos la

fórmula matemática siguiente:

$$\boldsymbol{D = \sqrt{(AX - BX)^2 + (AY - BY)^2}}$$

Esta fórmula la podemos encontrar en los textos de matemática básica o geometría plana. Determinado el proceso de cálculo construimos el pseudocódigo.

**INICIO**

**Leer** (AX)

**Leer** (AY ) // se lee las coordenadas del punto A

**Leer** ( BX)

**Leer** ( BY ) // se lee las coordenadas del punto B

// Procedemos a realizar el cálculo matemático, el símbolo ^ es usado para la potencia

D ← **(**(AX-BX) ^2 + (AY-BY) ^2**)** ^0.5

Escribir ( D )

**FIN**

**Nota:** El cálculo de la raíz cuadrada se da usando el símbolo de potencia elevado a la 0.5.

El código del Ejercicio 6 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio6: DISTANCIA ENTRE 2 PUNTOS A y B, en 2D.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese coordenadas del Punto A: ")
AX = float(input("Ax: "))
AY = float(input("Ay: "))

print("Ingrese coordenadas del Punto B: ")
BX = float(input("Bx: "))
```

```
BY = float(input("By: "))

#Proceso
D = ( (AX-BX)**2 + (AY-BY)**2 )**0.5

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Resultado:", D)
```

```
Ingrese coordenadas del Punto A:
Ax: 0
Ay: 0
Ingrese coordenadas del Punto B:
Bx: 1
By: 1

SALIDA:
-------------------------------------------------------
Resultado: 1.4142135623730951
```

## 3.5 Pseudocódigo y sus partes

El pseudocódigo utiliza una serie de palabras clave o palabras especiales que va indicando lo que significa en el algoritmo según el dato ingresado.

```
Algoritmo <nombre alg>
Var
  <nombre>: <tipo de dato>
Const
  <nombre> = valor
Inicio
  <Instrucciones>
     …..
     …..
Fin
```

**Algoritmo:** Esta parte es donde se coloca el nombre del algoritmo, se escribe la palabra "Algoritmo" seguida a continuación del nombre del algoritmo. El nombre debe hacer alusión a lo que el

algoritmo realiza.

**Inicio y Fin:** En este segmento es donde se colocan las instrucciones que va a ejecutar el algoritmo. Similar a lo que hicimos en los ejercicios anteriores.

**Var:** Aquí se declaran las variables que se usaran en el algoritmo, se escribe la palabra “Var” y debajo se lista todas las variables que se usaran.

**Const**: Sección de declaración de constantes, se escribe la palabra “Const” y debajo se lista todas las constantes a utilizar con su valor correspondiente, aquí se utiliza el símbolo igual “=” en vez de la flecha.

## 3.6 Diagramas de flujo símbolos usados

Se dijo que los diagramas de flujo son representaciones graficas de un algoritmo, en ese sentido utiliza simbologías propias (Tabla 2.1), mismas que son necesarias conocer para poder desarrollar las soluciones algorítmicas. Si bien es cierto en la tabla no se indican todos los símbolos que emplea un diagrama de flujo si están los más usados, algunos otros especiales se indicaran en los siguientes temas.

**Tabla 11**

*Símbolos usados frecuentemente en diagramas de flujo*

| Símbolo | Descripción |
|---|---|
| | Terminal: Indica el inicio o fin del algoritmo |
| | Lectura (ingreso de datos) |
| | Asignación (procesos o instrucciones que realizara el algoritmo) |
| | Condición Simple (SI), Condición Múltiple (EN CASO), bucles de control |
| | Escritura (Muestra los resultados o el mensaje deseado por el programador) |
| | Flechas de dirección |
| | Conector en la misma pagina, conector en pagina diferente |

**Ejercicio 7.**

Elaborar un algoritmo que solicite 2 números y muestre el promedio de ambos.

**Solución.**

Determinamos los datos de entrada, salida y procesos.

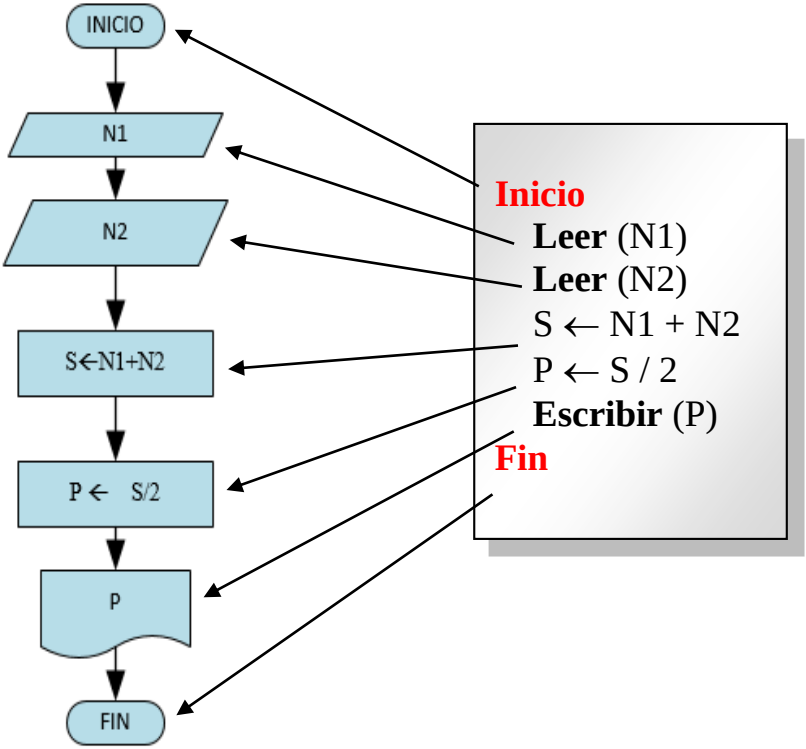
| **Entrada** | **Identificador** |
|---|---|
| Número 1 | **N1** |
| Número 2 | **N2** |
| **Salida** | |
| Promedio | **P** |
| **Otras variables** | |
| Suma parcial | **S** |

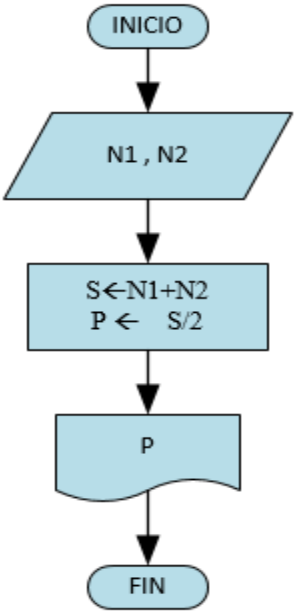
Construimos el Pseudocódigo según la estructura que se indicó, obsérvese que el nombre del algoritmo se ubica al principio, las variables se declaran separadas por comas para después indicar el tipo de dato al que pertenece ese grupo mediante los dos puntos (:), finalmente se escribe todo el proceso que se llevara a cabo dentro del segmento *Inicio. fin*. (Jordi et al., 2006)

**PSEUDOCÓDIGO**

```
ALGORITMO Promedio
Var
        N1, N2, S: Entero
        P: Real
Inicio
    Leer (N1)
    Leer (N2)
    S ← N1 + N2
    P ← S / 2
    Escribir (P)
Fin
```

El diagrama de flujo se construye reemplazando cada comando con su correspondiente símbolo gráfico, la construcción inicia desde la instrucción INICIO y termina, en FIN. En el diagrama de flujo no se declaran las variables ni constantes.(Regino, 2003)






Una forma reducida del diagrama de flujo se muestra a continuación, obsérvese que la lectura de datos de N1 y N2 se agrupa en un solo símbolo de lectura, por ser operaciones continuas y que ejecutan la misma instrucción "leer".

De forma similar para el caso de asignar valores a la variable S y P.

El código del Ejercicio 7 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio7: PROMEDIO DE DOS NÚMEROS.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese notas: ")
N1 = int( input("N1: "))
N2 = int( input("N2: "))

#Proceso
S = N1 + N2
P = S/2

#Salida
print("\nSALIDA: ")
```

```
print("-------------------------------------------------------")
print("Promedio:", P)
```

```
#Salida
-------------------------------------------------------
Ejercicio7: PROMEDIO DE DOS NÚMEROS.
-------------------------------------------------------
Ingrese notas:
N1: 12
N2: 9

SALIDA:
-------------------------------------------------------
Promedio: 10.5
```

**Ejercicio 8.**

Construya un diagrama de flujo tal que, dado como datos la base y la altura de un rectángulo, calcule el perímetro y la superficie de este.

**Solución**.

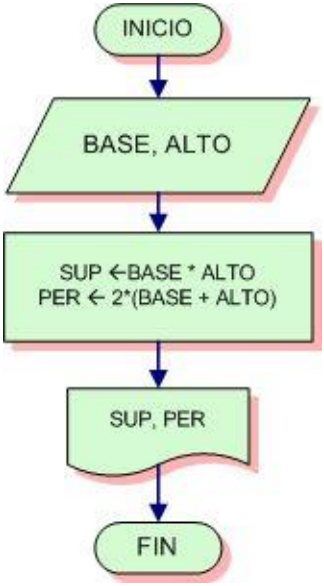
Determinamos los datos de entrada, salida y procesos.

| **Entrada** | **Identificador** |
|---|---|
| Base del rectángulo | **BASE** |
| Altura del rectángulo | **ALTO** |
| | |
| **Salida** | |
| Superficie | **SUP** |
| Perímetro | **PER** |

La superficie (SUP) se calcula multiplicando la base (BASE) por la altura (ALTO), en el caso del perímetro (PER) es 2 veces la suma de la base y la altura 2 x (BASE+ALTO)

**PSEUDOCODIGO**

**ALGORITMO** Rectángulo
**Var**
    **BASE, ALTO: Real**
    **SUP, PER: Real**
**Inicio**
    Leer (BASE, ALTO)
    SUP ← BASE * ALTO
    PER ← 2*(BASE+ALTO)
    Escribir (SUP, PER)
**Fin**




El código del Ejercicio 8 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio8: CALCULAR PERÍMETRO Y SUPERFICIE DEL RECTÁNGU
LO")
print("-------------------------------------------------------")

#Entradas
print("Ingrese Base y Alto: ")
BASE = float( input("Base: "))
ALTO = float( input("Alto: "))

#Proceso
```

```
SUP = BASE*ALTO
PER = 2*(BASE + ALTO)

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Superficie:", SUP)
print("Perímetro:", PER)
```

```
-------------------------------------------------------
Ejercicio8: CALCULAR PERÍMETRO Y SUPERFICIE DEL RECTÁNGULO
-------------------------------------------------------
Ingrese Base y Alto:
Base: 6
Alto: 8

SALIDA:
-------------------------------------------------------
Superficie: 48.0
Perímetro: 28.0
```

**Ejercicio 9.**

Construya un diagrama de flujo (DF) que resuelva un problema que tiene una gasolinera. Los dispensadores de esta registran lo que "surten" en galones, pero el precio de la gasolina está fijado en litros. El DF debe calcular e imprimir lo que hay que cobrarle al cliente.

**Solución**.

| **Entrada** | **Identificador** |
|---|---|
| Litros por galón (constante) | **LITXG** |
| Precio por litro (constante) | **PRECIOXL** |
| Cantidad surtida | **CONSU** |
| | |
| **Salida** | |

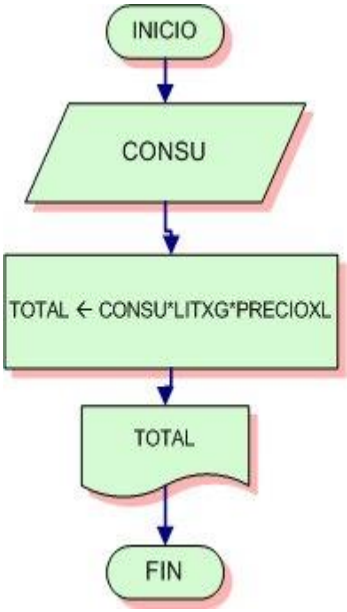
| | |
|---|---|
| Total a pagar por lo surtido | **TOTAL** |

Para calcular el total a pagar por lo surtido se debe multiplicar lo surtido (CONSU), cuyo valor está en galones, por la equivalencia de cuantos litros hay en un galón (LITXG), así obtendremos el total de litros que se surtió, a este resultado se le multiplica por el precio de cada litro (PRECIOXL).

TOTAL = COSU * LITXG* PRECIOXL

**PSEUDOCODIGO**

**ALGORITMO** Gasolinera
**Const**
    **LITXG = 3.785**
    **PRECIOXL = 4.50**
**Var**
    CONSU, TOTAL**: Real**
**Inicio**
    Leer (CONSU)
    TOTAL ← COSU * LITXG* PRECIOXL
    Escribir (TOTAL)
**Fin**




El código del Ejercicio 9 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
```

```python
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio9: GASOLINERA.")
print("-------------------------------------------------------")

#Constantes
LITXG = 3.785
PRECIOXL = 4.50

#Entradas
consu = float( input("Ingresar consumo: "))

#Proceso
total = consu*LITXG*PRECIOXL

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Total:", total)
```

```
-------------------------------------------------------
Ejercicio9: GASOLINERA.
-------------------------------------------------------
Ingresar consumo: 4

SALIDA:
-------------------------------------------------------
Total: 68.13
```

**Ejercicio 10.**

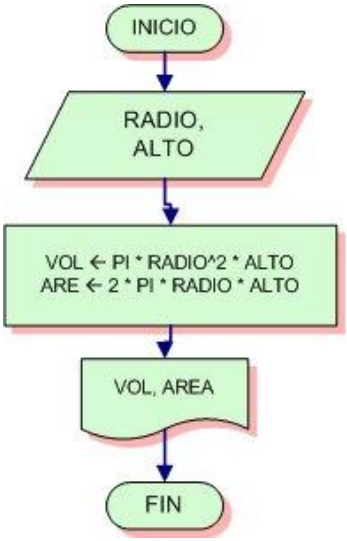
Construya un DF tal que, dado como datos el radio y la altura de un cilindro, calcule e imprima el área y su volumen.

**Solución.**

| **Entrada** | | **Identificador** |
|---|---|---|
| Valor de PI (constante) | **PI** | |
| Radio | | **RADIO** |
| Altura del cilindro | | **ALTO** |
| **Salida** | | |
| Volumen del cilindro | | **VOL** |
| Área del cilindro | | **ARE** |

**PSEUDOCODIGO**

**ALGORITMO** Cilindro
**Const**
  **PI** = 3.1416
**Var**
  RADIO, ALTO, VOL, AREA: **Real**
**Inicio**
  Leer (RADIO, ALTO)
  VOL ← PI * RADIO^2 * ALTO
  ARE ← 2*PI * RADIO (ALTO+RADIO)
  Escribir (VOL, ARE)
**Fin**




El código del Ejercicio 10 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio10: CALCULAR ÁREA Y VOLUMEN DEL CILINDRO.")
```

```
print("-------------------------------------------------------")

#Constantes
PI = 3.1416

#Entradas
print("Ingrese Radio y Alto: ")
radio = float( input("Radio: "))
alto = float ( input("Alto: "))

#Proceso
vol = PI * radio**2 * alto
are = 2*PI*radio*(radio + alto)

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Volumen:", vol)
print("área:", are)
```

```
-------------------------------------------------------
Ejercicio10: CALCULAR ÁREA Y VOLUMEN DEL CILINDRO.
-------------------------------------------------------
Ingrese Radio y Alto:
Radio: 3
Alto: 4

SALIDA:
-------------------------------------------------------
Volumen: 113.0976
área:  131.9468
```

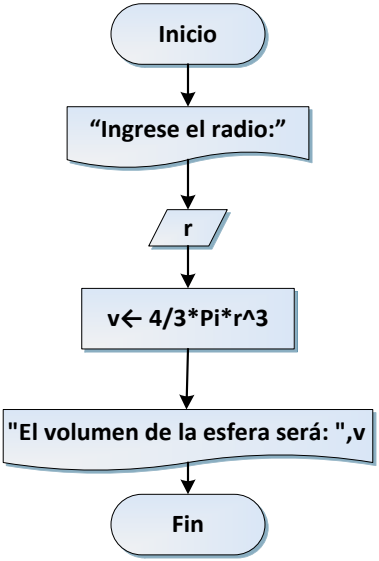
## 3.7 Ejercicios complementarios

1. Dado el radio de un círculo, calcule el volumen de la esfera correspondiente.

   Pseudocódigo.

```
Algoritmo esfera
    Var
        v,r: real   //volumen=v y radio=r
    Const
        Pi=3.1416
    Inicio
        Escribir (“Ingrese el radio:”)
        Leer (r)
        v← 4/3*Pi*r^3
            Escribir ("El volumen de la esfera es: ",v)
    Fin
```

   Diagrama de Flujo.



**Nota:** Si se desea imprimir un texto este se coloca entre comillas “”, caso contrario se tomará como variable y se imprimirá lo que contiene. Para imprimir varias cosas se separan por comas.

El código del Complementario 1 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento1: CALCULAR VOLUMEN DE LA ESFERA.")
print("-------------------------------------------------------")

#Constantes
PI = 3.1416

#Entradas
r = float( input("Ingrese Radio: "))

#Proceso
v = 4/3 * PI * r**3

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("El volumen de la esfera es:", v)
```

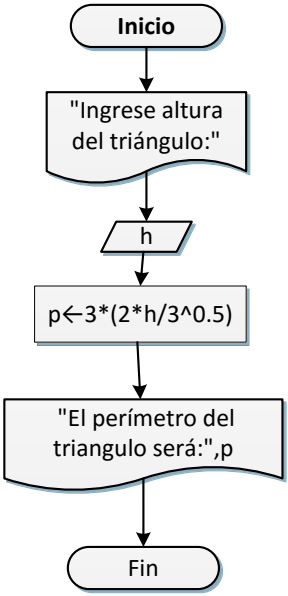
```
-------------------------------------------------------
Complemento1: CALCULAR VOLUMEN DE LA ESFERA.
-------------------------------------------------------
Ingrese Radio: 1

SALIDA:
-------------------------------------------------------
El volumen de la esfera es: 4.1888
```

2. Calcular el perímetro de un triángulo equilátero, teniendo como dato de entrada la altura de este triángulo.

Pseudocódigo.

```
Algoritmo perimetro
    Var
        p,h: real    //h=altura y p=perímetro
    Inicio
        Escribir ("Ingrese altura del triángulo:")
        Leer (h)
        p←3*(2*h/3^0.5)
        Escribir ("El perímetro del triangulo será:",p)
    Fin
```

Diagrama de Flujo.




El código del Complementario 2 en Python es el siguiente:

```python
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento2: PERÍMETRO TRIANGULO EQUILÁTERO.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese altura del triángulo: ")
h = float( input())

#Proceso
p = 3*(2*h)/3**0.5

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("El perímetro del triángulo será:", p)
```

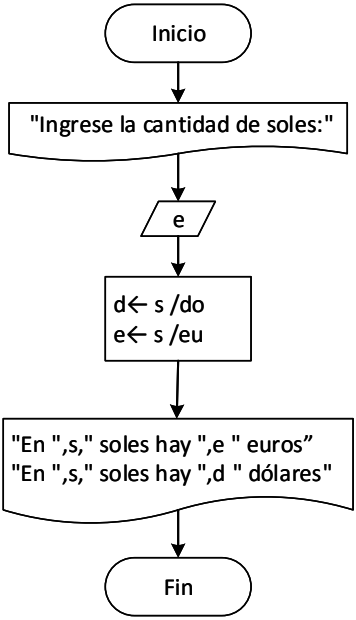
```
-------------------------------------------------------
Complemento2: PERÍMETRO TRIANGULO EQUILÁTERO.
-------------------------------------------------------
Ingrese altura del triángulo:
4

SALIDA:
-------------------------------------------------------
El perímetro del triángulo será: 13.85640646055102
```

3. Si tengo una cantidad de soles, dar su equivalente en dólares y después en euros. Se sabe que 1 dólar =3.25 soles y 1 euro=3.83 soles.

Pseudocódigo.

```
Algoritmo cambio
    Var
        d,e,s: real   //d=dólares,  e=euros  y s=soles
    Const
        eu=3.84
        do=2.82
    Inicio
        Escribir ("Ingrese la cantidad de soles:")
        Leer (s)
        d← s /do
        e← s /eu
        Escribir ("En ",s," soles hay ",e " euros")
        Escribir ("En ",s," soles hay ",d " dólares")
    Fin
```

Diagrama de Flujo.



El código del Complementario 3 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento3: CAMBIOS DE SOLES a EUROS y DÓLARES")
print("-------------------------------------------------------")

#Constantes
EU =  3.84
DO = 2.28

#Entradas
print("Ingrese la cantidad de soles:")
s = float( input())

#Proceso
d = s/DO
e = s/EU

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("En", s, "soles hay ", e, "euros")
print("En", s, "soles hay ", d, "dólares")
```

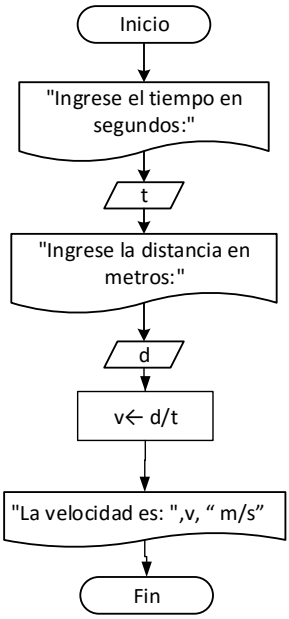
```
-------------------------------------------------------
Complemento3: CAMBIOS DE SOLES a EUROS y DÓLARES
-------------------------------------------------------
Ingrese la cantidad de soles:
50

SALIDA:
-------------------------------------------------------
En 50.0 soles hay  13.020833333333334 euros
En 50.0 soles hay  21.92982456140351 dólares
```

4. Dado el tiempo en segundos y la distancia en metros de un móvil, ingresados por teclado, calcule la velocidad correspondiente.

Pseudocódigo.

```
Algoritmo velocidad
    Var
        v,t,d: real   //v=velocidad, t=tiempo y d=distancia
    Inicio
        Escribir ("Ingrese el tiempo en segundos:")
        Leer (t)
        Escribir ("Ingrese la distancia en metros:")
        Leer (d)
        v← d/t
        Escribir ("La velocidad es: ",v, " m/s")
    Fin
```

Diagrama de Flujo.



El código del Complementario 4 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento4: CALCULAR LA VELOCIDAD DE UN MÓVIL.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese el tiempo en segundos:")
t = float( input())
print("Ingrese la distancia en metros:")
d = float( input())

#Proceso
v = d/t

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("La velocidad es:", v, "m/s")
```

```
-------------------------------------------------------
Complemento4: CALCULAR LA VELOCIDAD DE UN MÓVIL.
-------------------------------------------------------
Ingrese el tiempo en segundos:
10
Ingrese la distancia en metros:
200

SALIDA:
-------------------------------------------------------
La velocidad es: 20.0 m/s
```

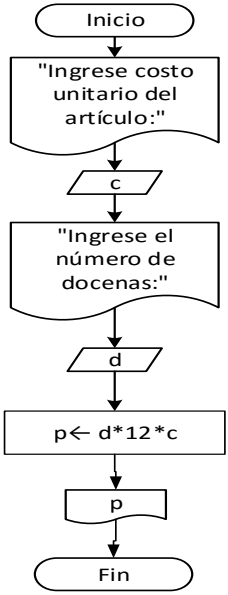
5. Calcular el monto a pagar por un artículo si se tiene como datos de entrada la cantidad de docenas que compra y el costo por unidad de este artículo.

Pseudocódigo.

```
Algoritmo precio_artículo
    Var
        p,c: real
        d:entero
    Inicio
        Escribir ("Ingrese costo unitario del artículo:")
        Leer (c)
        Escribir ("Ingrese el número de docenas:")
        Leer (d)
        p← d*12*c
        Escribir ("El precio del artículo es: ",p)
    Fin
```

**Nota:** Al número de docenas "d" se multiplica por 12 para obtener el número de unidades del artículo y poder multiplicar por el precio unitario del artículo "c".

Diagrama de Flujo.



El código del Complementario 5 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento5: CALCULAR MONTO A PAGAR.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese costo unitario del artículo:")
c = float( input())
print("Ingrese el número de docenas:")
d = int( input())

#Proceso
p = d*12 * c

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("El precio del artículo es:", p)
```

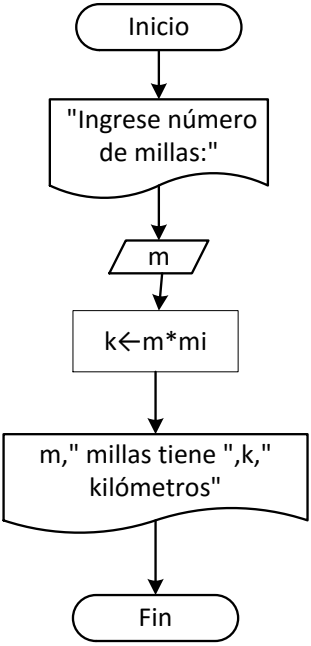
```
-------------------------------------------------------
Complemento5: CALCULAR MONTO A PAGAR.
-------------------------------------------------------
Ingrese costo unitario del artículo:
1.5
Ingrese el número de docenas:
5

SALIDA:
-------------------------------------------------------
El precio del artículo es: 90.0
```

6. Realice un diagrama de flujo que convierta millas a kilómetros. Se sabe que una milla equivale a 1.609344 kilómetros.

Pseudocódigo.

```
Algoritmo millas_kilom
    Var
        m,k: real  //m=millas y k=kilómetros
    Const
        mi=1.609344
    Inicio
        Escribir ("Ingrese número de millas:")
        Leer (m)
        k←m*mi
        Escribir (m," millas tiene ",k," kilómetros")
    Fin
```

Diagrama de Flujo.



El código del Complementario 6 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento6: Millas a Kilómetros.")
print("-------------------------------------------------------")
#Constantes
MI = 1.609344

#Entradas
print("Ingrese número de millas:")
m = float( input())

#Proceso
k = m*MI

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(m, "millas tiene", k, "kilómetros")
```

```
-------------------------------------------------------
Complemento6: Millas a Kilómetros.
-------------------------------------------------------
Ingrese número de millas:
30

SALIDA:
-------------------------------------------------------
30.0 millas tiene 48.28032 kilómetros
```

7. Si se conoce la longitud de dos de los lados de un triángulo (b y c) y el ángulo entre ellos (alfa), expresado en grados sexagesimales, la longitud del tercer lado (a) se calcula por la fórmula:

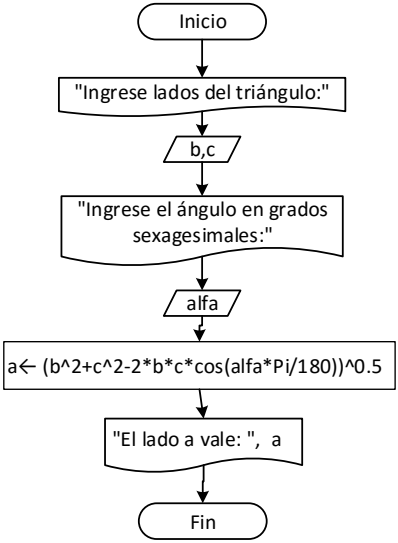
$$a^2 = b^2 + c^2 - 2bc*\cos(alfa)$$

Pseudocódigo.

El ángulo “alfa” esta dado en grados sexagesimales, para el cálculo del lado “a” hay que pasar “alfa” a radianes, transformando se tiene “**alfa*Pi/180**”

```
Algoritmo tercer_lado
   Var
      b,c,alfa: real
   Const
      Pi=3.1416
   Inicio
      Escribir ("Ingrese lados del triángulo:")
      Leer (b,c)
      Escribir ("Ingrese el ángulo en grados sexagesimales:")
      Leer (alfa)
      //formula para calcular lado “a” con “alfa” transformado
      a← (b^2+c^2-2*b*c*cos(alfa*Pi/180))^0.5
      Escribir ("El lado a es: ",  a)
   Fin
```

Diagrama de Flujo.



El código del Complementario 7 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Librerias
import math #Necearia para fórmulas matemáticas

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento7: CALCULAR EL TERCER LADO DEL TRIANGULO")
print("-------------------------------------------------------")

#Constantes
PI = 3.1416

#Entradas
print("Ingrese lados del triángulo:")
b = float( input("Lado b: "))
c = float( input("Lado c: "))
print("Ingrese el ángulo en grados sexagesimales:")
alfa = float( input())


#Proceso
#fórmula para calcular lado 'a' con alfa transformado
a = ( b**2 + c**2 - 2*b*c * math.cos( alfa*PI/180 ) )**0.50

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("El lado a es:", a)
```

```
-------------------------------------------------------
Complemento7: CALCULAR EL TERCER LADO DEL TRIANGULO
-------------------------------------------------------
Ingrese lados del triángulo:
Lado b: 4
Lado c: 3
Ingrese el ángulo en grados sexagesimales:
90

SALIDA:
```

```
-------------------------------------------------------
El lado a es: 5.000008815684476
```

8. Dado la velocidad de 2 cuerpos que se dirigen uno al encuentro de otro determinar el tiempo de encuentro si la distancia que los separa inicialmente es “D”.

Pseudocódigo.

va = velocidad del cuerpo “a”

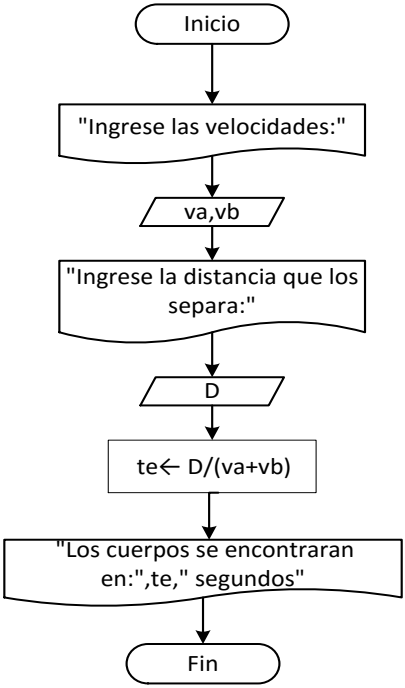
vb = velocidad del cuerpo “b”

te = tiempo de encuentro

D = distancia que separa “a” de “b”

```
Algoritmo edad
    Var
        D,va,vb,te: real
    Inicio
        Escribir ("Ingrese las velocidades:")
        Leer (va,vb)
        Escribir ("Ingrese la distancia que los separa:")
        Leer (D)
        te← D/(va+vb)
    Escribir ("Los cuerpos se encontraran en:",te," segundos")
    Fin
```

Diagrama de Flujo.




El código del Complementario 8 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento8: TIEMPO DE ENCUENTRO.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese las velocidades:")
va = float( input("Va: "))
vb = float( input("Vb: "))
print("Ingrese la distancia que los separa:")
```

```
D = float( input())

#Proceso
te = D/(va+vb)

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Los cuerpos se encontraran en:", te, "segundos")
```

```
-------------------------------------------------------
Complemento8: TIEMPO DE ENCUENTRO.
-------------------------------------------------------
Ingrese las velocidades:
Va: 5
Vb: 4
Ingrese la distancia que los separa:
50

SALIDA:
-------------------------------------------------------
Los cuerpos se encontrarán en: 5.555555555555555 segundos
```

9. Dado un ángulo en *Radianes* conviértalo a grados *Sexagesimales* y luego a *Centesimales*.

Pseudocódigo.

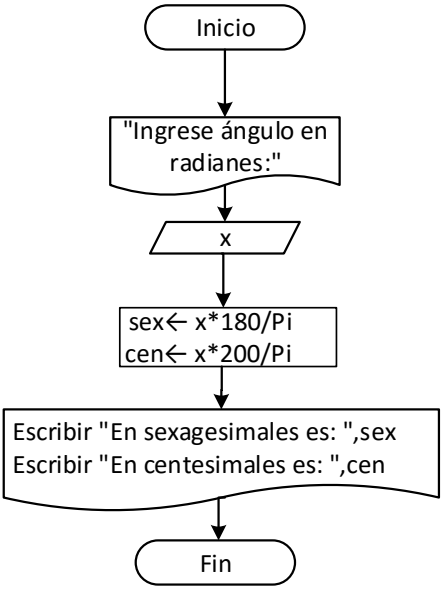
**x** = ángulo en radianes

**sex** = ángulo en grados sexagesimales

**cen** = ángulo en grados centesimales

```
Algoritmo grados
    Var
        x,sex,cen: real
    Const
        Pi=3.1416
    Inicio
        Escribir ("Ingrese ángulo en radianes:")
        Leer (x)
        sex← x*180/Pi
        cen← x*200/Pi
        Escribir ("En sexagesimales es: ",sex)
        Escribir ("en centesimales es: ",cen)
    Fin
```

Diagrama de Flujo.



El código del Complementario 9 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento9: RADIANES a SEXAGESIMALES y CENTESIMALES")
print("-------------------------------------------------------")

#Constantes
PI = 3.1416

#Entradas
print("Ingrese ángulo en radianes:")
x = float( input())

sex = x*180/PI
cen = x*200/PI

#Proceso

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("En sexagesimales es:", sex)
print("En centesimales es:", cen)
```

```
-------------------------------------------------------
Complemento9: RADIANES a SEXAGESIMALES y CENTESIMALES
-------------------------------------------------------
Ingrese ángulo en radianes:
1

SALIDA:
-------------------------------------------------------
En sexagesimales es: 57.29564553093965
En centesimales es: 63.66182836771072
```

10. Determine la distancia entre dos puntos en el espacio.

    (x1,y1, z1) y (x2, y2, z2)

Pseudocódigo.

La fórmula para hallar la distancia entre 2 puntos en el espacio es

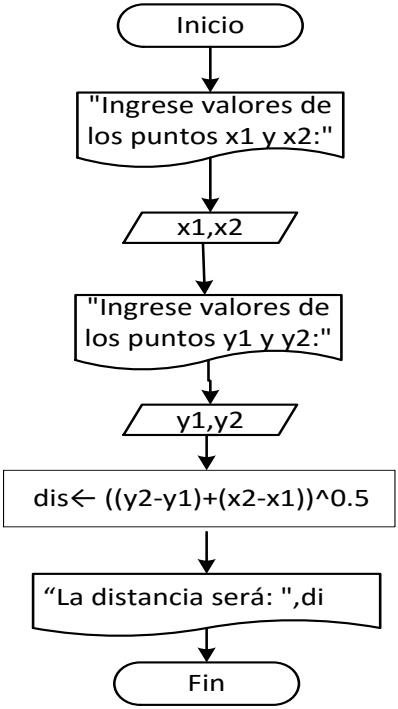
$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} = d$$

Punto 1: (x1, y1, z1)  Punto 2: (x2, y2, z2)

dis = distancia entre puntos

```
Algoritmo distancia
    Var
        dis,x1,y1,z1,x2,y2,z2: real
    Inicio
        Escribir ("Ingrese valores de los puntos x1 ,y1 y z1")
        Leer (x1,y1,z1)
        Escribir ("Ingrese valores de los puntos x2,y2 y z2")
        Leer (x2, y2, z2)
        dis← ((z2-z1)^2+(y2-y1)^2+(x2-x1)^2)^0.5
        Escribir ("La distancia es: ",dis)
    Fin
```

Diagrama de Flujo:




El código del Complementario 10 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento10: DISTANCIA DE DOS PUNTOS en 3D.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese valores del punto A(x1, y1 y z1): ")
x1 = float( input("x1: "))
y1 = float( input("y1: "))
z1 = float( input("z1: "))
```

```
print("Ingrese valores del punto B(x2, y2 y z2): ")
x2 = float( input("x2: "))
y2 = float( input("y2: "))
z2 = float( input("z2: "))

#Proceso
dis = ( (z2-z1)**2 + (y2-y1)**2 + (x2-x1)**2 )**0.5

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("La distancia es:", dis)
```

```
-------------------------------------------------------
Complemento10: DISTANCIA DE DOS PUNTOS en 3D.
-------------------------------------------------------
Ingrese valores del punto A(x1, y1 y z1):
x1: 0
y1: 0
z1: 0
Ingrese valores del punto B(x2, y2 y z2):
x2: 1
y2: 1
z2: 0
SALIDA:
-------------------------------------------------------
La distancia es: 1.4142135623730951
```

# 4

## Estructuras Selectivas

Las estructuras selectivas conocidas también como lógicas selectivas se encuentran en la solución algorítmica de casi todo tipo de problemas. La utilizamos cuando en el desarrollo de la solución de un problema debemos tomar una *decisión*, para establecer un proceso o **señalar un camino alternativo a seguir**.

Esta toma de decisión expresada en un rombo en el diagrama de flujo se basa en la evaluación de una o más condiciones que nos señalaran como alternativa el camino a seguir.

Las estructuras algorítmicas selectivas se clasifican en tres:

- Simple
- Doble
- Múltiple

### 4.1 Estructura selectiva simple

Se identifican porque están compuestos únicamente de un solo camino para la condición. La estructura selectiva de tipo simple evalúa esta *condición*, si el resultado es verdadero (*entonces*) ejecuta el conjunto de acciones asociadas a él (operaciones o acciones), caso contrario, si es falso, no hace nada.(*Lenguaje Logo Iii. Explorando la Programación*, s. f.)

Su representación en pseudocódigo es:

**Sí** <condición > **entonces**

<Operaciones o acciones>

**…**

**fin sí**

Su representación en diagrama de flujo se muestra en la figura siguiente

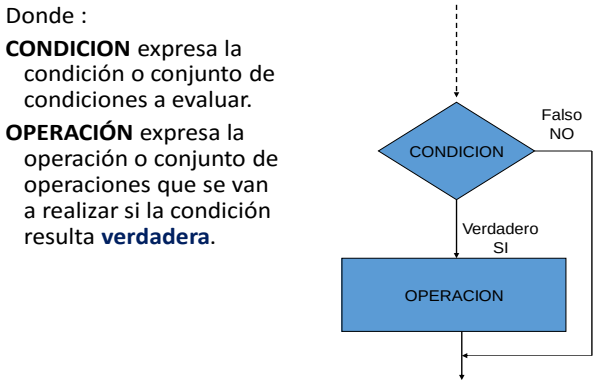



***Figura 22.*** Diagrama de flujo de una estructura condicional simple

En el siguiente ejercicio, se aplica la estructura condicional simple.

**Ejemplo**: Dado un número entero, evalué, si resulta ser menor que 100 mostrar un mensaje que diga "número pequeño".

**Solución**.

Definiendo los datos de entrada y la salida correspondiente:

| **Entrada** | **Identificador** |
|---|---|
| Numero entero | **NUM** |
| **Salida** | |
| Mensaje impreso | **"Numero pequeño"** |

De lo planteado se deduce que después de ingresar el número entero NUM se debe realizar una evaluación de este valor, evaluar **condición**, en el caso que el número sea menor que 100 **entonces** debe imprimirse un mensaje "número pequeño", caso contrario no se hace nada.

El pseudocódigo seria así:

```
1.  Algoritmo Número
2.    Var
3.        x: Entero
4.    Inicio
5.        Escribir ("Ingrese un número")
6.        Leer (x)
7.        Si (x < 100) entonces
8.            Escribir ("NUMERO PEQUEÑO")
9.        Fin si
10.   Fin
```

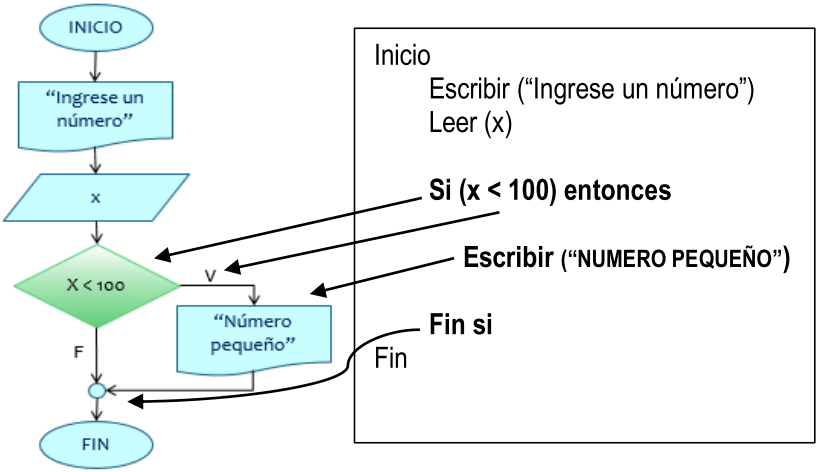
Las instrucciones que se encuentran desde la línea 7 a la 9 comprende el uso de la estructura condicional simple. Analizando:

**Línea 7**: Habiendo ingresado un valor entero en la variable "**x**" se procede a evaluar ese valor en la condición **Si,** preguntando ¿**x** es menor que 100?, si el resultado es *verdadero* se procede con **entonces** y se pasa a la línea 8, caso contrario (resultado es *falso*) se salta hasta la línea 9 **Fin si**.

**Línea 8**: Se procede con la instrucción **escribir** y se imprime el texto "NUMERO PEQUEÑO", luego se pasa a la línea siguiente, línea 9.

**Línea 9**: Esta línea indica la finalización del condicional simple, siempre debe indicarse el final del condicional con **Fin sí.**

El diagrama de flujo correspondiente se muestra a continuación.




El código de este ejemplo en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("EVALUAR SI NUMERO MENOR A 100.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese un número: ")
x = int( input())

#Proceso
if x < 100 :
```

```
#NOTA: para inidicar que tdodo esta dentro
#de la estructura IF hay que hacer uso de la
#indentación con ayuda de la tecla TABULADOR
print("Número Pequeño")
```

```
-----------------------------------------------------
EVALUAR SI NUMERO MENOR A 100.
-----------------------------------------------------
Ingrese un número:
40
Número Pequeño
```

## 4.2 Estructura selectiva doble

Son estructuras lógicas que permiten controlar la ejecución de varias acciones y se utilizan cuando se tienen dos opciones de acción, se evalúa la **condición** y si es verdad se ejecuta el conjunto de acciones asociadas a la parte **entonces**, si es falso se ejecuta el conjunto de acciones asociadas a la parte **sino**.(Regino, 2003)

**Si** <condición> **entonces**
< acciones tipo 1 >
**sino**
< acciones tipo 2 >

**Fin si**

Su representación en diagrama de flujo se muestra en la figura 23.

Donde :

**CONDICION** expresa la condición o conjunto de condiciones a evaluar.

**OPERACIÓN1** expresa la operación o conjunto de operaciones que se van a realizar si el resultado de la condición es **verdadero**.

**OPERACIÓN2** expresa la operación o conjunto de operaciones que se van a realizar si la el resultado de la condición es **falso**.

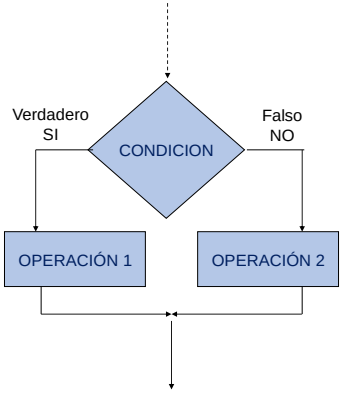



***Figura 23.*** Diagrama de flujo de una estructura condicional doble

En el siguiente ejercicio, se aplica la estructura condicional doble.

**Ejemplo**: Ingresar 2 números enteros diferentes e imprimir el menor de ellos con el mensaje "El menor es: --".

**Solución**.

Los datos de entrada y salida correspondientes son:

| **Entrada** | **Identificador** |
|---|---|
| Numero X | x |
| Numero Y | y |
| **Salida** | |
| Mensaje impreso | **"El menor es" #** |

Dados los números x e y, si **x** resulta siendo mayor que **y** se imprime el mensaje **"El menor es " #,** el símbolo "#" se reemplaza por el valor de **x,** en el caso que **y** sea mayor se

imprimirá el mismo mensaje pero con el valor de **y.** El pseudocódigo es el siguiente:

1. Algoritmo Menor
2. Var
3. x, y : Entero
4. Inicio
5. Escribir (“Ingrese 2 números”)
6. Leer (x, y)
7. **Si (x > y) entonces**
8. **Escribir (“El menor es : ”, y)**
9. **sino**
10. **Escribir (“El menor es : ”, x)**
11. **Fin si**
12. Fin

Las instrucciones que se encuentran desde la línea 7 a la 11 comprende el uso de la estructura condicional doble. Analizando:

**Línea 7**: Habiendo ingresado un valor entero en la variable “x” y otro en la variable “y” se procede a comparar ambos, en la condición **Si**, preguntando ¿x es mayor que y?, si el resultado es verdadero se procede con **entonces** y se pasa a la línea 8, caso contrario (resultado es falso) se salta hasta **sino** la línea 9.

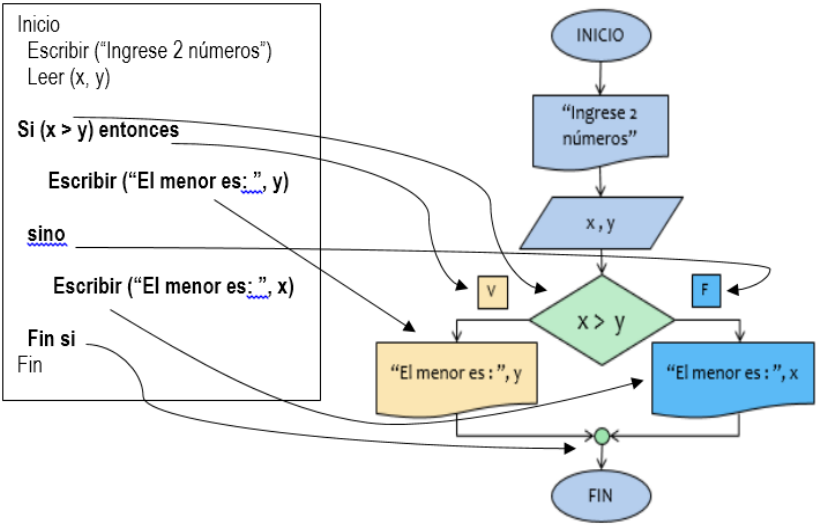
**Línea 8**: Se procede con la instrucción escribir y se imprime el texto “El menor es:” **x**, luego se pasa a la línea 11.

**Línea 9**: Se llega aquí si el resultado de la comparación x>y es **falso, sino** es el comienzo de esta sección del condicional doble si, de esta línea se pasa a la línea 10.

**Línea 10**: Se imprime el mensaje “El menor es:” **y**, luego se pasa a la línea 11.

**Línea 11**: Aquí termina el condicional doble.

El diagrama de flujo correspondiente se muestra a continuación.




El código de este ejemplo en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejemplo2: IMPRIMIR EL MENOR DE DOS NÚMEROS.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese 2 números: ")
x = int( input("Primer Número: "))
y = int( input("Segundo Número: "))

#Salida
```

```
print("\nSALIDA: ")
print("-------------------------------------------------------")

#NOTA: No olvidarse de las indentación.
if x > y :
   #Esto esta dentro del IF
   print("El menor es:", y)
else:
   #Esto esta dentro del ELSE
   print("El menor es:", x)
```

```
-------------------------------------------------------
Ejemplo2: IMPRIMIR EL MENOR DE DOS NÚMEROS.
-------------------------------------------------------
Ingrese 2 números:
Primer Número: 10
Segundo Número: 20

SALIDA:
-------------------------------------------------------
El menor es: 10
```

## 4.3 Estructura selectiva múltiple

En este tipo de estructura se evalúa una variable **selector** que puede tomar diversos valores. Según el valor que la variable selector tome en cada momento se ejecutan las acciones correspondientes al valor. En realidad, equivale a un conjunto de condiciones anidadas. En un lenguaje de programación se le conoce como Case, Select case o Switch.

Si <selector> igual

<Valor1>: <operación 1>

<valor2>: <operación 2>

.....

**[<En otro caso>:** <operacin n+1>]

fin selector

**Nota:** Dentro del selector múltiple el término **<En otro caso>** se ejecuta cuando la expresión no toma ninguno de los valores que aparecen antes.

Su representación en diagrama de flujo se muestra en la figura 24.

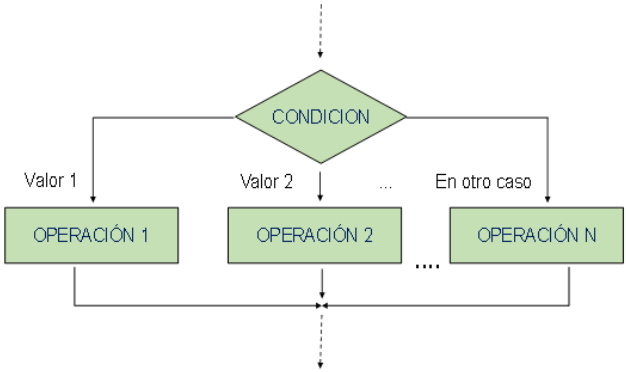



*Figura 24. Diagrama de flujo de una estructura condicional múltiple*

El código de una estructura de condición múltiple en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#En Python no existe la selección multiple o SWITCH
#para simular la funcionalidad se usa un Diccionario
switcher = {
    1: "Enero",
    2: "Febrero",
    3: "Marzo",
    4: "Abril",
    5: "Mayo",
    6: "Junio",
    7: "Julio",
```

```
    8: "Agosto",
    9: "Septiembre",
    10: "Octubre",
    11: "Noviembre",
    12: "Diciembre"
}
argument = int( input("Ingrese número de mes: "))

#para acceder a los elementos se hace uso de la funcion .get del diccionario
# diccionario.get(AlgunArgumento, mensaje por Defecto)

nombreDeMes = switcher.get(argument, "Mes invalido")
print(nombreDeMes)
```

```
Ingrese número de mes: 5
Mayo

Ingrese número de mes: 15
Mes invalido
```

**Ejercicio 11**

Construir un diagrama de flujo, tal que dado como dato la calificación de un alumno, escriba “aprobado” en caso de que esa calificación sea mayor a 10.

**Solución.**

**Dato:** CAL (variable real que representa la calificación del alumno).

**PSEUDOCODIGO**

**ALGORITMO** Resultado
**Var**
    **CAL: Real**
**Inicio**
    Leer (CAL)
    **Si CAL > 10 entonces**
        **Escribir ("Aprobado")**
    **Fin_si**
**Fin**

El código del ejercicio 11 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio11: VERIFICAR SI UN ALUMNO ESTA APROBADO")
print("-------------------------------------------------------")

#Entradas
CAL = float( input("Ingrese Calificación: "))

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")

if CAL > 10 :
   print("Aprobado")
```

```
-------------------------------------------------------
Ejercicio11: VERIFICAR SI UN ALUMNO ESTA APROBADO
-------------------------------------------------------
```

```
Ingrese Calificación: 12

SALIDA:
-------------------------------------------------------
Aprobado
```

**Ejercicio 12**

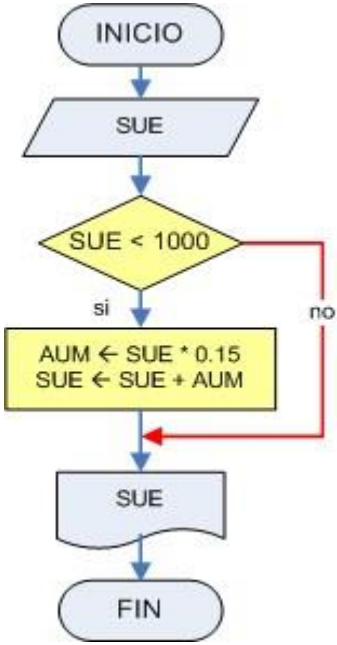
Dado el sueldo de un trabajador, aplique un aumento del 15% si su sueldo es inferior a $1000. Imprima el sueldo que percibirá.

**Solución**.

**Dato**: SUE (Variable de tipo Real, representa el sueldo del trabajador)

**PSEUDOCODIGO**

```
ALGORITMO Resultado
Var
   SUE: Real
Inicio
   Leer (SUE)
   Si SUE < 1000 entonces
         AUM ← SUE * 0.15
         SUE ← SUE + AUM
   Fin_si
   Escribir (SUE )
Fin
```




El código del ejercicio 12 en Python es el siguiente:

```python
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio12: SUELDO A PERCIBIR")
print("-------------------------------------------------------")

#Entradas
SUE = float( input("Ingrese Sueldo: "))

#Proceso
if SUE < 1000:
   AUM = SUE*0.15
   SUE = SUE + AUM

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("El sueldo es:", SUE)
```

```
-------------------------------------------------------
Ejercicio12: SUELDO A PERCIBIR
-------------------------------------------------------
Ingrese Sueldo: 500

SALIDA:
-------------------------------------------------------
El sueldo es: 575.0


Ingrese Sueldo: 1200

SALIDA:
-------------------------------------------------------
El sueldo es: 1200.0
```

## Ejercicio 13

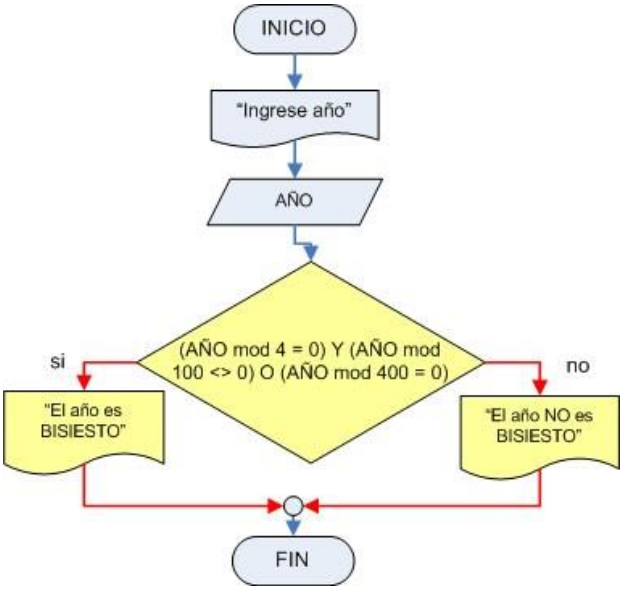
Desarrolle un algoritmo para determinar si un año leído por teclado es o no bisiesto.

**Solución.**

**Dato:** AÑO (Variable tipo Entero, almacena el año a evaluar). Un año bisiesto es aquel múltiplo de 4 pero no de 100 o si es múltiplo de 400.

**PSEUDOCODIGO**

```
ALGORITMO Resultado
Var
    año: Entero
Inicio
    Escribir ("Ingrese año")
    Leer (año)
    Si (año mod 4) Y (año mod 100 <> 0) O (año mod 400 = 0) entonces
        Escribir ("El año es BISIESTO")
    sino
        Escribir ("El año NO es BISIESTO")
    Fin_si
Fin
```




El código del ejercicio 13 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("EjercicioN: VERIFICAR SI EL AÑO ES BISIESTO.")
print("-------------------------------------------------------")

#Entradas
anio = int( input("Ingrese año: "))

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
if (anio % 400 == 0) or (anio % 4 == 0) and (anio % 100 != 0):
   print("El año es BISIESTO")
else:
   print("El año NO es BISIESTO")
```

```
-------------------------------------------------------
EjercicioN: VERIFICAR SI EL AÑO ES BISIESTO.
-------------------------------------------------------
Ingrese año: 2000

SALIDA:
-------------------------------------------------------
El año es BISIESTO

Ingrese año: 1800

SALIDA:
-------------------------------------------------------
El año NO es BISIESTO
```

**Ejercicio 14**

Construya un D.F., tal que, dado como datos 2 variables de tipo entero, obtenga el resultado de la siguiente función:

$$VAL = \left\{ \begin{array}{ll} 100 * V & si_NUM = 1 \\ 100 \wedge V & si_NUM = 2 \\ 100 / V & si_NUM = 3 \\ 0 & para_cualquier_otro_valor \end{array} \right\}$$

**Solución.**

**Datos:** NUM, V

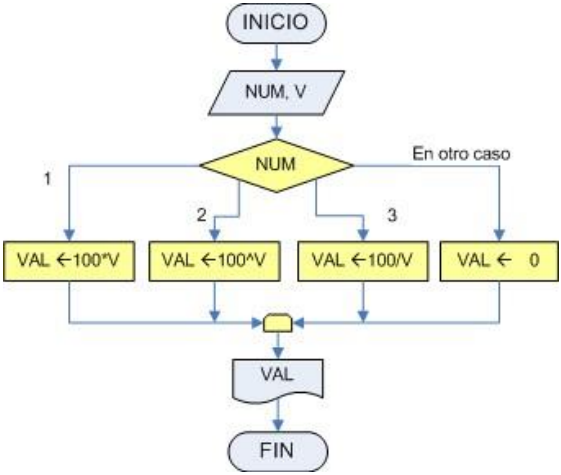
**Donde:**

NUM: Es una variable de tipo entero, representa el tipo de calculo que se va a realizar.

V: Variable de tipo entero que se utiliza para el cálculo de la función.

**PSEUDOCODIGO**

**ALGORITMO** Función
**Var**
    **NUM, V: Entero**
**Inicio**
    Leer (NUM, V)
    **Si NUM igual**
        1: VAL ← 100*V
        2: VAL ← 100^V
        3: VAL ← 100/V
        **En otro caso**: VAL ← 0
    **Fin Selector**
    Escribir (VAL)
**Fin**




El código del ejercicio 14 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
```

```
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio14: FUNCIÓN.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese valores: ")
NUM = int( input("Tipo de Calculo: "))
V = int( input("Ingrese V: "))

#Proceso
#usando un diccionario
Funcion = {
   1 : 100*V,
   2 : 100**V,
   3 : 100/V
}

#DICCIONARIO.get(ElementoABuscarEnDiccionario, PorDefecto)
#porDefecto: En caso de que el elemnto no se encuentre.
VAL = Funcion.get(NUM, 0)

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(VAL)
```

```
-------------------------------------------------------
Ejercicio14: FUNCIÓN.
-------------------------------------------------------
Ingrese valores:
Tipo de Calculo: 1
Ingrese V: 3

SALIDA:
-------------------------------------------------------
300

Ingrese valores:
Tipo de Calculo: 2
Ingrese V: 3
```

```
SALIDA:
-------------------------------------------------------
1000000

Ingrese valores:
Tipo de Calculo: 3
Ingrese V: 3

SALIDA:
-------------------------------------------------------
33.333333333333336

Ingrese valores:
Tipo de Calculo: 4
Ingrese V: 3

SALIDA:
-------------------------------------------------------
0
```

**Ejercicio 15**

Se desea leer por teclado un número comprendido entre 0 y 10 (inclusive) y se desea visualizar si el número es par o impar.

**Solución.**

**Dato:** NUM variable de tipo entero, almacena el número leído por teclado.

**PSEUDOCODIGO**

**ALGORITMO** Par_Impar
**Var**
    **NUM: Entero**
**Inicio**
    Escribir (“Ingrese número”)
    Leer (NUM)
    **Si NUM igual**
        1,3,5,7,9 : Escribir (“Impar”)
        0,2,4,6,8,10 : Escribir(“Par”)
    **Fin Selector**
**Fin**

El código del ejercicio 15 en Python es el siguiente:

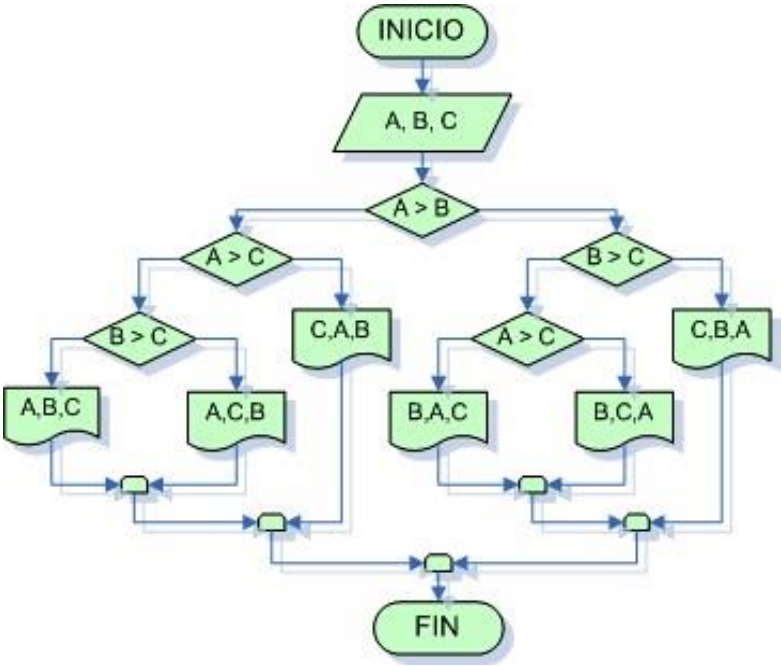
```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio15: PAR O IMPAR.")
print("-------------------------------------------------------")

#Entradas
NUM = int( input("Ingrese número:" ))

#Proceso
#Usando un diccionario
par_Impar = {
   1 : 'Impar',
   3 : 'Impar',
   5 : 'Impar',
   7 : 'Impar',
   9 : 'Impar',
   2 : 'Par',
   4 : 'Par',
   6 : 'Par',
   8 : 'Par',
   10 : 'Par'
}
```

```
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(par_Impar.get(NUM, "Numero fuera de Rango"))
```

```
-------------------------------------------------------
Ejercicio15: PAR O IMPAR.
-------------------------------------------------------
Ingrese número: 10

SALIDA:
-------------------------------------------------------
Par

Ingrese número: 5

SALIDA:
-------------------------------------------------------
Impar

Ingrese número: 11

SALIDA:
-------------------------------------------------------
Numero fuera de Rango
```

## Ejercicio 16

Dado el siguiente diagrama de flujo determine la salida obtenida para los siguientes datos: A=2, B=90 y C=45.




El código del ejercicio 16 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio16: DETERMINA LA SALIDA.")
print("-------------------------------------------------------")

#Entradas
A = int( input("Ingrese A: "))
B = int( input("Ingrese B: "))
C = int( input("Ingrese C: "))

print("\nSALIDA: ")
print("-------------------------------------------------------")
```

```
#Proceso
if A > B :
   if A > C:
      if B > C:
         print(A, B, C)
      else:
         print(A,C,B)
   else:
      print(C,A,B)
else:
   if B > C :
      if A > C :
         print(B,A,C)
      else:
         print(B,C,A)
   else:
      print(C,B,A)
```

```
-------------------------------------------------------
Ejercicio16: DETERMINA LA SALIDA.
-------------------------------------------------------
Ingrese A: 2
Ingrese B: 90
Ingrese C: 45

SALIDA:
-------------------------------------------------------
90 45 2
```

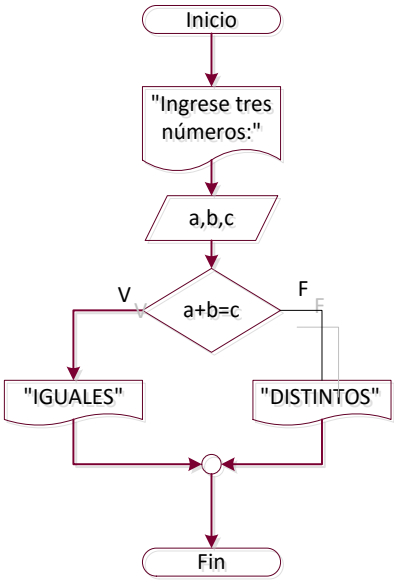
## 4.4 Ejercicios complementarios

1. Construya un algoritmo que, dados tres números, muestre el mensaje “IGUALES” si la suma de los dos primeros es igual al otro número y el mensaje “DISTINTOS” en caso contrario.

Pseudocódigo.

```
Algoritmo igdis
    Var
        a,b,c: entero
    Inicio
        Escribir ("Ingrese tres números:")
        Leer (a,b,c)
        Si (a+b=c) entonces
            Escribir ("IGUALES")
        sino
            Escribir ("DISTINTOS")
        Fin Si
    Fin
```

Diagrama de Flujo.



El código del complementario 1 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento1: IGUALES O DISTINTOS.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese tres números")
a = int( input("Ingrese a: "))
b = int( input("Ingrese b: "))
c = int( input("Ingrese c: "))

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")

if a + b == c :
   print("IGUALES")
else:
   print("DISTINTOS")
```

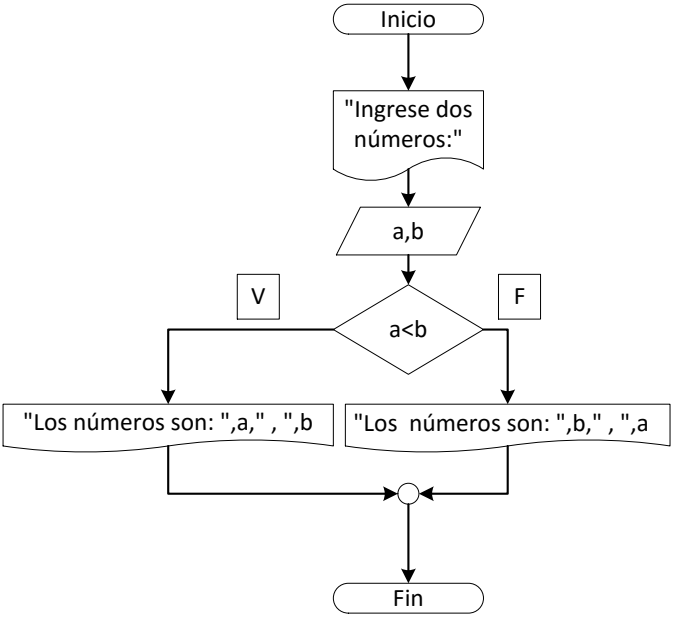
```
-------------------------------------------------------
Complemento1: IGUALES O DISTINTOS.
-------------------------------------------------------
Ingrese tres números
Ingrese a: 10
Ingrese b: 5
Ingrese c: 15

SALIDA:
-------------------------------------------------------
IGUALES
```

2. Algoritmo, que dado dos números “a” y “b”, muestre sus valores en orden de menor a mayor.

Pseudocódigo.

```
Algoritmo orden_creciente
    Var
        a,b: entero
    Inicio
        Escribir ("Ingrese dos números:")
        Leer a,b
        Si (a<b) entonces
            Escribir ("Los números son: ",a," , ",b)
        sino
            Escribir ("Los  números son: ",b," , ",a)
        Fin Si
    Fin
```

Diagrama de Flujo.



El código del complementario 2 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento2: MOSTRAR DE MENOR A MAYOR.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese dos números: ")
a = int( input("Ingrese a: "))
b = int( input("Ingrese b: "))

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")

if a < b:
   print("Los números son:", a, b)
else:
   print("Los números son:", b, a)
```

```
-------------------------------------------------------
Complemento2: MOSTRAR DE MENOR A MAYOR.
-------------------------------------------------------
Ingrese dos números:
Ingrese a: 50
Ingrese b: 40

SALIDA:
-------------------------------------------------------
Los números son: 40 50
```

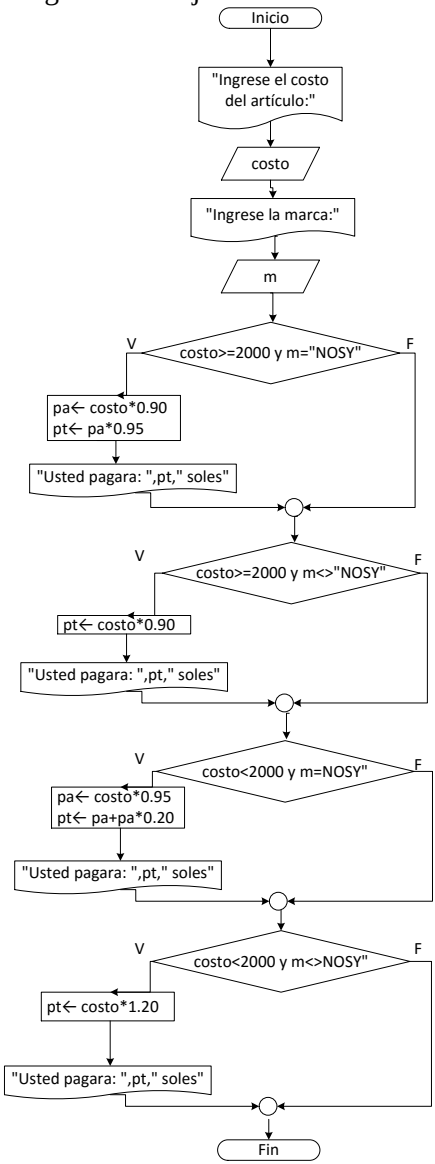
3. Un proveedor de estéreos ofrece un descuento del 10% sobre el precio sin IGV, de algún aparato si esta cuesta $2000 o más. Además, independientemente de esto, ofrece un 5% de descuento adicional sobre el precio si la marca es “NOSY”.

Determinar cuánto pagará, con IGV incluido, un cliente cualquiera por la compra de su aparato. IGV = 20%

Pseudocódigo.

```
Algoritmo costoart
    Var
        pa,pt,costo: real
        m:cadena
    Inicio
        Escribir ("Ingrese el costo del artículo:")
        Leer costo
        Escribir ("Ingrese la marca:")
        Leer (m)
        Si (costo>=2000) y (m="NOSY")  entonces
            pa← costo*0.90
            pt← pa*0.95
            Escribir ("Usted pagara: ",pt," soles")
        Fin Si
        Si (costo>=2000) y (m<>"NOSY")  entonces
            pt← costo*0.90
            Escribir ("Usted pagara: ",pt," soles")
        Fin Si
        Si (costo<2000) y (m="NOSY")  entonces
            pa← costo*0.95
            pt← pa+pa*0.20
            Escribir ("Usted pagara: ",pt," soles")
        Fin Si
        Si (costo<2000) y (m<>"NOSY")  entonces
            pt← costo*1.20
            Escribir ("Usted pagara: ",pt," soles")
        Fin Si
    Fin
```

Diagrama de flujo.



El código del complementario 3 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento3: COSTO DE ARTICULO.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese el Costo del artículo: ")
costo = float( input())

print("Ingrese la marca: ")
#m no necesita conversión ya que la entradas son Texto
m = input()

#Proceso
if costo >= 2000 and m == "NOSY" :
   pa = costo*0.90
   pt = pa*0.95

elif costo >= 200 and m != "NOSY" :
   pt = costo*0.90

elif costo < 2000 and m == "NOSY" :
   pa = costo*0.95
   pt = pa + pa*0.20

elif costo < 2000 and m != "NOSY" :
   pa = costo*1.20

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Usted pagara:", pt, "soles")
```

```
Complemento3: COSTO DE ARTICULO.
-------------------------------------------------------
Ingrese el Costo del artículo:
150
Ingrese la marca:
```

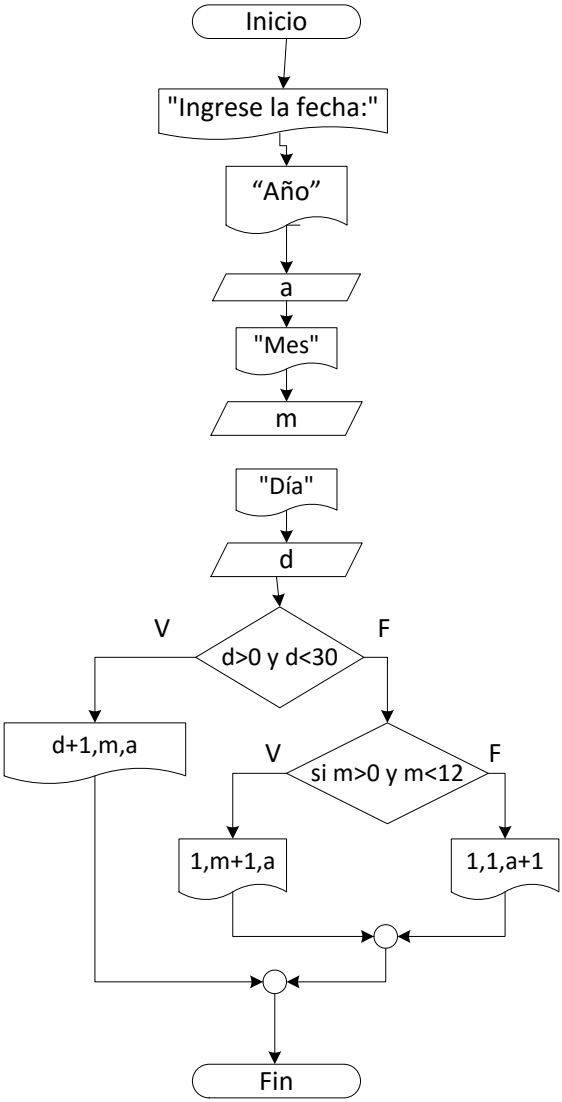
```
NOSY

SALIDA:
-------------------------------------------------------
Usted pagara: 171.0 soles
```

4. Algoritmo, que dada una fecha del año 2000 (representada por el día, el mes y el año en formato numérico dd/mm/aaaa), calcule el día siguiente. Asuma que el mes tiene 30 días.

   Pseudocódigo.

```
Algoritmo fecha
    Var
        a,m,d:entero
    Inicio
        Escribir ("Ingrese la fecha")
        Escribir ("Año")
        Leer (a)
        Escribir ("Mes")
        Leer (m)
        Escribir ("Día")
        Leer (d)
        si d>0 y d<30 Entonces
            Escribir (d+1)
            Escribir (m)
            Escribir (a)
        Sino
            si m>0 y m<12 Entonces
                Escribir ("1")
                Escribir (m+1)
                Escribir (a)
            Sino
                Escribir ("1")
                Escribir ("1")
                Escribir (a+1)
            Fin Si
        Fin Si
    Fin
```

Diagrama de Flujo.



El código del complementario 4 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento4: CALCULA EL DÍA SIGUIENTE.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese la fecha: ")
a = int( input("Año: "))
m = int( input("Mes: "))
d = int( input("Día: "))

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
if d > 0 and d < 30 :
   print("Mañana es:", d+1, m, a)
else:
   if m > 0 and m < 12 :
      print("Mañana es:", 1, m+1, a)
   else:
      print("Mañana es:", 1, 1, a+1)
```

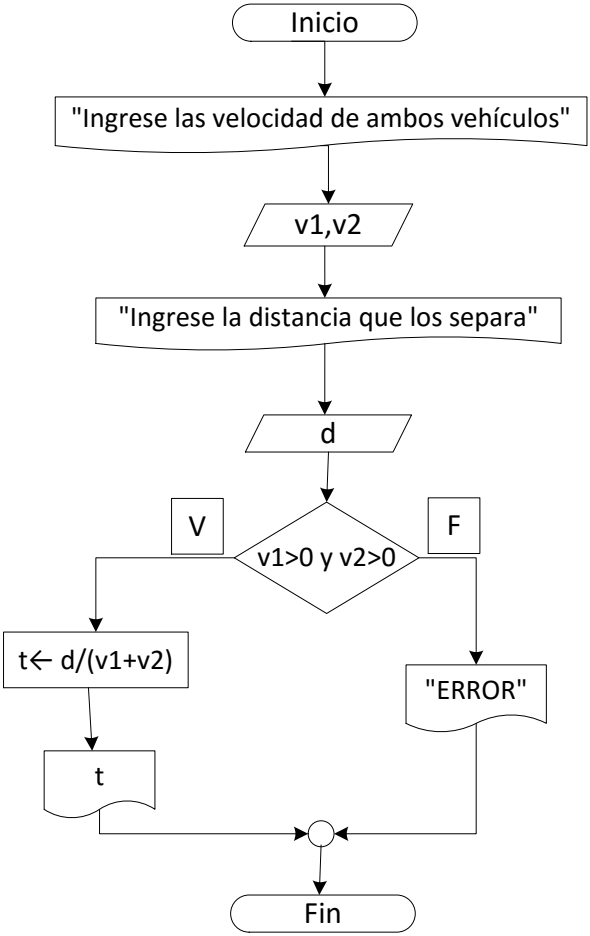
```
-------------------------------------------------------
Complemento4: CALCULA EL DÍA SIGUIENTE.
-------------------------------------------------------
Ingrese la fecha:
Año: 2019
Mes: 11
Día: 24

SALIDA:
-------------------------------------------------------
Mañana es: 25 11 2019
```

5. Algoritmo, que calcule el tiempo de encuentro de 2 vehículos que van en sentido opuesto, teniendo como datos la distancia inicial que los separa y la velocidad de cada uno.

Pseudocódigo.

```
Algoritmo tiempodeenc
    Var
        v1,v2,t,d:real
    Inicio
        Escribir ("Ingrese las velocidad de ambos vehículos")
        Leer (v1)
        Leer (v2)
        Escribir ("Ingrese la distancia que los separa")
        Leer (d)
        si v1>0 y v2>0 Entonces
            t← d/(v1+v2)
            Escribir (t)
        Sino
            Escribir ("ERROR")
        Fin Si
    Fin
```

Diagrama de Flujo.




El código del complementario 5 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("Complemento5: CALCULAR TIEMPO DE ENCUENTRO.")
print("-----------------------------------------------------")
```

```
#Entradas
print("Ingrese las velocidades de ambos vehículos: ")
v1 = float( input("V1: "))
v2 = float( input("V2: "))
print("Ingrese la distancia que los separa: ")
d = float( input("Distancia: "))

#Proceso

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")

if v1 > 0 and v2 > 0 :
   t = d/(v1+v2)
   print(t, "segundos")
else:
   print("ERROR")
```

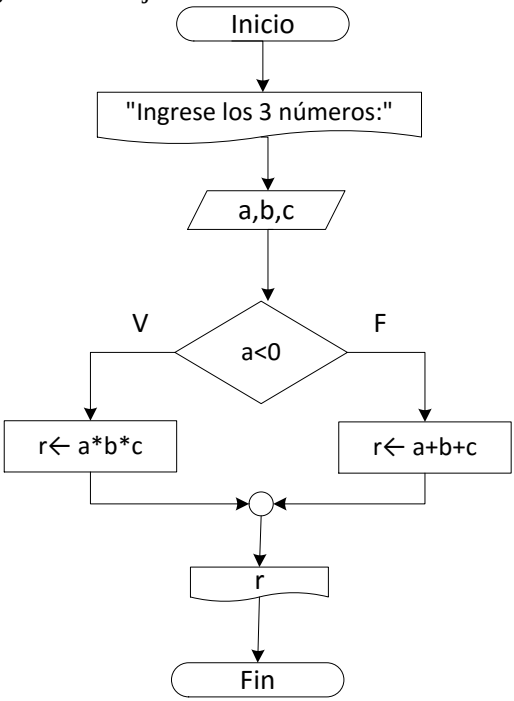
```
-------------------------------------------------------
Complemento5: CALCULAR TIEMPO DE ENCUENTRO.
-------------------------------------------------------
Ingrese las velocidades de ambos vehículos:
V1: 5.5
V2: 8
Ingrese la distancia que los separa:
Distancia: 400

SALIDA:
-------------------------------------------------------
29.62962962962963 segundos
```

6. Leer tres números enteros y, si el primero de ellos es negativo, calcular el producto de los tres, en caso contrario calcular la suma de ellos.

Pseudocódigo.

```
Algoritmo numeros
    Var
        a,b,c,r:entero
    Inicio
        Escribir ("Ingrese los 3 números:")
        Leer (a)
        Leer (b)
        Leer (c)
        si a<0 Entonces
            r← a*b*c
        Sino
            r← a+b+c
        Fin Si
        Escribir (r)
    Fin
```

Diagrama de Flujo.



El código del complementario 6 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento6: NÚMEROS, EL PRIMERO DECIDE.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese los 3 números:")
a = int( input("a: "))
b = int( input("b: "))
c = int( input("c: "))

#Proceso
if a < 0 :
   r = a*b*c
else:
   r = a + b + c

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(r)
```

```
-------------------------------------------------------
Complemento6: NÚMEROS, EL PRIMERO DECIDE.
-------------------------------------------------------
Ingrese los 3 números:
a: 4
b: 5
c: 3
SALIDA:
-------------------------------------------------------
12

Ingrese los 3 números:
a: -4
b: 5
c: 3
```

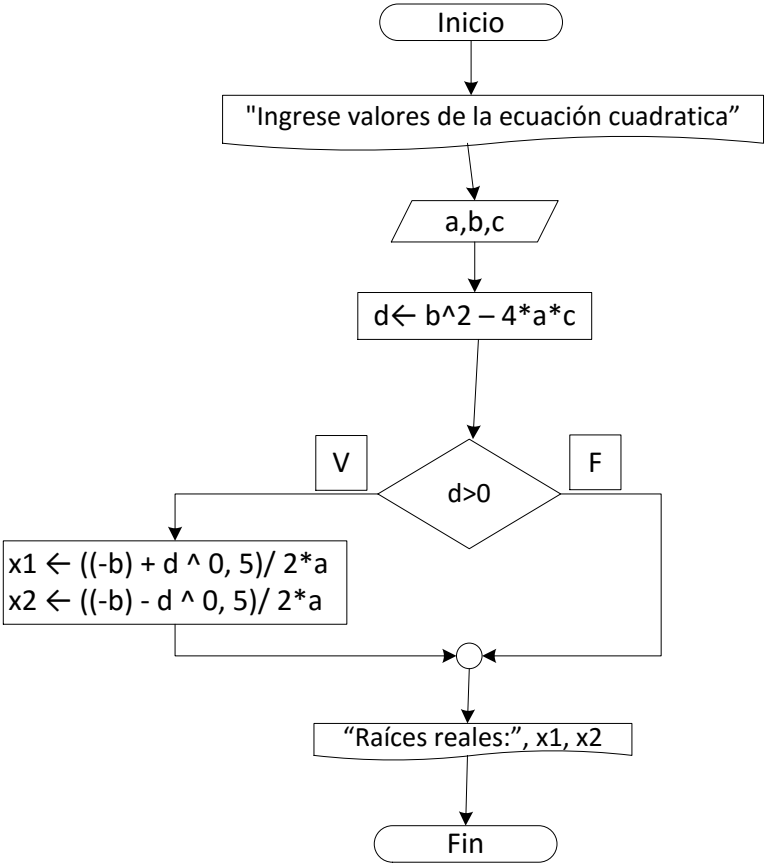
```
SALIDA:
-------------------------------------------------------
-60
```

7. Calcule las raíces reales, de una ecuación de segundo grado. La forma de una ecuación de segundo grado es $Ax^2 + Bx + C$, para este caso se debe tener presente el valor de la discriminante. Para el algoritmo se procede a calcular las raíces solo si la discriminante es mayor a CERO.

   Pseudocodigo.

```
Algoritmo raices
    Var
        a,b,c,d:real
    Inicio
        Escribir ("Ingrese valores de la ecuación cuadratica")
        Leer (a)
        Leer (b)
        Leer (c)
        d← b^2 – 4*a*c
        si d>0 Entonces
            x1 ← ((-b) + d ^ 0, 5)/ 2*a
            x2 ← ((-b) - d ^ 0, 5)/ 2*a
        Fin Si
        Escribir ("Raíces reales:", x1, x2)
    Fin
```

Diagrama de Flujo.




El código del complementario 7 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("Complemento7: RAÍCES DE ECUACIÓN CUADRÁTICA.")
print("-----------------------------------------------------")

#Entradas
```

```
print("Ingrese valores de la ecuación cuadrática:")
a = int( input("a: "))
b = int( input("b: "))
c = int( input("c: "))

#Proceso
d = b**2 - 4*a*c

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")

if d > 0 :
   x1 = ( (-b) + d**0.5 )/ 2*a
   x2 = ( (-b) - d**0.5 )/ 2*a
   print("Raíces reales:", x1, x2)
else:
   print("Raíces Irracionales")
```

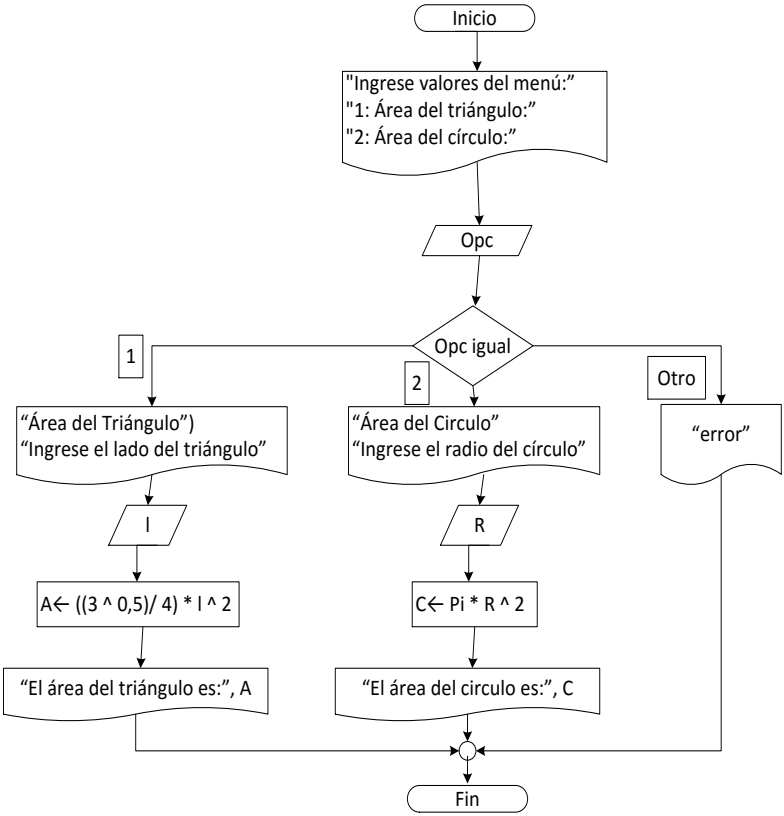
```
-------------------------------------------------------
Complemento7: RAÍCES DE ECUACIÓN CUADRÁTICA.
-------------------------------------------------------
Ingrese valores de la ecuación cuadrática:
a: 1
b: 5
c: 6

SALIDA:
-------------------------------------------------------
Raíces reales: -2.0 -3.0
```

8. Algoritmo que permita efectuar el cálculo del área de un círculo o un triángulo equilátero según la opción seleccionada por el usuario a través de un menú, además se deben ingresar los datos adicionales que se requieran para el cálculo del área.

Pseudocódigo.

```
Algoritmo menu
    Var
        opc:entero
        A,R,l,C:real
    Const
        Pi=3.1416
    Inicio
        Escribir ("Ingrese valores del menú:”)
        Escribir ("1: Área del triángulo:”)
        Escribir ("2: Área del círculo:”)
        Leer (Opc)
        Si Opc  igual
            1:  Escribir (“Área del Triángulo”)
                Escribir (“Ingrese el lado del triángulo”)
                Leer (l)
                A← ((3 ^ 0,5)/ 4) * l ^ 2
                Escribir (“El área del triángulo es:”, A)
            2:  Escribir(“Área del Circulo”)
                Escribir (“Ingrese el radio del círculo”)
                Leer (R)
                C← Pi * R ^ 2
                Escribir (“El área del círculo es:”, C)
            Otro:  “error”
        Fin selector
    Fin
```

Diagrama de Flujo.




El código del complementario 8 en Python es el siguiente:

```
#ELIF
#Para sustituir un Switch o IF's anidados
#se puede hacer uso de palabra reservada ELIF

if Condicion1:
   #Se ejecuta si la Condicion1 es Verdadera
   Declaraciones
elif Condicion2:
```

```
    #Si la Condicion1 no se Ejecuto
    #y si la Condicion2 es Verdadera
    Declaraciones
elif Condicion3:
    #Si la Condicion2 no se Ejecuto
    #y si la Condicion3 es Verdadera
    Declaraciones
else:
    #Si las condiciones anteriores no se cumplen
    Declaraciones
```

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento8: TRIÁNGULO O CÍRCULO.")
print("-------------------------------------------------------")

#Constantes
PI = 3.1416

#Entradas
print("Ingrese valores del menú:")
print("1: Área del triángulo:")
print("2: Área del círculo:")

Opc = int( input("Opc: "))

#Proceso
if Opc == 1 :
    print("Área del Triángulo")
    print("Ingrese el lado del triángulo")
    L = float( input("L: "))
    A = ( (3 ** 0.5)/ 4)  *  L**2
    print("\nEl área del triángulo es:", A)

elif Opc == 2:
    print("Área del Círculo")
    print("Ingrese el radio del círculo")
    R = float( input("R: "))
    C = PI * R**2
    print("\nEl área del círculo es:", C)
```

```
else:
    print("\nerror")
```

```
-------------------------------------------------------
Complemento8: TRIÁNGULO O CÍRCULO
-------------------------------------------------------
Ingrese valores del menú:
1: Área del triángulo:
2: Área del círculo:
Opc: 2
Área del Círculo
Ingrese el radio del círculo
R: 3

El área del círculo es: 28.2744
```

# 5

## Estructuras Repetitivas

Las computadoras están especialmente diseñadas para todas aquellas aplicaciones en las cuales una operación o conjunto de ellas deben repetirse muchas veces (ciclos repetitivos).

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan bucles. En algunos algoritmos podemos establecer a priori que el bucle se repetirá un número de veces. Es decir, el número de repeticiones no dependerá de las proposiciones dentro del ciclo, a esta estructura se le conoce como PARA O DESDE. (Lecca, 1998)

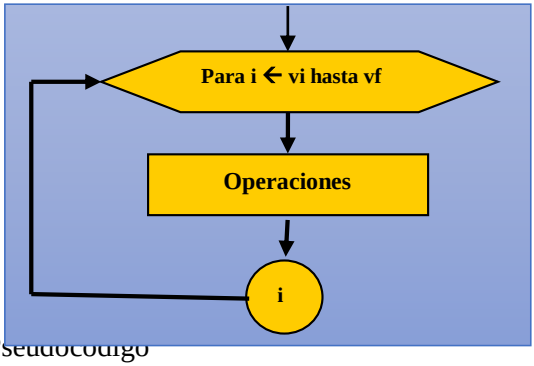
Por otra parte, en algunos algoritmos no podemos establecer a priori el bucle, sino que dependerá de las proposiciones dentro del mismo. En este grupo de estructuras se encuentra MIENTRAS y REPETIR.

### 5.1 Desde o Para (For):

Este tipo de bucles se utiliza cuando se sabe ya antes de ejecutar el bucle **el número exacto de veces que hay que ejecutarlo**.

Sintaxis:

Diagrama de Flujo.




Pseudocódigo

**Desde** <var índice> ← <vi> hasta <vf>
<operaciones>
…..
…..
**fin desde**

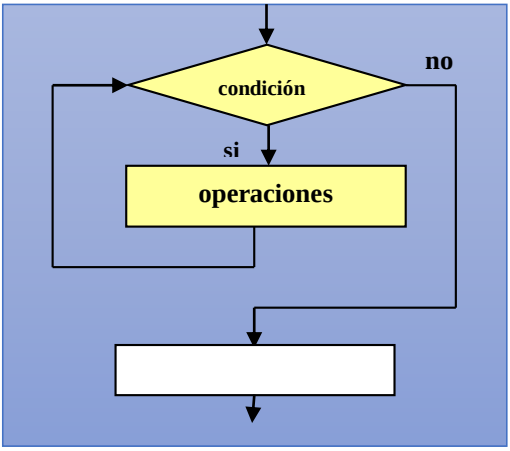
Donde:
- ✓ Var = variable índice de control
- ✓ Vi = Valor inicial
- ✓ Vf = Valor final

## 5.2 Mientras (While)

La condición del bucle **se evalúa al principio, antes de entrar en él**. *Si la condición es verdadera, comenzamos a ejecutar las acciones del bucle y después de la última volvemos a preguntar por la condición*. Usaremos obligatoriamente este tipo de bucle en el caso de que exista la posibilidad de que el bucle pueda **ejecutarse 0 veces**.

Sintaxis:

Diagrama de Flujo.



Pseudocódigo.

**Mientras** <condición> **hacer**

<operaciones>

…..

…..

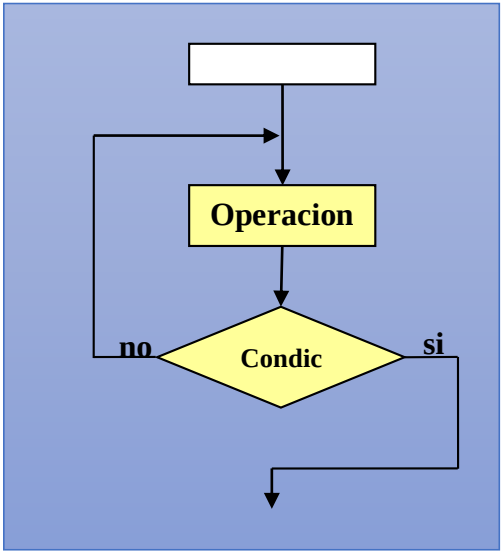
**fin mientras**

## 5.3 Repetir (Repit)

El bucle repite mientras la condición sea falsa. **La condición se evalúa siempre al final del bucle**, si es falsa volvemos a ejecutar las acciones, si es verdad se sale del bucle.

Cuando **un bucle se tenga que ejecutar como mínimo una vez**, podremos usar una estructura repetir.

Diagrama de Flujo.



Pseudocódigo.

**Repetir**
**<Operaciones>**
**…**
**…**
**hasta que <condición>**

Estructuras repetitivas en Python:

```
#WHILE

while condicion:
  #Se ejecuta mientras la condición es verdadera

#Ejemplo
i = 1
while i < 10:
    #Se ejecuta si i es menor a 10
    print(i)
    i = i + 1 #contador
    #Para evitar crear un loop infinito
    #debemos hacer que la condición sea FALSA en algún
    #momento. (será falsa cuando i sea igual a 10)
```

```
#FOR
#El FOR de Python no es el clásico que existe en C++ o
#java, es un FOREACH.
#Significa que necesita de una lista para iterar
#Itera tantos elementos tenga la lista.

lista = [1,3,5,2,4,7,8,6,9]
#este for imprime pro pantalla los valores de la lista
for elemento in lista:
    #En cada iteración ELEMENTO toma un valor de LISTA
    print(elemento)
```

```
#FUNCIÓN RANGE
#RANGE es una función que crea lista de acuerdo a sus paramentros

#si tiene un solo parámetro: de cero hasta menor que el numero
range(10) #Crea una lista desde 0 hasta 9: [0,1,2,3,4,5,6,7,8,9]

#range(inicio, fin)
range(1, 10) #Crea una lista desde 1 hasta 9: [1,2,3,4,5,6,7,8,9]

#range(inicio, fin, incremento/decremento)
range(1, 10, 2)
#Crea una lista desde 1 a 9 de 2 en 2: [1,3,5,7,9]

#NOTA: El último elemento no se toma en cuenta. Si es 10, se cuenta hasta 9
```

```
#FOR Y RANGE
#Se usa en situaciones en el que queremos
# que un for itere cierta cantidad de veces.

#Funcionamiento
#   Range crea una LISTA de acuerdo con sus parámetros
#   En cada iteración i toma un valor de la LISTA
for i in range(10):
    print(i)
```

```
#BREAK
#Sentencia que permite Salir de un Bucle
for i in range(100):
    print(i)
    if i == 4:
        #Cuando i tenga el valor de 4
        break #Se rompe el bucle
```

```
#CONTINUE
#Permite detener la iteración actual y
#continuar con la siguiente

for i in range(10)
    if i == 5:
        #cuando i tome el valor de 5
        #Saltará a la siguiente iteración
```

```
    #por lo tanto no se imprimirá su valor
    continue

print(i)
```

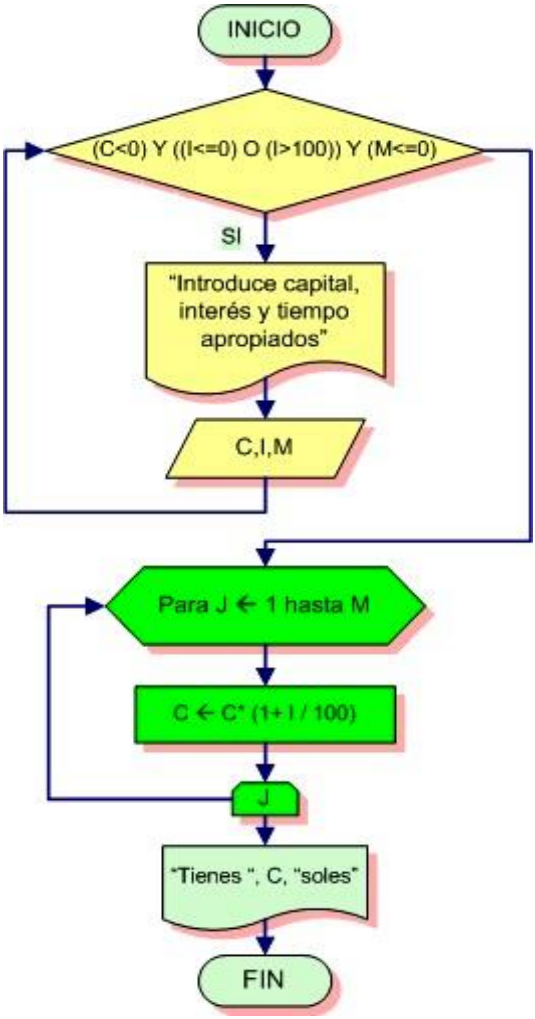
## 5.4 Ejercicios del Tema

1. Se coloca un capital C, a un interés I (que oscila entre 0 y 100), durante M años y se desea saber en cuanto se habrá convertido ese capital en "M" años, sabiendo que es acumulativo.

   PSEUDOCODIGO:

```
Algoritmo interés
Var
  I, j, M : entero
  C : real

Inicio
  Mientras (C<0) o (I<=0) o (I>=100) o (M <=0)
    Escribir ("Introduce el capital, el interés y el tiempo apropiados")
    Leer (C, I, M)
  Fin Mientras

  Desde j=1 hasta M
    C ← C * (1+I/100)
  Fin desde
  Escribir ("Tienes ", C , " soles")

Fin
```

Diagrama de Flujo.



El código del ejercicio 1 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio1: CALCULA EL INTERÉS.")
print("-------------------------------------------------------")

#Inicialización
C = -1
I = 0
M = 0
#Entradas
while (C<0) or (I<=0) or (I>=100) or (M <=0):
   print("Introduce el capital, el interés y el tiempo apropiados")
   C = int( input("Capital: "))
   I = int( input("Interés: "))
   M = int( input("Tiempo en Años: "))
#Proceso
for i in range(M):
   C = C*( 1 + I/100)
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("Tienes", C , "soles")
```

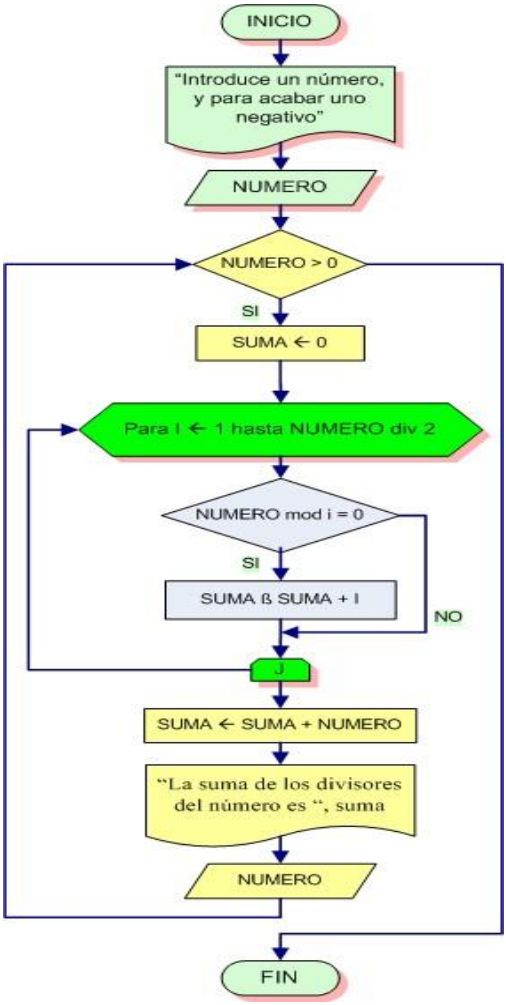
```
-------------------------------------------------------
Ejercicio1: CALCULA EL INTERÉS.
-------------------------------------------------------
Introduce el capital, el interés y el tiempo apropiados
Capital: 1000
Interés: 10
Tiempo en Años: 5

SALIDA:
-------------------------------------------------------
Tienes 1610.5100000000002 soles
```

2. Calcular la suma de los divisores de cada número introducido por teclado. Terminaremos cuando el número ingresado sea negativo.

   Pseudocódigo.

```
Algoritmo Divisores
Var
  Numero, i, suma : entero
Inicio
  Escribir ("Introduce un número, y para acabar uno negativo")
  Leer (Número)
  Mientras (numero > 0)
    Suma ← 0
    Desde i=1 hasta numero div 2
      Si (Numero mod i =0) entonces
        suma ← suma + i
      Fin si
    Fin desde
    suma ← suma + numero
    Escribir ("La suma de los divisores del número es ", suma)
    Leer (Numero)
  Fin mientras
Fin
```

Diagrama de Flujo.



El código del ejercicio 2 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("Ejercicio2: Calcula la suma de divisores.")
print("-----------------------------------------------------")
#Entradas
print("Introduce un número, y para acabar uno negativo:")
numero = int( input("Núm: "))
while numero > 0 :
   Suma = 0

   for i in range(1,numero+1):
      if numero % i == 0 :
         Suma = Suma + i
   #Salida
   print("\nSALIDA: ")
   print("-----------------------------------------------------")
   print("La suma de los divisores del número es:", Suma, "\n" )

   print("Introduce un número, y para acabar uno negativo:")
   numero = int( input("Núm: "))
-----------------------------------------------------
Ejercicio2: Calcula la suma de divisores.
-----------------------------------------------------
Introduce un número, y para acabar uno negativo:
Núm: 10

SALIDA:
-----------------------------------------------------
La suma de los divisores del número es: 18

Introduce un número, y para acabar uno negativo:
Núm: -1
```

3. ¿Cuáles y cuántos son los números primos comprendidos entre 1 y 1000? Construya su diagrama de flujo.

Pseudocódigo.

```
Algoritmo n_primos
Const
  primero = 1
  limite = 1000
Var
  Cont, i, j : entero
  primo : booleano
Inicio
  Cont ← 0
  Desde i= primero hasta limite
    primo ← verdadero
    j ← 2
    mientras (i > j) y (primo = verdadero)
      Si (i mod j = 0) entonces
        primo ← falso
      sino
        j ← j + 1
      Fin si
    Fin mientras
    Si primo = verdadero entonces
        escribir (i, " es primo")
       Cont ← Cont + 1
    Fin si
  Fin desde
  Escribir ("Entre ", primero, " y " , limite, " hay ",
Cont, " nº primos")
Fin
```

El código del ejercicio 3 en Python es el siguiente:

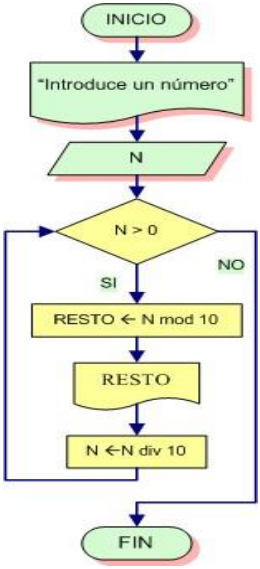
```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio3: NÚMEROS PRIMOS ENTRE 1 Y 1000.")
print("-------------------------------------------------------")
#Constantes
primero = 2
limite = 1000
#Inicializando
Cont = 0
#Proceso
for i in range(primero, limite):
    primo = True
    j = 2

    while (i > j) and (primo == True):
        if i%j == 0 :
            #como no se considerá el Número(i) y el 1
            #Si otro numero divide a i entonces ya tendría
            # tres divisores. por lo tanto, no es primo
            primo = False
            break #Por tal motivo se rompe el ciclo
        else:
            j = j + 1

    if primo == True :
        print(i, "es primo.")
        Cont = Cont + 1

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Entre", primero, "y" , limite, "hay", Cont, "nº primos")
```

```
-------------------------------------------------------
Ejercicio3: NÚMEROS PRIMOS ENTRE 1 Y 1000.
-------------------------------------------------------
2 es primo.
3 es primo.
```

```
5 es primo.
7 es primo.
11 es primo.
13 es primo.
.
..
977 es primo.
983 es primo.
991 es primo.
997 es primo.

SALIDA:
-------------------------------------------------------
Entre 2 y 1000 hay 168 nº primos
```

4. Determine la salida del diagrama de flujo para el dato de entrada **456871**

   Diagrama de Flujo.



El código del ejercicio 4 en Python es el siguiente:

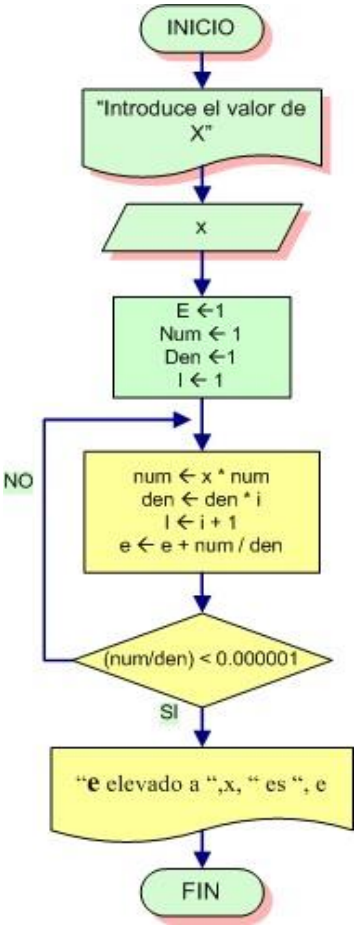
```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("Ejercicio4: DETERMINAR LA SALIDA.")
print("-----------------------------------------------------")

#Constantes

#Entradas
print("Introduce un número: ")
N = int( input())

while N > 0 :
   RESTO = N % 10
   print(RESTO)

   N = N // 10 #División Entera, N/10, es una división decimal
```

```
-----------------------------------------------------
Ejercicio4: DETERMINAR LA SALIDA.
-----------------------------------------------------
Introduce un número:
456871
1
7
8
6
5
4
```

5. Dado que el valor de $\mathbf{e^x}$ se puede calcular por aproximación de la siguiente suma:

$$\mathbf{e = 1 + x + x^2/2! + x^3/3! + \ldots\ldots + x^n/n!}$$

Realizar el algoritmo (Diagrama de flujo) que tome un valor para **X y** calcule $\mathbf{e^x}$ hasta que $x^n/n!$ (error o aproximación) sea menor a 0.00001




El código del ejercicio 5 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio5: CALCULANDO e^x.")
print("-------------------------------------------------------")
```

```
#Constantes
#Entradas
print("Ingrese el valor de X: ")
x = int( input())

#Inicialización
e = 1
num = 1
den = 1
i = 1
#Sería un caso de Do While
#PERO: Python no tiene implementado la sintaxis Do While
#por lo tanto habrá que ingeniárselas
#DO
num = x**i
den = den*i
i = i + 1
e = e + num/den
#WHILE
while not (num/den < 0.000001):
    num = x**i
    den = den*i
    i = i + 1

    e = e + num/den
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("e elevado al", x, "es: ", e)
```

```
-------------------------------------------------------
Ejercicio5: CALCULANDO e^x.
-------------------------------------------------------
Ingrese el valor de X:
2
SALIDA:
-------------------------------------------------------
e elevado al 2 es:  7.3890560703259105
```

6. Dado el siguiente Pseudocódigo determine cuál es el valor y el rango del primer término que supere el valor 300

```
Algoritmo serie
Var
  A1, A2, an, cont: entero
Inicio
  A1 ← 1
  A2 ← 0
  An ← A1 + (2 * A2)
  Mientras an < 300
    A2 ← A1
    A1 ← an
    an ← A1 + (2 * A2)
    cont ← cont + 1
  Fin mientras
  Escribir ("El rango es ", cont, " y el resultado es", an)
fin
```

El código del ejercicio 6 en Python es el siguiente:

```python
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Ejercicio6: DETERMINAR VALOR Y RANGO MAYOR A 300.")
print("-------------------------------------------------------")
#Inicialización
A1 = 1
A2 = 0
An = A1 + (2 * A2)
cont = 0

while An < 300:
   A2 = A1
   A1 = An
   An = A1 + (2 * A2)
   cont = cont + 1
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
```

```
print("El rango es:", cont, "y el resultado es:", An)
```

```
-------------------------------------------------------
Ejercicio6: DETERMINAR VALOR Y RANGO MAYOR A 300.
-------------------------------------------------------

SALIDA:
-------------------------------------------------------
El rango es: 8 y el resultado es: 34
```

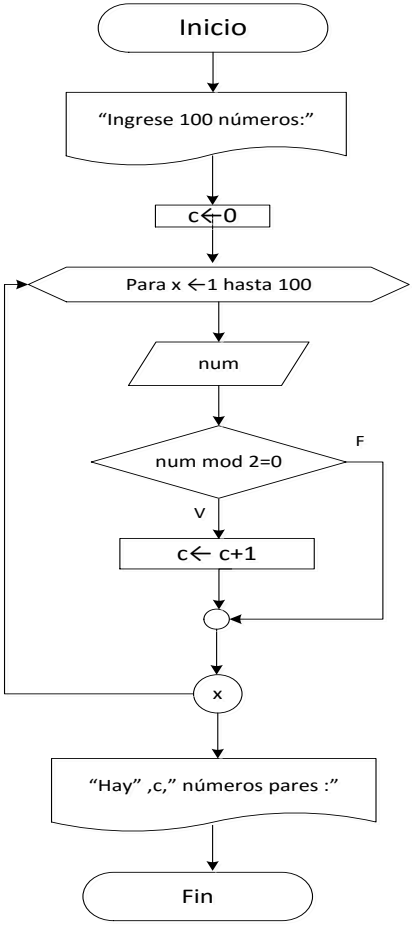
## 5.5 Ejercicios Complementarios

1) Ingresar por teclado 100 números enteros y calcular cuántos de ellos son pares. Se emprime el resultado.

   Pseudocódigo.

```
Algoritmo numeros_pares
    Var
        c,x,num: entero
    Inicio
        c←0
        Escribir (“Ingrese 100 números:”)
        Para x←1 hasta 100 salto 1
            Leer (num)
            Si num mod 2 = 0 entonces
                c←c+1
            Fin si
        Fin para
        Escribir (“Hay “,c,” números pares :”)
    Fin
```

Diagrama de Flujo.




El código del complementario 1 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento1: CONTAR NÚMEROS PARES.")
```

```
print("-------------------------------------------------------")

#Inicializar
c = 0

#Proceso
print("Ingrese 10 números: ")

for i in range(1, 10 + 1):
    num = int( input("Ingrese Número: "))

    if num % 2 == 0 :
        c = c + 1

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("Hay", c, "números pares")
```

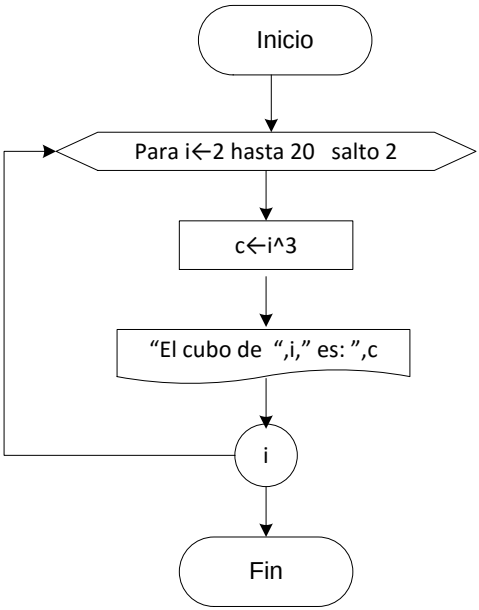
```
-------------------------------------------------------
Complemento1: CONTAR NÚMEROS PARES.
-------------------------------------------------------
Ingrese 10 números:
Ingrese Número: 2
Ingrese Número: 3
Ingrese Número: 4
Ingrese Número: 7
Ingrese Número: 4
Ingrese Número: 5
Ingrese Número: 7
Ingrese Número: 8
Ingrese Número: 4
Ingrese Número: 5

SALIDA:
-------------------------------------------------------
Hay 5 números pares
```

2) Escribir un algoritmo que imprima los 10 primeros números pares comenzando en 2 e imprima también sus respectivos cubos. Por ejemplo: 2 – 8 ; 4 – 64; 6 – 216 …

Pseudocódigo.

```
Algoritmo cubos_pares
    Var
        c,i: entero
    Inicio
        Para i←2 hasta 20  salto 2
            c← i^3
        Fin para
        Escribir (“El cubo de  ",i," es:  ",c )
    Fin
```

Diagrama de Flujo.



El código del complementario 2 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento2: PARES Y SUS CUBOS.")
print("-------------------------------------------------------")

#Proceso
start = 2
stop = 20
step = 2    #incrementar de 2 en 2
for i in range(start, stop+1, step):
    c = i**3

    print("El cubo de ", i, "es ", c)
```

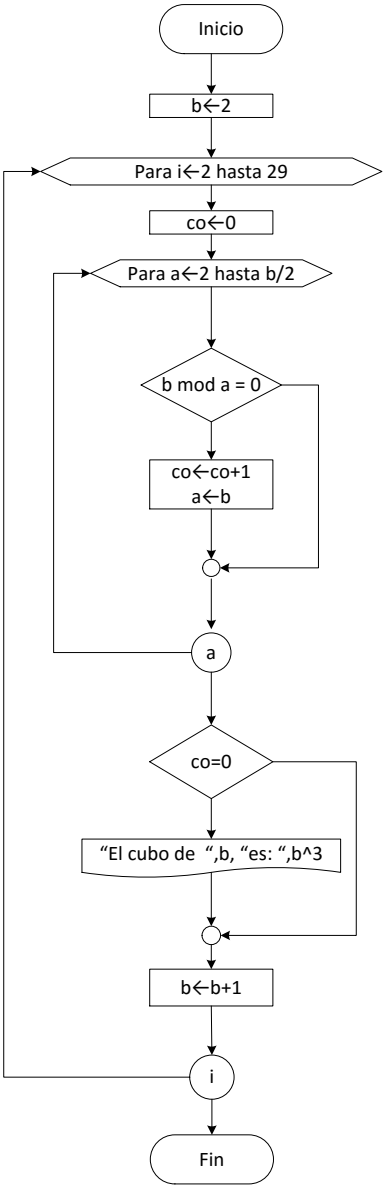
```
-------------------------------------------------------
Complemento2: PARES Y SUS CUBOS.
-------------------------------------------------------
El cubo de  2 es  8
El cubo de  4 es  64
El cubo de  6 es  216
El cubo de  8 es  512
El cubo de  10 es  1000
El cubo de  12 es  1728
El cubo de  14 es  2744
El cubo de  16 es  4096
El cubo de  18 es  5832
El cubo de  20 es  8000
```

3) Escribir un algoritmo que imprima los 10 primeros números primos comenzando en 2 e imprima también sus respectivos cubos. Por ejemplo: 2 – 8 ; 3 – 27; 5 –125 …

Pseudocódigo.

```
Algoritmo cubos_primos
    Var
        i,a,b,co: entero
    Inicio
        b←2
        Para i←2 hasta 29
            co← 0
            Para a←2 hasta b/2
                Si b mod a = 0 entonces
                    co←co+1
                    a← b
                Fin si
            Fin para
            Si co = 0  entonces
                Escribir ("El cubo de  ", b," es: ", b^3)
            Fin si
            b← b+1
        Fin para
    Fin
```

Diagrama de Flujo.



El código del complementario 3 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento3: CUBO DE UN NÚMERO PRIMO.")
print("-------------------------------------------------------")

#Inicializar
b = 2

#Proceso
for i in range(1, 29):

    co = 0

    for a in range(2, b//2):
        if b % a == 0 :
            co = co + 1
            a = b

    if co == 0 :
        print("El cubo de", b," es: ", b**3)
    b = b + 1
```

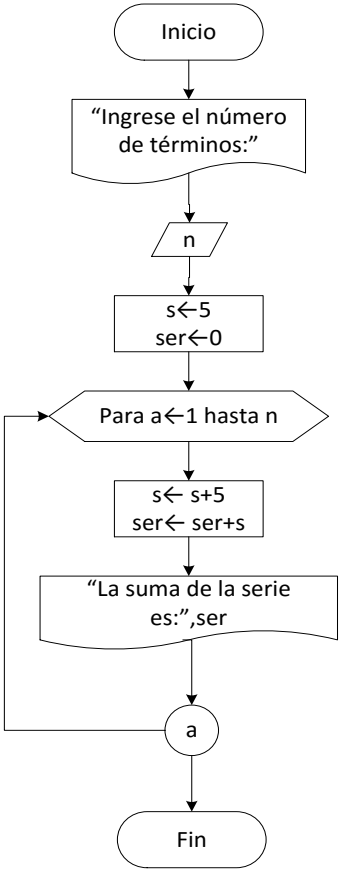
```
-------------------------------------------------------
Complemento3: CUBO DE UN NÚMERO PRIMO.
-------------------------------------------------------
El cubo de 2  es: 8
El cubo de 3  es: 27
El cubo de 4  es: 64
El cubo de 5  es: 125
El cubo de 7  es: 343
El cubo de 11  es: 1331
El cubo de 13  es: 2197
El cubo de 17  es: 4913
El cubo de 19  es: 6859
El cubo de 23  es: 12167
El cubo de 29  es: 24389
```

4) Construya un algoritmo que, dado una entrada n, calcule la suma de los términos de una progresión aritmética como la que se muestra a continuación: 10; 15; 20; 25; 30 …..Tn. Esto es, su algoritmo debe calcular el valor: 10 + 15 + 20 + 25 + 30 + … + Tn.

Pseudocódigo.

```
Algoritmo serie
    Var
        s,ser,a,n: entero
    Inicio
        Escribir (“Ingrese el número de términos:”)
        Leer (n)
        s←5
        ser←0
        Para a←1 hasta n
            s← s+5
            ser←ser+s
        Fin para
        Escribir (“La suma de la serie  es:  ",ser)
    Fin
```

Diagrama de Flujo




El código del complementario 4 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento4: CALCULAR SUMA DE SERIE.")
print("-------------------------------------------------------")
```

```
#Entradas
print( "Ingrese el número de términos: ")
n = int( input())
#Inicializar
s = 5
ser = 0

#Proceso
for a in range(1, n+1):
   s = s + 5
   ser = ser + s
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("La suma de la serie es:", ser)
```

```
-------------------------------------------------------
Complemento4: CALCULAR SUMA DE SERIE.
-------------------------------------------------------
Ingrese el número de términos:
3

SALIDA:
-------------------------------------------------------
La suma de la serie es: 45
```

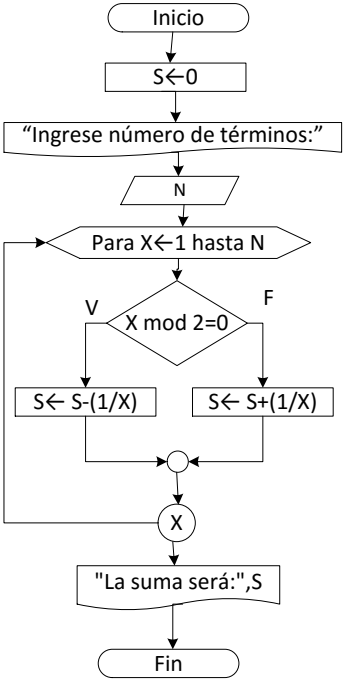
5) Escriba un algoritmo que lea un número entero N y calcule el resultado de la siguiente serie:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots \pm \frac{1}{N}$$

Pseudocódigo.

```
Algoritmo suma
    Var
        S,X,N:
    Inicio
        S← 0
        Escribir ("Ingrese numero de términos:")
        Leer (N)
        Para X←1 hasta N
            Si X mod 2=0 entonces
                S← S-(1/X)
            sino
                S← S+(1/X)
            Fin Si
        Fin para
        Escribir ("La suma será:",S)
    Fin
```

Diagrama de Flujo.



El código del complementario 5 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento5: CALCULAR SUMA DE SERIE 2.")
print("-------------------------------------------------------")

#Inicializar
S = 0
#Entradas
print ("Ingrese número de términos:")
N = int( input())

#Proceso
for x in range(1, N+1):

    if x % 2 == 0 :
        S = S - (1/x)
    else:
        S = S + (1/x)
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("La suma será:", S)
```

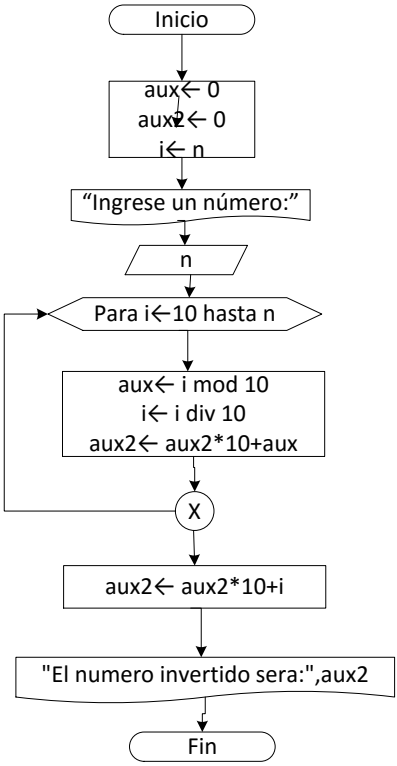
```
-------------------------------------------------------
Complemento5: CALCULAR SUMA DE SERIE 2.
-------------------------------------------------------
Ingrese número de términos:
20

SALIDA:
-------------------------------------------------------
La suma será: 0.6687714031754279
```

6) Escribir un algoritmo que invierta los dígitos de un número positivo entero. Usar operadores módulo y división para ir obteniendo los dígitos uno a uno. Por ejemplo, si se ingresa 37368 debe retornar el numero 86373

Pseudocódigo.

```
Algoritmo invertido
    Var
        aux,aux2,i,n : entero
    Inicio
        aux← 0
        aux2← 0
        i← n
        Escribir ("Ingrese un numero:")
        Leer (n)
        Para i←10 hasta n
            aux← i mod 10
            i← i div 10
            aux2← aux2*10+aux
        Fin Para
        aux2← aux2*10+i
        Escribir ("El numero invertido sera:",aux2)
    Fin
```

Diagrama de Flujo:




El código del complementario 6 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento6: INVERTIR NÚMERO.")
print("-------------------------------------------------------")

#Inicializar
aux = 0
aux2 = 0

#Entradas
print("Ingrese un número: ")
```

```
n = int( input())
#Proceso
i = 10
#TODO: REPORTAR COMO ERROR
while i <= n:
    aux = n%10
    n = n // 10
    aux2 = aux2*10 + aux
aux2 = aux2*10 + n
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print("El número invertido será:", aux2)
```

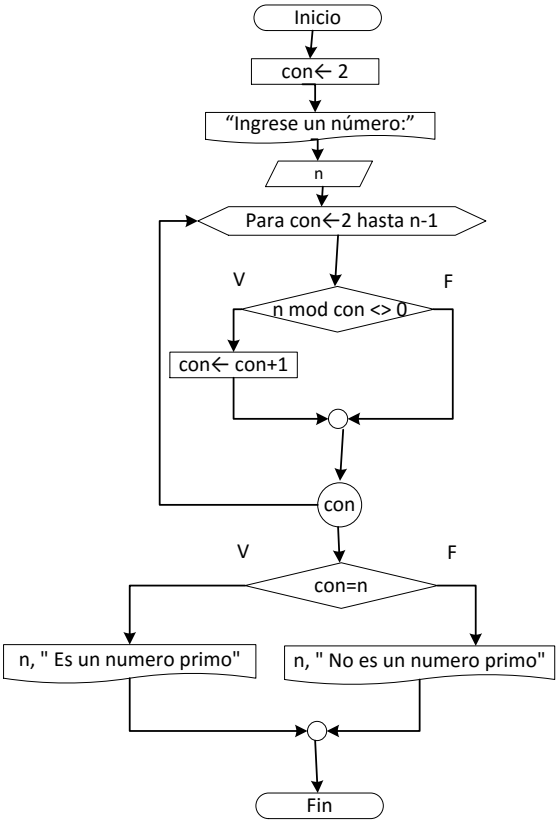
```
-------------------------------------------------------
Complemento6: INVERTIR NÚMERO.
-------------------------------------------------------
Ingrese un número:
37368
SALIDA:
-------------------------------------------------------
El número invertido será: 86373
```

7) Ingresar un número y determinar si es número PRIMO.

Pseudocódigo.

```
Algoritmo primo
    Var
        con,n : entero
    Inicio
        Escribir ("Ingrese un numero:")
        leer (n)
        Para con← 2 hasta n-1
            Si n mod con <>0 entonces
            con← con+1
            Fin Si
        Fin para
        si con=n Entonces
            escribir (n, " Es un numero primo")
        sino
            escribir (n, " No es un numero primo")
        Fin Si
    Fin
```

Diagrama de Flujo:




El código del complementario 7 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento7: DETERMINAR SI UN NÚMERO ES PRIMO.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese un número:")
n = int( input())
```

```
#Inicializar
con = 0

#Proceso
for i in range(2, n):
    if n % i == 0 :
        con = con + 1

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")

if con == 0:
    print (n, " Es un número primo")
else:
    print (n, " No es un número primo")
```

```
-------------------------------------------------------
Complemento7: DETERMINAR SI UN NÚMERO ES PRIMO.
-------------------------------------------------------
Ingrese un número:
23

SALIDA:
-------------------------------------------------------
23  Es un número primo

Ingrese un número:
27

SALIDA:
-------------------------------------------------------
27  No es un número primo
```

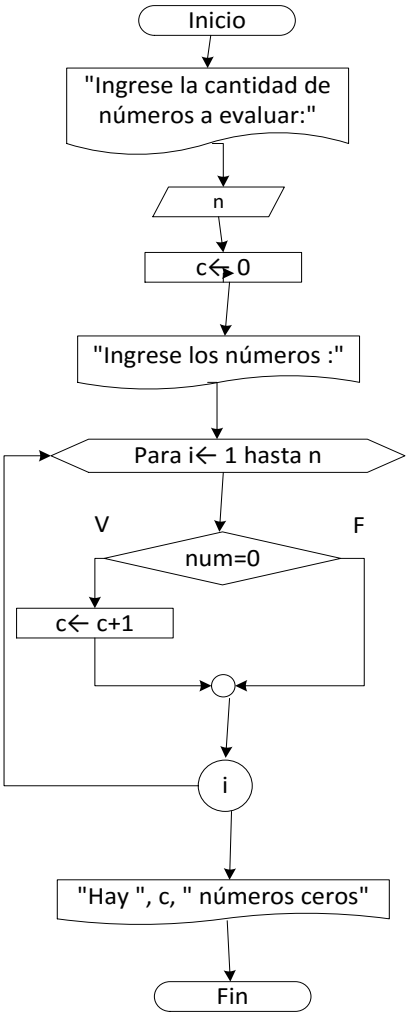
8) Escriba un algoritmo tal que dado como datos X números enteros, obtenga el número de ceros que hay entre estos números. Por ejemplo, si se ingresa 6 datos: 9 0 4 8 0 1

El algoritmo arroja que hay 2 ceros.

Pseudocódigo.

```
Algoritmo ceros
    Var
        c,num,i,n : entero
    Inicio
        Escribir ("Ingrese la cantidad de números a evaluar:")
        Leer (n)
        c← 0
        Escribir ("Ingrese los números:")
        Para i← 1 hasta n
            Leer (num)
            Si num=0 Entonces
                c← c+1
            FinSi
        Fin para
        Escribir ("Hay ", c, " números ceros")
    Fin
```

Diagrama de Flujo:



El código del complementario 8 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento8: NÚMERO DE CEROS ENTRE NÚMEROS.")
print("-------------------------------------------------------")

#Entradas
print ("Ingrese la cantidad de números a evaluar:")
n = int( input())

#Inicializar
c = 0

#Proceso
for i in range(1, n+1):
    num = int( input("Ingrese número: "))
    if num == 0 :
        c = c+1

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("Hay ", c, " números ceros")
```

```
-------------------------------------------------------
Complemento8: NÚMERO DE CEROS ENTRE NÚMEROS.
-------------------------------------------------------
Ingrese la cantidad de números a evaluar:
10
Ingrese número: 1
Ingrese número: 0
Ingrese número: 5
Ingrese número: 0
Ingrese número: 8
Ingrese número: 0
Ingrese número: 4
Ingrese número: 5
Ingrese número: 6
Ingrese número: 0
```

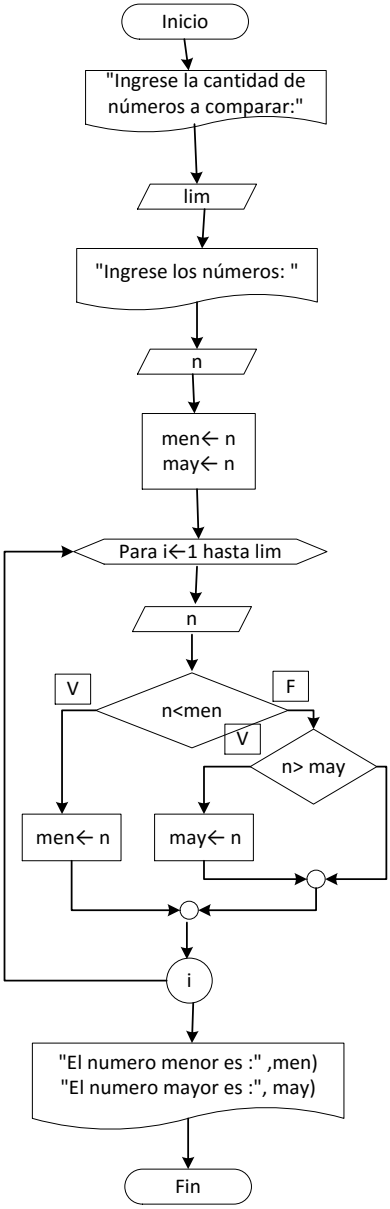
```
SALIDA:
-------------------------------------------------------
Hay 4  números ceros
```

9) Crear un algoritmo, que dado N números enteros ingresados por teclado determine cuál de ellos es el menor y mayor respectivamente.

Pseudocódigo.

```
Algoritmo mayme
    Var
        may,men,lim,i,n : entero
    Inicio
        Escribir ("Ingrese la cantidad de números a comparar:")
        leer (lim)
        Escribir ("Ingrese los números: ")
        leer (n)
        men← n
        may← n
        Para i← 1 hasta lim
            leer (n)
            si n<men Entonces
                men← n
            Sino
                si n> may Entonces
                    may← n
                FinSi
            Fin Si
        Fin Para
        Escribir ("El numero menor es :" ,men)
        Escribir ("El numero mayor es :", may)
    Fin
```

Diagrama de Flujo.


Inicio
"Ingrese la cantidad de números a comparar:"
lim
"Ingrese los números: "
n
men← n
may← n
Para i←1 hasta lim
n
V
F
n<men
V
n> may
men← n
may← n
i
"El numero menor es :" ,men)
"El numero mayor es :", may)
Fin

El código del complementario 9 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento9: DETERMINA EL MAYOR Y MENOR.")
print("-------------------------------------------------------")

#Entradas
print ("Ingrese la cantidad de números a comparar:")
lim = int( input())
print ("Ingrese los números: ")
n = int( input("Ingrese número: "))

#Proceso
men = n
may = n

for i in range(1, lim):
    n = int( input("Ingrese número: "))

    if n < men :
        men = n
    else:
        if n > may :
            may = n

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("El número menor es :" ,men)
print ("El número mayor es :", may)
```

```
-------------------------------------------------------
Complemento9: DETERMINA EL MAYOR Y MENOR.
-------------------------------------------------------
Ingrese la cantidad de números a comparar:
5
Ingrese los números:
Ingrese número: 4
Ingrese número: 20
```

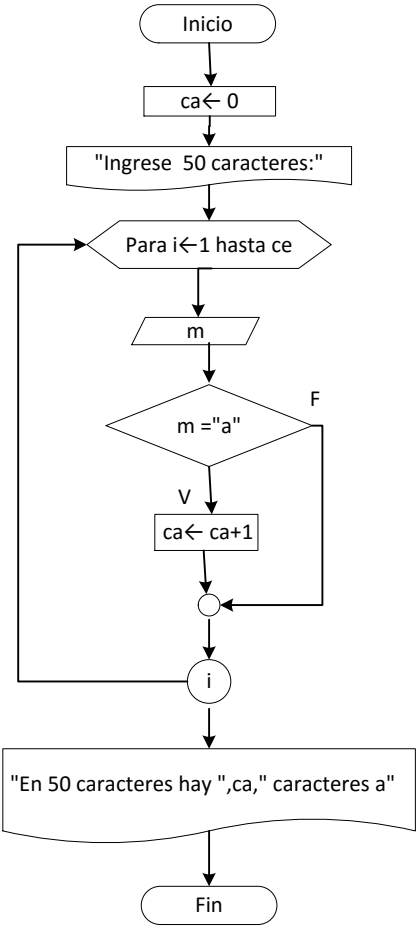
```
Ingrese número: 35
Ingrese número: 4
Ingrese número: 3

SALIDA:
-------------------------------------------------------
El número menor es: 3
El número mayor es: 35
```

10) Ingresar 50 caracteres e indicar cuantas veces se repite el carácter ‘a’.

Pseudocódigo.

```
Algoritmo caracter
    Var
        can,m,a,i : entero
    Inicio
        ca← 0
        Escribir ("Ingrese  50 caracteres:")
        Para i←1 hasta 50
            Leer (m)
            Si m ="a" Entonces
              ca← ca+1
            Fin Si
        Fin Para
        Escribir ("En 50 caracteres hay ",ca," caracteres a")
    Fin
```

Diagrama de Flujo.




El código del complementario 10 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print('Complemento10: CUANTAS VECES SE REPITE "a".')
print("-------------------------------------------------------")

#Inicializar
```

```
ca = 0
#Entradas
numCar = 10
print("Ingrese", numCar, "caracteres: ")

#Proceso
for i in range(0, numCar):
    m = input("Ingrese Caracter: ")
    if m == "a" :
        ca = ca + 1

#Salida
print("\nSALIDA: ")
print("-----------------------------------------------------")
print ("En", numCar, "caracteres hay", ca, "caracteres 'a'")
```

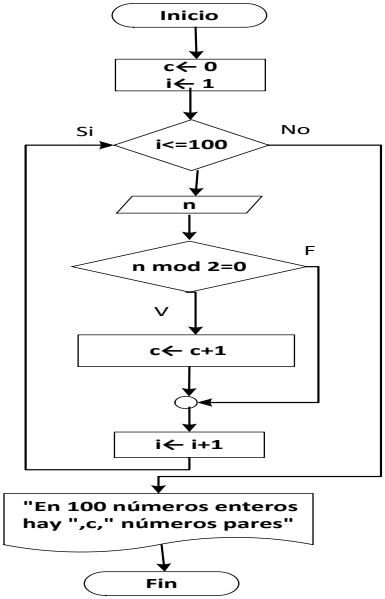
```
#Salida
-----------------------------------------------------
Complemento10: CUANTAS VECES SE REPITE "a".
-----------------------------------------------------
Ingrese 10 caracteres:
Ingrese Caracter: a
Ingrese Caracter: p
Ingrese Caracter: l
Ingrese Caracter: a
Ingrese Caracter: c
Ingrese Caracter: a
Ingrese Caracter: b
Ingrese Caracter: l
Ingrese Caracter: e
Ingrese Caracter: s

SALIDA:
-----------------------------------------------------
En 10 caracteres hay 3 caracteres 'a'
```

11) Ingresar por teclado 100 números enteros y calcular cuántos de ellos son pares. Se imprime el resultado.

Pseudocódigo.

```
Algoritmo par
   Var
      i,c,n:entero
   Inicio
      i← 1
      c← 0
      Mientras i<=100 hacer
         Leer (n)
         Si n mod 2=0 Entonces
            c← c+1
         Fin Si
         i← i+1
      Fin Mientras
      Escribir ("En 100 números enteros hay ",c," números pares")
   Fin
```

Diagrama de Flujo.



El código del complementario 11 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento11: CONTAR CUANTOS SON PARES.")
print("-------------------------------------------------------")

#Inicializar
i = 1
c = 0
numEntradas = 10

#proceso
print("Ingrese", numEntradas, "Números:")
while i <= numEntradas:
    n = int( input("Ingrese número: "))
    if n%2 == 0 :
        c = c + 1

    i = i + 1

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("En", numEntradas, "números enteros hay", c, "números pares")
```

```
-------------------------------------------------------
Complemento11: CONTAR CUANTOS SON PARES.
-------------------------------------------------------
Ingrese 10 Números:
Ingrese número: 5
Ingrese número: 20
Ingrese número: 35
Ingrese número: 11
Ingrese número: 16
Ingrese número: 4
Ingrese número: 80
Ingrese número: 21
Ingrese número: 23
Ingrese número: 50
```

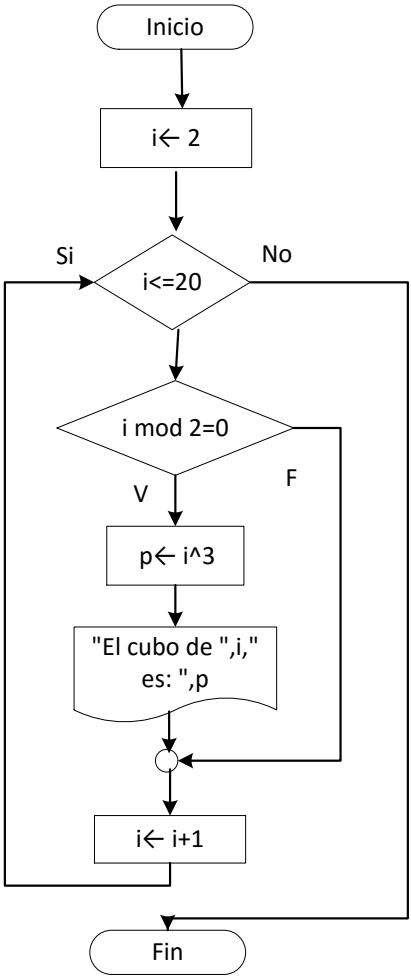
```
SALIDA:
-------------------------------------------------------
En 10 números enteros hay 5 números pares
```

12)Escribir un algoritmo que imprima los 10 primeros números pares comenzando en 2 e imprima también sus respectivos cubos. Por ejemplo: 2 – 8 ; 4 – 64; 6 – 216 …

Pseudocódigo.

```
Algoritmo cupar
    Var
        i,p:entero
    Inicio
        i← 2
        Mientras i<=20 Hacer
            Si i mod 2=0 Entonces
            p← i^3
            Escribir ("El cubo de ",i," es: ",p)
            Fin Si
        i← i+1
        Fin Mientras
    Fin
```

Diagrama de Flujo.



El código del complementario 12 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento12: IMPRIMIR PARES Y SUS CUBOS.")
print("-------------------------------------------------------")

#Entradas

#Inicializar
i = 2

#Proceso
while i <= 20 :
   if i%2 == 0 :
      p = i**3
      print("El cubo de", i, "es:", p)
   i = i + 1
```

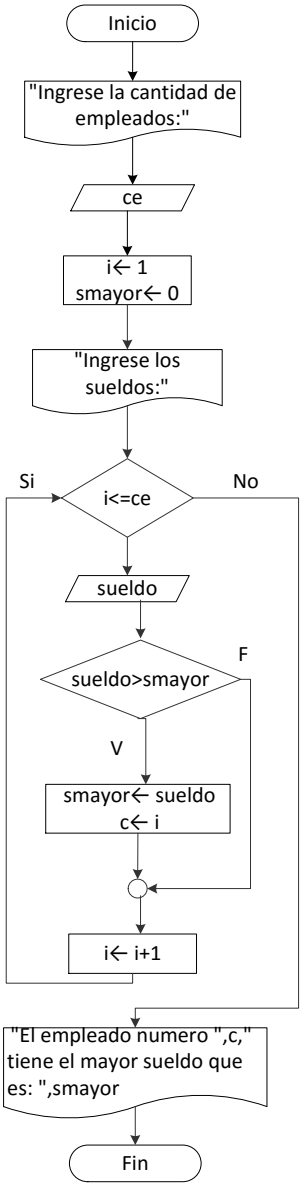
```
-------------------------------------------------------
Complemento12: IMPRIMIR PARES Y SUS CUBOS.
-------------------------------------------------------
El cubo de 2 es: 8
El cubo de 4 es: 64
El cubo de 6 es: 216
El cubo de 8 es: 512
El cubo de 10 es: 1000
El cubo de 12 es: 1728
El cubo de 14 es: 2744
El cubo de 16 es: 4096
El cubo de 18 es: 5832
El cubo de 20 es: 8000
```

13) Para una empresa con N empleados, se desarrolla un algoritmo donde se ingresa como datos el número de orden y sueldo de cada empleado, debe imprimirse el numero de orden del empleado con el mayor sueldo asi como su sueldo. Haga el algoritmo correspondiente.

Pseudocódigo.

```
Algoritmo sueldopo
    Var
        c,i,ce : entero
        smayor,sueldo: real
    Inicio
        Escribir ("Ingrese la cantidad de empleados:")
        Leer (ce)
        i← 1
        smayor← 0
        Escribir ("Ingrese los sueldos:")
        Mientras i<=ce Hacer
            Leer (sueldo)
            Si (sueldo>smayor) Entonces
                smayor← sueldo
                c← i
            FinSi
            i← i+1
        Fin Mientras
        Escribir "El empleado numero ",c," tiene el mayor sueldo que es: ",smayor
    Fin
```

Diagrama de Flujo.



El código del complementario 14 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento14: EMPLEADOS, SUELDO MAYOR.")
print("-------------------------------------------------------")

#Entradas
print ("Ingrese la cantidad de empleados:")
ce = int( input())

#Inicializar
i = 1
smayor = 0.0    #Inicializando Real

#Proceso
print("Ingrese los sueldos: ")
while i <= ce :
    sueldo = float( input("Ingrese sueldo {0}: ".format(i)))

    if sueldo > smayor :
        smayor = sueldo
        c = i

    i = i + 1

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("El empleado numero ", c, "tiene el mayor sueldo que es:", smayor)
```

```
-------------------------------------------------------
Complemento14: EMPLEADOS, SUELDO MAYOR.
-------------------------------------------------------
Ingrese la cantidad de empleados:
5
Ingrese los sueldos:
Ingrese sueldo 1: 1000
Ingrese sueldo 2: 800
Ingrese sueldo 3: 1500
```

```
Ingrese sueldo 4: 900
Ingrese sueldo 5: 850

SALIDA:
-------------------------------------------------------
El empleado numero 3 tiene el mayor sueldo que es: 1500.0
```

# 6

## Vectores y Matrices Funcionales

En los temas anteriores se ha introducido el concepto de datos de tipo simple, como un número entero, real o carácter. Muchas veces se necesita, procesar una colección de valores que están relacionados entre sí por algún método, por ejemplo, una serie de temperaturas, una lista de trabajadores, relación de inscritos a un evento, etc. El procesar estos datos utilizando datos simples, resultaría muy complejo, por ello los lenguajes de programación incluyen características de estructuras de datos, representado por los Arrays (vectores y matrices) de una o muchas dimensiones respectivamente.

Un array es una secuencia de posiciones de la memoria central a la que se puede acceder directamente, que contiene datos del mismo tipo y que puede ser relacionado y usado mediante el empleo de índices.

### 6.1 Vectores y Matrices

Los datos se clasificación según su estructura:

- Simples:

  Estándar (entero, real, carácter, booleano)

- Estructurados:

  Estáticos (arrays, cadena, registros, ficheros)

A continuación, veremos los datos estructurados conocidos como Arrays.

### 6.2 Arrays Unidimensionales: Vectores.

Un **array unidimensional, o lineal, o vector**, es un conjunto finito y ordenado de elementos homogéneos (del mismo tipo).

Podemos acceder a cada elemento del array de manera independiente a través de los índices.

**<nom_array>: array [LI .. LS] de <tipo>**

***Ejemplo:***

sueldo: array [1 .. 8] de real

sueldo: array [1990 .. 1997] de real

Asignación de un dato a una posición concreta del array:

**<nom_array>[indice] ← valor**

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 8000 | | | | | | | | | |

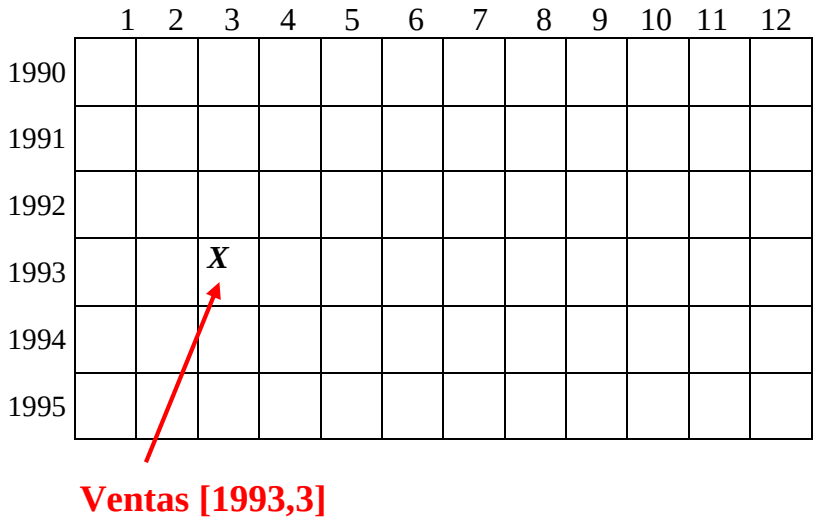
**Ventas [3] ← 8000**

## 6.3 Arrays Multidimensionales o Matrices

En un array unidimensional o vector cada elemento se referencia por un índice, en un **array multidimensional, cada elemento se va a referenciar por 2 o más índices**, y ahora la representación lógica ya no va a ser un vector, sino una matriz.

Podemos definir un array de 2 dimensiones de la siguiente manera.

**<nom_array>: array [LI1..LS2 , LI2..LS2 , ….] de <tipo>**




```
#LISTA
# En Python existen diferentes estructuras de datos
# El que más se parece a vectores y Matrices son las
# LISTAS.

# Las litas son elementos muy dinámicos, es decir, su dimensión
# puede variar dinámicamente, pueden iniciar teniendo 0
# elementos, y añadirle tantos elementos como se desee.
```

```
#DECLARACIÓN DE UNA LISTA
# Si se quiere trabajar con listas estas primero deben ser
# Declaradas Caso contrario el programa no lo interpreta
# como una lista.

#LISTA VACÍA
#Una lista vacía se declara usando los caracteres '[]'
miLista = []

#Una lista al declararse ya puede iniciar con elementos
frutas = ["Pera", "Manzana", "Piña", "Fresa"]
```

```
#FUNCIÓN APPEND
```

```
# Mediante la función APPEND se puede agregar elementos a la
# lista. Cada elemento se añadirá al último de la lista

 miLista.append("Leche")
 miLista.append("Pan")
 miLista.append("Queso")

# miLista: es ahora ["Leche", "Pan", "Queso"]
```

```
#ACCEDER A LOS ELEMENTOS DE UNA LISTA
# para acceder a los elementos se usa el operador '[índice]'
# índice: debe ser una posición del elemento que exista
# sino lanzará un error

# Ejemplo:
print(miLista[0])   #Accede al primer elemento
print(miLista[1])   #Accede al segundo elemento
print(miLista[2])   #Accede al tercer elemento

# NOTA: los índices van de 0 hasta tamaño de Lista menos 1
```

```
#FUNCIÓN SORT
#Las litas tienen muchas funciones para interactuar con ellas
# Una de ellas es SORT.
# SORT: ordena los elementos de manera ascendente
numeros = [5, 1, 3, 2, 4]
numeros.sort()  #lista ordenada: [1,2,3,4,5]
print(numero)
```

```
#LISTA POR NÚMERO
# En algunas ocasiones se desea que la lista tenga una dimensión # definida
y que se llene con valores por defecto.
# Una forma de hacer esto es mediante un producto: lista por
# número
numeros = 5*[0] #se crea la siguiente lista [0, 0, 0, 0, 0]
texto = ['abc']*3 #se crea la lista ['abc', 'abc', 'abc']
#NOTA: las expresiones 5*[0] es igual a [0]*5
#SUB LISTAS
# En ocasiones se desea extraer subconjuntos de la lista:
numeros = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
#se desea un subconjunto del 2 al 7
subconjunto = numeros[2:8]  #subconjunto: [2, 3, 4, 5, 6, 7]
print(subconjunto)
```

```
#FUNCIÓN LEN
# LEN es una función de Python muy dinámica, que permite
# conocer el tamaño de su parámetro: Este puede ser Listas,
# o cadenas de texto u otra estructura de datos.
numeros = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
tamNumeros = len(numeros)   #tamNumeros es igual a 10
print(tamNumeros)
```

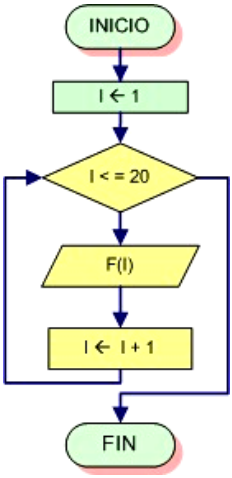
```
#Matrices en Python
# Para simular una matriz en Python Se puede aprovechar la
# capacidad de almacenar listas dentro de otras listas.
#Se tiene los clientes: sus nombres y teléfonos
clienteA = ["Juan","123-234"]
clienteB = ["Pedro","345-746"]

Agenda = []
Agenda.append(clienteA)
Agenda.append(clienteB)
print(Agenda) #Agenda: [["Juan","123-234"], ["Pedro","345-746"]]
#ACCEDER A SUS DATOS
# Se debe hacer uso de los índices, como en otros lenguajes.
# El conteo de índices inicia en 0

#nombre del clienteA
print(Agenda[0][0]) #Salida: Juan

#Telefono del ClienteB
print(Agenda[1][1]) #Salida: 345-746
```

## 6.4 Ejercicios del Tema

### Vectores

1. Lectura de 20 elementos enteros de un vector denominado **alfa.**

| *APLICANDO DESDE* | *APLICANDO MIENTRAS* |
|---|---|
| **Algoritmo** Leer_Vector<br>**Var**<br>**F :** array (1..20) de entero<br>**Inicio**<br>Desde i ← 1 hasta 20 hacer<br>Leer (F(i))<br>Fin desde<br>**Fin** | **Algoritmo** Leer_Vector<br>**Var**<br>**F :** array (1..20) de entero<br>**Inicio**<br>i ← 1<br>Mientras (i < = 20) hacer<br>Leer(F(i))<br>i ← i + 1<br>Fin_mientras<br>**Fin** |



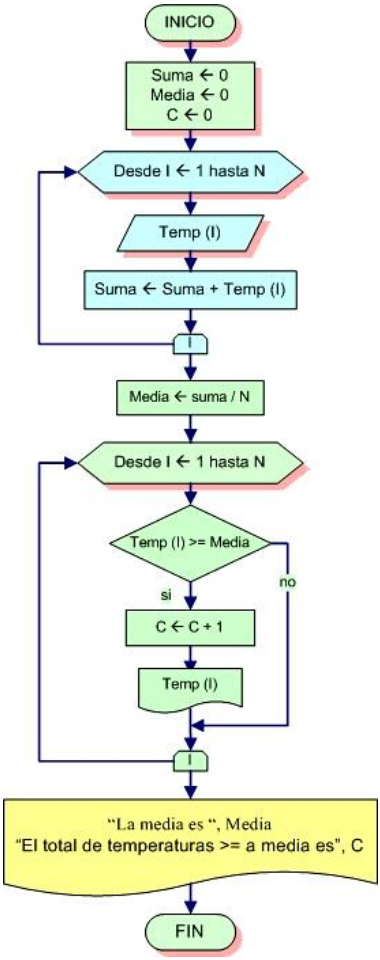
El código del ejercicio 1 de Vectores en Python usando mientras (while) es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Vector1: LECTURA DE N ELEMENTOS ENTEROS.")
print("-------------------------------------------------------")
#inicializar
i = 1
F = []      #Inicializamos una LISTA VACÍA
#Entrada
print("Ingrese Número de elementos a Ingresar: ")
numElementos = int( input())
#Proceso
while i <= numElementos:
    elemento = int( input("Ingrese Elemento: "))
    F.append(elemento)      #Agregamos el elemento a la lista

    i = i + 1
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(F)    #Imprimimos la lista
```

```
-------------------------------------------------------
Vector1: LECTURA DE N ELEMENTOS ENTEROS.
-------------------------------------------------------
Ingrese Número de elementos a Ingresar:
10
Ingrese Elemento: 5
Ingrese Elemento: 7
Ingrese Elemento: 8
Ingrese Elemento: 9
Ingrese Elemento: 10
Ingrese Elemento: 3
Ingrese Elemento: 4
Ingrese Elemento: 2
Ingrese Elemento: 23
Ingrese Elemento: 29
SALIDA:
```

```
-------------------------------------------------------
[5, 7, 8, 9, 10, 3, 4, 2, 23, 29]
```

2. Se tiene N temperaturas. Se desea calcular su media y determinar entre todas ellas cuantas son superiores o iguales a esa media.



```
Algoritmo Temperaturas
Const
   N = 100
Var
   Temp : array (1..N) de real
   I,C : Entero
   Suma, Media : Real
Inicio
Suma ← 0
Media ← 0
C ← 0
Desde I ← 1 hasta N hacer
   Leer (Temp(I))
   Suma ← Suma + Temp(I)
Fin desde
Media ← Suma / N
Desde I ← 1 hasta N hacer
  Si Temp(I) >= Media entonces
      C ← C + 1
      Escribir (Temp(I))
  Fin Si
Fin desde
Escribir ("La media es ", Media)
Escribir ("Total de temperaturas >= a media
   es", C)
Fin
```

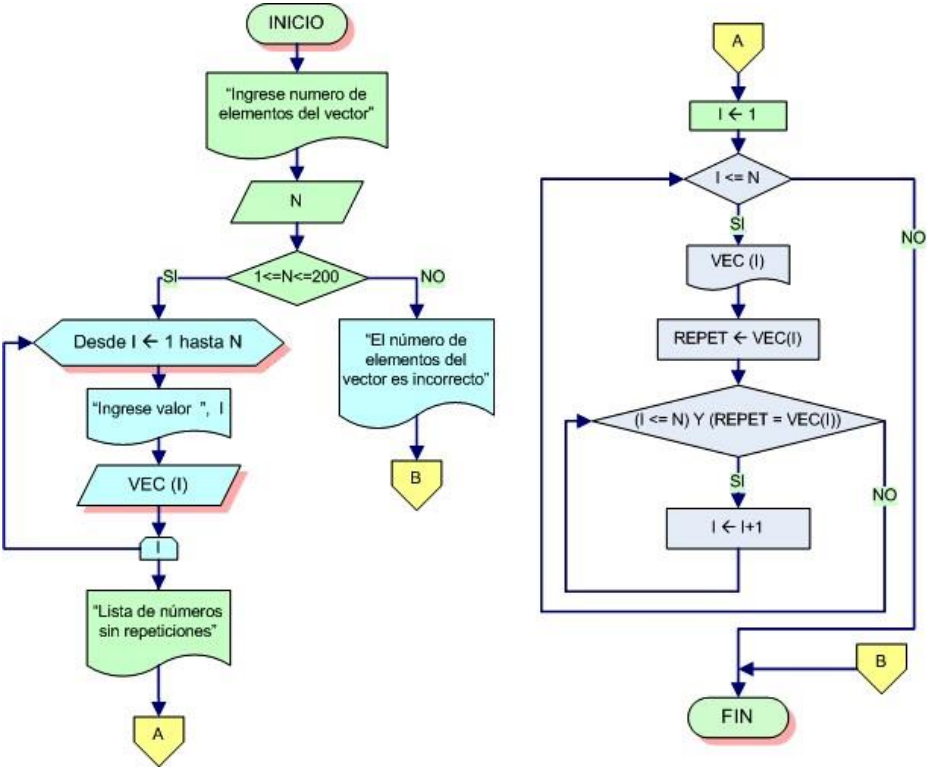
El código del ejercicio 2 de Vectores en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Vector2: MEDIA DE TEMPERATURAS.")
print("-------------------------------------------------------")

#Inicializar
Suma = 0
Media = 0.0
C = 0
Temp = []   #Lista vacía para almacenar temperaturas

#Entradas
print("Ingrese cantidad de Temperaturas: ")
N = int( input())

#Proceso
for i in range(N):
   temperatura = float( input("Ingrese Temperatura {0}: ".format(i + 1) ))
   Temp.append(temperatura)
   Suma = Suma + Temp[i]
   #Tambien se puede usar esta línea En lugar de la anterior:
   #Suma = Suma + temperatura

Media = Suma / N    #División real

#Contar cuantas temperaturas son mayor que la MEDIA

for tempElement in Temp:     #La magia del FOR de PYTHON inicia.
   if tempElement >= Media:
      C = C + 1
      print(tempElement)

#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print ("La media es ", Media)
print ("Total de temperaturas >= a la media es", C)
```

```
-------------------------------------------------------
Vector2: MEDIA DE TEMPERATURAS.
-------------------------------------------------------
Ingrese cantidad de Temperaturas:
5
Ingrese Temperatura 1: 20.5
Ingrese Temperatura 2: 10
Ingrese Temperatura 3: 15
Ingrese Temperatura 4: 22
Ingrese Temperatura 5: 5
20.5
15.0
22.0

SALIDA:
-------------------------------------------------------
La media es  14.5
Total de temperaturas >= a la media es 3
```

3. Escriba un diagrama de flujo que dado un vector (máximo 200 elementos) ordenado de enteros y con posibles repeticiones de valores, obtenga como salida una lista de los números ordenados, pero sin repeticiones.

**Dato:** VEC (1..N) ; donde **1<=N<=200** y **VEC** es un arreglo unidimensional de enteros ordenados, cuya capacidad máxima es de 200 elementos.




El código del ejercicio 3 de Vectores en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Vector3: ORDENAR Y QUITAR ELEMENTOS REPETIDOS DEL VECTOR.")
print("-------------------------------------------------------")
#Inicializar
```

```
VEC = []    #Inicializamos una lista vacía

#Entradas
print("Ingrese número de elementos del vector")
N = int( input())
#Proceso
if 1 <= N and N <= 200:

    for i in range(1,N+1):
        elemento = int( input("Ingrese Entero {0}: ".format(i)))
        VEC.append(elemento)

    i = 0

    #TODO:Reportar Array como Imcompleto
    lista_nueva = []  #Una lista vacía para poner los elementos sin repetición.

    for elemento in VEC:

        #Si el elemento no esta en la lista_nueva:
        if elemento not in lista_nueva:
            lista_nueva.append(elemento) #Agregamos elemento

    #ordenamos la lista usando una función de lista en Python
    #USANDO SORT
    lista_nueva.sort()
#Salida
print("\nSALIDA: ")
print("-------------------------------------------------------")
print(lista_nueva)
```

```
-------------------------------------------------------
Vector3: ORDENAR Y QUITAR ELEMENTOS REPETIDOS DEL VECTOR.
-------------------------------------------------------
Ingrese número de elementos del vector
5
Ingrese Entero 1: 12
Ingrese Entero 2: 13
Ingrese Entero 3: 12
Ingrese Entero 4: 5
Ingrese Entero 5: 4
```

```
SALIDA:
-------------------------------------------------------
[4, 5, 12, 13]
```

## Matrices

1. Realizar el Pseudocódigo para sumar 2 matrices.

   Sean 2 matrices A y B ambas bidimensionales, para que se puedan sumar deben poseer el mismo número de filas y columnas, cada elemento de la matriz resultante será la suma de los correspondientes elementos de las matrices A y B.

   $$C(i , j) = A (i , j) + B (i , j)$$

   La matriz resultante es "C".

```
Algoritmo Suma de Matrices
Var
   A, B, C : Array (1..100, 1..100 ) de entero
   R, S, i, j : Entero
Inicio
 Escribir ("Ingrese dimensión de la matriz, máximo
  100")
 Escribir ("Dimensión S")
 Leer (S)
 Escribir ("Dimensión R")
 Leer (R)
 //Ingreso de datos en la matriz A yB
 Desde i ← 1 hasta S
     Desde j ← 1 hasta R
        Leer ( A(i,j) )
        Leer ( B(i,j) )
        C (i,j) ← A(i,j) + B(i,j)
     Fin desde
 Fin desde
 //Finalmente se imprimirá el resultado
Fin
```

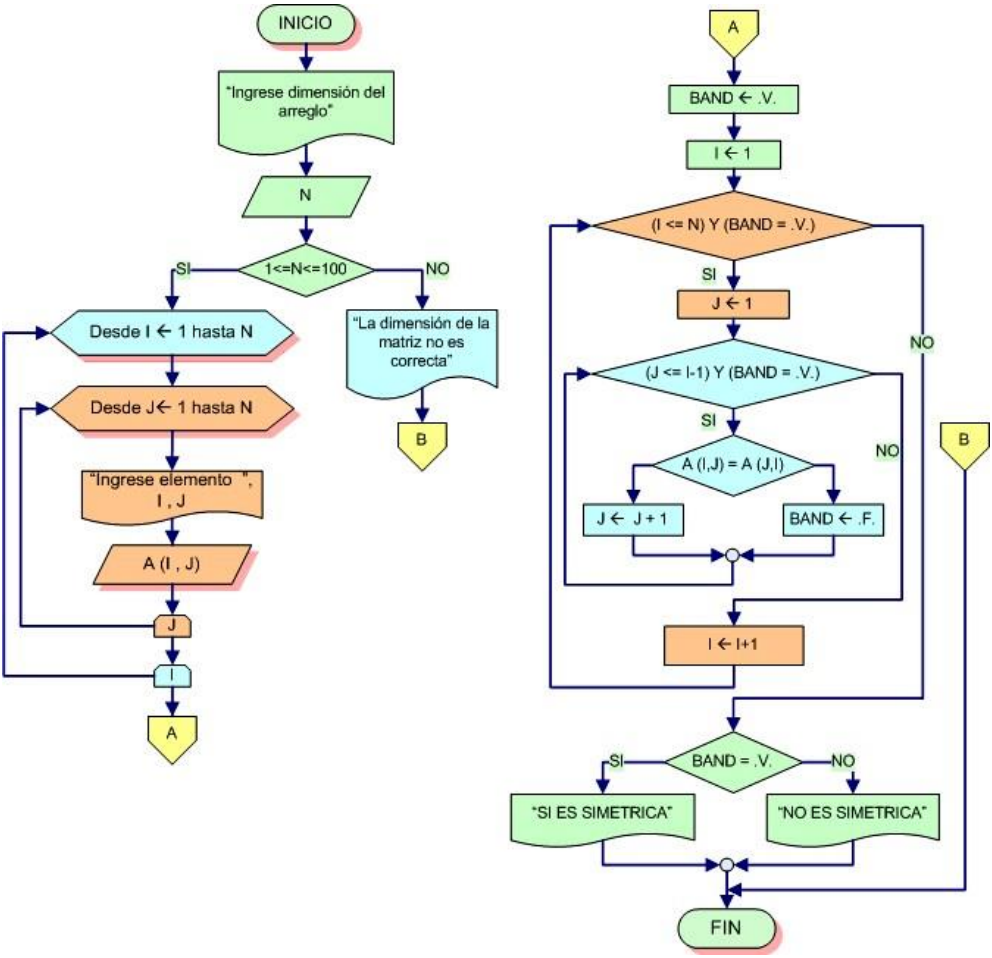
El código del ejercicio 1 de Matrices en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("Matriz1: SUMA DE MATRICES.")
print("-----------------------------------------------------")

#Inicializar
A = []
B = []
C = []

#Entradas
print ("Ingrese dimensión de la matriz,máximo 100")
S = int( input("Número de Filas: "))
R = int( input("Número Columnas: "))

#Proceso
for  i in range(S):
   A.append( [] )    #Agregamos una i fila vacía en A
   B.append( [] )    #Agregamos una i fila vacía en B
   C.append( [] )    #Agregamos una i fila vacía en C
   for j in  range(R):
      A[i].append( int( input("A{}{}: ".format(i+1,j+1))))
      B[i].append( int( input("B{}{}: ".format(i+1,j+1))))
      C[i].append( A[i][j] + B[i][j])


#Salida
print("\nSALIDA: ")
print("-----------------------------------------------------")
print(A)
print(B)
print(C)
```

```
-------------------------------------------------------
Matriz1: SUMA DE MATRICES.
-------------------------------------------------------
Ingrese dimensión de la matriz, máximo 100
Número de Filas: 3
Número Columnas: 2
A11: 1
B11: 2
A12: 2
B12: 3
A21: 3
B21: 4
A22: 4
B22: 5
A31: 5
B31: 6
A32: 6
B32: 7

SALIDA:
-------------------------------------------------------
Matriz A: [[1, 2], [3, 4], [5, 6]]
Matriz B: [[2, 3], [4, 5], [6, 7]]
Matriz C: [[3, 5], [7, 9], [11, 13]]
```

2. Dada una matriz cuadrada A, construya un diagrama de flujo que permita determinar si dicha matriz es simétrica. Se considera a una matriz simétrica si A (i , j) = A (j , i) y esto se cumple para todos los elementos i , j de la matriz.

**Dato:** A (1..N, 1..N) donde 1<= N <=100 y A es un arreglo bidimensional de tipo entero.



El código del ejercicio 2 de Matrices en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Matriz2: VERIFICAR SI LA MATRIZ ES SIMÉTRICA.")
print("-------------------------------------------------------")
#Inicializar
A = []
#Entradas
print("Ingrese dimensiones del arreglo")
N = int( input())
#Proceso
if 1 <= N and N <= 100:

    #Ingresar datos al array
    for i in range(N):
        A.append( [] )      #Se agrega la fila i

        for j in range(N):
            elemento = input( "A{}{}: ".format(i, j) )
            A[i].append( int(elemento))
    BAND = True
    i = 0
    while i < N and BAND == True:

        j = 0
        while j < i-1 and BAND == True:

            if A[i][j] == A[j][i]:
                j = j + 1
            else:
                BAND = False
        i = i + 1
    if BAND:
        print("SI ES SIMÉTRICA")
    else:
        print("NO ES SIMÉTRICA")
else:
    print("La dimensión de la matriz no es correcta.")
```

```
-------------------------------------------------------
Matriz2: VERIFICAR SI LA MATRIZ ES SIMÉTRICA.
-------------------------------------------------------
Ingrese dimensiones del arreglo
3
A00: 1
A01: 2
A02: 3
A10: 2
A11: 5
A12: 0
A20: 3
A21: 0
A22: 5
SI ES SIMÉTRICA
```

3. Una empresa automotriz necesita un programa para manejar los montos de ventas de sus N sucursales, a lo largo de los últimos M años. Los datos son dados de esta forma: M, N

| | | | |
|---|---|---|---|
| $MONTO_{1\,1}$ | $MONTO_{1\,2}$ | ….. | $MONTO_{1\,N}$ |
| $MONTO_{2\,1}$ | $MONTO_{2\,2}$ | ….. | $MONTO_{2\,N}$ |
| $MONTO_{M\,1}$ | $MONTO_{M\,2}$ | ….. | $MONTO_{M\,N}$ |

Donde:

M es una variable entera que representa el número de años entre 1 y 30 inclusive.

N es una variable entera que representa el número de sucursales de la empresa entre 1 y 35 inclusive.

$MONTO_{i\,j}$ Variable real (matriz de 2 dimensiones) representa lo que se vendió en el año I en la sucursal J

La información que necesitan los directores de la empresa para tomar decisiones es la siguiente:

a. Sucursal que más ha vendido en los M años.

b. Promedio de ventas por año.

c. Año con mayor promedio de ventas.

```
Algoritmo Automotriz
Var
    MONTO : array (1..30, 1..35) de real
    I,J , SUC , AÑO  : Entero
    SUMA, MAX, PROM : Real
Inicio
Escribir ("Ingrese numero de sucursales y años")
Leer ( N , M)
Desde I ← 1 hasta M hacer
    Desde J ← 1 hasta N hacer
        Escribir ("Ingrese ventas de la sucursal", J , "en el año". I )
        Leer ( MONTO(I , J) )
   Fin desde
Fin desde

MAX ← 0
Desde J ← 1 hasta N hacer
    SUMA ← 0
    Desde I ← 1 hasta M hacer
        SUMA ← SUMA + MONTO (I , J)
    Fin desde
    Si SUMA > MAX entonces
        MAX ← SUMA
        SUC ← J
    Fin si
Fin desde
Escribir ("Sucursal que mas vendió", SUC )

MAX ← 0
Desde I ← 1 hasta M hacer
    SUMA ← 0
    Desde J ← 1 hasta N hacer
        SUMA ← SUMA + MONTO (I , J)
    Fin desde
    PROM ← SUMA / N
    Escribir ("Promedio de ventas del año" , I, " es ", PROM)
    Si PROM > MAX entonces
        MAX ← PROM
        AÑO ← I
    Fin si
Fin desde
Escribir ("Año con mayor promedio", AÑO )

Fin
```

El código del ejercicio 3 de Matrices en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Matriz3: SUCURSALES DE UNA EMPRESA.")
print("-------------------------------------------------------")

#Inicializar
MONTO = []  #Se crea una lista vacía
#Entradas
print("Ingrese número de sucursales y años: ")
N = int( input("Número de Sucursales: "))
M = int( input("Número de Años: "))

for i in range(M):
   MONTO.append( [] )  #Se Agrega la fila i

   for j in range(N):
      print("Ingrese ventas de la sucursal", j+1 , "en el año", i+1 )
      venta = int( input())
      MONTO[i].append(venta)

#Proceso
print("\nSUCURSAL CON MÁS VENTAS: ")
print("-------------------------------------------------------")
MAX = 0
for j in range(N):
   SUMA = 0
   for i in range(M):
      SUMA = SUMA + MONTO[i][j]

   print ("Número de ventas de la Sucursal" , j+1, "es", SUMA)
   if SUMA > MAX :
      MAX  = SUMA
      SUC = j + 1     #Se incrementa j, por conteo desde 0

print("Sucursal que más vendió: ", SUC)


print("\nPROMEDIO DE VENTAS DEL AÑO: ")
```

```
print("-------------------------------------------------------")
MAX = 0
for i in range(M):
    SUMA = 0

    for j in range(N):
        SUMA = SUMA + MONTO[i][j]
    PROM = SUMA/N
    print ("Promedio de ventas del año" , i+1, "es", PROM)

    if PROM > MAX :
        MAX = PROM
        ANIO = i + 1    #Se incrementa i, por conteo desde 0

print("Año con mayor promedio", ANIO)
```

```
-------------------------------------------------------
Matriz3: SUCURSALES DE UNA EMPRESA.
-------------------------------------------------------
Ingrese número de sucursales y años:
Número de Sucursales: 2
Número de Años: 3
Ingrese ventas de la sucursal 1 en el año 1
300
Ingrese ventas de la sucursal 2 en el año 1
400
Ingrese ventas de la sucursal 1 en el año 2
250
Ingrese ventas de la sucursal 2 en el año 2
350
Ingrese ventas de la sucursal 1 en el año 3
350
Ingrese ventas de la sucursal 2 en el año 3
500

SUCURSAL CON MÁS VENTAS:
-------------------------------------------------------
Número de ventas de la Sucursal 1 es 900
Número de ventas de la Sucursal 2 es 1250
Sucursal que más vendió: 2

PROMEDIO DE VENTAS DEL AÑO:
```

```
-------------------------------------------------------
Promedio de ventas del año 1 es 350.0
Promedio de ventas del año 2 es 300.0
Promedio de ventas del año 3 es 425.0
Año con mayor promedio 3
```

## 6.5 Ejercicios Complementarios

1) Escribir un algoritmo que calcule el producto escalar y vectorial de dos vectores de 3 elementos cuyos valores se introducen por pantalla.

   Pseudocódigo.

```
Algoritmo productoescalar
   Var
      V1: array(1..3) de entero
      V2: array(1..3) de entero
      sum,x,y,z,i,P:entero
   Inicio
      Para i←1 hasta 3
         Leer (V1(i))
      Fin para
      Para i←1 hasta 3
         Leer (V2(i))
      Fin para
      sum← 0
      Para i←1 hasta 3
         P=V1(i)*V2(i)
         sum←sum+P
      Fin para
      Escribir ("El producto escalar es:", sum)
      x←V1[2]* V2[3]- V1[3]* V2[2]
      y← V1[1]* V2[3]- V1[3]* V2[1]
      z← V1[1]* V2[2]- V1[2]* V2[1]
      Escribir ("El producto vectorial es: ",x,"i - ",y,"j + ", z," k")
   Fin
```

El

Código del complementario 1 en Python es el siguiente:

```
# -*- coding: utf-8 -*-
#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento1: PRODUCTO VECTORIAL Y ESCALAR.")
print("-------------------------------------------------------")

#Inicializa

#Para inicializar vectores se puede usar:
#La operación lista por vector
#N*[ALGO] o [ALGO]*N,
#N: Es un escalar
#ALGO: Debe representar un valor por defecto

V1 = 3*[0]  #Se crea el sigue vector: [0, 0, 0]
V2 = [0]*3  #Idéntico al anterior: [0, 0, 0]

#Entradas
for i in range(3):
    V1[i] = int( input("V1({}): ".format(i+1)))

for i in range(3):
    V2[i] = int( input("V2({}): ".format(i+1)))

#Proceso
sum = 0

for i in range(3):
    P = V1[i]*V2[i]
    sum = sum + P

print("El producto escalar es:", sum)
x = V1[1]* V2[2] - V1[2]* V2[1]
y = -( V1[0]* V2[2] - V1[2]* V2[0] )
z = V1[0]* V2[1] - V1[1]* V2[0]
print("El producto vectorial es: {}i {}j {}k".format(x, y, z))
```

```
-------------------------------------------------------
Complemento1: PRODUCTO VECTORIAL Y ESCALAR.
-------------------------------------------------------
```

```
V1(1): 1
V1(2): 2
V1(3): 3
V2(1): -1
V2(2): 1
V2(3): 2
El producto escalar es: 7
El producto vectorial es: 1i -5j 3k
```

2) Escribir un algoritmo que pida un vector de caracteres por pantalla e invierta el orden de los caracteres mostrándolo por pantalla. La inversión se hará sin utilizar otro vector auxiliar.

Pseudocódigo.

```
Algoritmo inv_caract
    Var
        v: array (1..n) de caracter
        n, i, z, d: entero
    Inicio
        Leer (n)
        Para   i←1 hasta n
            Leer (v(i))
        Fin para
        z← " "
        d← n
        Para i← 1 hasta n div 2
                z← v(i)
                v(i)← v(d)
                v(d)← z
                d← d – 1
        Fin para
        Para  i← 1 hasta n
            Escribir (v(i))
        Fin para
    Fin
```

El código del complementario 2 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento2: INVERTIR VECTOR DE CARACTERES.")
print("-------------------------------------------------------")

#Entradas
print("Ingrese dimensión del vector: ");
n = int( input())
v = n*['']  #Se inicializa un vector con valores por defecto

for i in range(n):
    v[i] = input("Ingrese Caracter: ")


#Proceso
z = ''
d = n

for i in range(n//2):
    z = v[i]
    v[i] = v[d-1]
    v[d-1] = z
    d = d - 1

#Salida
for i in range(n):
    print(v[i])
```

```
-------------------------------------------------------
Complemento2: INVERTIR VECTOR DE CARACTERES.
-------------------------------------------------------
Ingrese dimensión del vector:
10
Ingrese Caracter: c
Ingrese Caracter: a
Ingrese Caracter: r
Ingrese Caracter: p
Ingrese Caracter: e
```

```
Ingrese Caracter: t
Ingrese Caracter: i
Ingrese Caracter: t
Ingrese Caracter: a
Ingrese Caracter: s
s
a
t
i
t
e
p
r
a
c
```

3) Escribir un algoritmo que calcule los números primos de 0 a 100 utilizando el llamado método de la criba de Eratóstenes. Este método consiste en definir e inicializar con todos sus elementos a True un vector de 100 elementos binarios e ir "tachando" (pasando a False) en pasadas sucesivas todos los múltiplos de los números primos (2, 3, 5, 7...) hasta obtener sólo los números primos. Es decir:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | ... |

En el ejemplo en gris claro se señalan los múltiplos de 2, mientras que en gris oscuro los múltiplos de 3 (que no son múltiplos de 2).

Nota: se emplearán 2 vectores, uno para los números y el otro para los valores booleanos.

Pseudocódigo.

```
Algoritmo primos
    Var
        N: array (1..100) de entero
        B: array  (1..100) de  lógico
        i,j: entero
    Inicio
        Desde i← 1 hasta 100
            N(i)← i
        Fin desde
        N(1)← 0
        Desde i← 2 hasta 99
            Desde j← i+1 hasta 100
                Si (N(j) mod N(i) =0) entonces
                        B(j)← 0
                Fin si
            Fin desde
        Fin desde
        Desde i← 1 hasta 100
            Si (B(i)=1) Entonces
                Escribir (N(i))
            Fin si
        Fin desde
    Fin
```

El código del complementario 3 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento3: CRIBA DE ERATÓSTENES.")
print("-------------------------------------------------------")

#inicializar

#LLenar B con True, suponiendo que todos son primos
B = 100*[True]

N = []
#Llenar N con elementos de 1 a 100
```

```
for i in range(1, 100+1):
    N.append(i)

#Proceso
#como 1 no es primo, y se encuentra la posicion 0
B[0] = False

for i in range(1, 99):

    for j in range(i+1, 100):

        if N[j] % N[i] == 0:
            B[j] = False


for i in range(100):
    #Si el valor de B[i] = 1
    if B[i]:
        print(N[i])
```

```
#Salida
-------------------------------------------------------
Complemento3: CRIBA DE ERATÓSTENES.
-------------------------------------------------------
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
```

```
67
71
73
79
83
89
97
```

4) Realizar un algoritmo que maneje un vector de enteros a través de un menú con seis opciones:

   1.- Añadir un elemento al vector (comprobando que el vector no esté lleno)

   2.- Eliminar un elemento del vector (comprobando que el vector no esté vacío)

   3.- Listar el contenido del vector

   4.- Contar las apariciones de un número en el vector

   5.- Calcular la media y el máximo de los elementos del vector

   0.- Terminar

Pseudocódigo.

```
Algoritmo  opciones
    Var
        V:array(1..100) real
        i, j, a,nn, c, num,max, ma: real
        n,opc:entero
    Inicio
        i← 0
        Escribir ("Ingrese tamaño del vector")
        Leer (n)
        Repetir
        Escribir("Ingrese 1 para añadir un elemento al vector”)
        Escribir(“Ingrese 2 para eliminar un elemento del vector”)
        Escribir(“Ingrese 3 para listar el contenido del vector ”)
        Escribir(“Ingrese 4 para contar las  apariciones de Un número en el vector”)
        Escribir(“Ingrese 5 para calcular la media y el máximo de los elemnetos de un vector”)
        Escribir(“Ingrese 0 para terminar")
        Leer (opc)
            Si opc igual
            1:  Si (i < n) entonces
                i← i + 1
                Leer (V(i))
                Fin si
            2:  Escribir ("Ingrese el número  que desea eliminar")
                Leer (num)
                Si (num> 0) entonces
                    a← 0
                    Desde j← 1 hasta i
                        Si a(j) = nn entonces
                            a← j
                            j← i
                        Fin si
                    Fin desde
                    Si a> 0 y a<= i entonces
                        Desde j ← a hasta i – 1
                            V(j) ← V(j + 1)
                        Fin desde
                        V(i) ← 0
                        i← i – 1
                    Fin si
                Fin si
            3:  Si i > 0 entonces
                    Desde j ← 1 hasta i
                        Escribir (a(j))
                    Fin desde
                Fin si
            4:  c← 0
                Escribir ("Ingrese numero para contar número de apariciones")
                Leer (nn)
                Desde j← 1 hasta i
                    Si nn = a(j) entonces
                     c← c + 1
                    Fin_si
                Fin_ desde
                Escribir (" El número de apariciones es: " , c)
            5:  max← a(1)
                ma← 0
                Desde j ← 1 hasta i
                    Si max<a(j) entonces
                     max← a(j)
                    Fin si
                    ma← ma + a(j)
                Fin desde
                ma← ma / i
                Escribir ("El maximo es:", max, " y la media es: ", ma)
            0:  Fin Selector  hasta que
                opc← 0
    Fin
```

El código del complementario 4 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento4: INTERACTUAR CON UN VECTOR A TRAVEZ DE OPC
IONES.")
print("-------------------------------------------------------")

#Inicializar
V = 100*[0]     #Inicializamos un vector con 100 elementos por defecto
i = 0

#Entradas
print ("Ingrese tamaño del vector")
n = int( input())
opc = -1
while opc != 0:
   print("\n-------------------------------------------------------")
   print("Ingrese 1 para añadir un elemento al vector")
   print("Ingrese 2 para eliminar un elemento del vector")
   print("Ingrese 3 para listar el contenido del vector ")
   print("Ingrese 4 para contar las apariciones de Un número en el vector")
   print("Ingrese 5 para calcular la media y el máximo de los elementos de un vector
")
   print("Ingrese 0 para terminar")

   #Leer Opcion
   opc = int( input("Ingrese Opción: "))
   #print("\n")

   if opc == 1:
      if (i < n) :
         #Agraga en la posicion 0 y luego incrementa i
         V[i] = int( input("IngreseEntero: "))
         i = i + 1

   elif opc == 2:
```

```
    print("Ingrese el número que desea eliminar")
    num = int( input())

    if num > 0:
        a = 0
        #Busca NUM en el array
        for j in range(i):
            if V[j] == num :
                a = j
                break

        #Re-organizar el array
        if a >= 0  and a <= i:
            for j in range(a, i-1 +1):
                V[j] = V[j+1]

            V[i] = 0
            i = i - 1


elif opc == 3:
    if i > 0:
        for j in range(i):
            print(V[j])

elif opc == 4:
    c = 0
    print("Ingrese numero para contar número de apariciones: ")
    num = int( input())

    for j in range(i):
        if num == V[j] :
            c = c + 1

    print(" El número de apariciones es: ", c)

elif opc == 5:
    max = V[0]
    ma = 0
    for j in range(i):
        if max < V[j]:
            max = V[j]
```

```
        ma = ma + V[j]

    ma = ma/i
    print("El maximo es:", max, " y la media es: ", ma)

elif opc == 0:
    print("FIN DE PROGRAMA")
    break
```

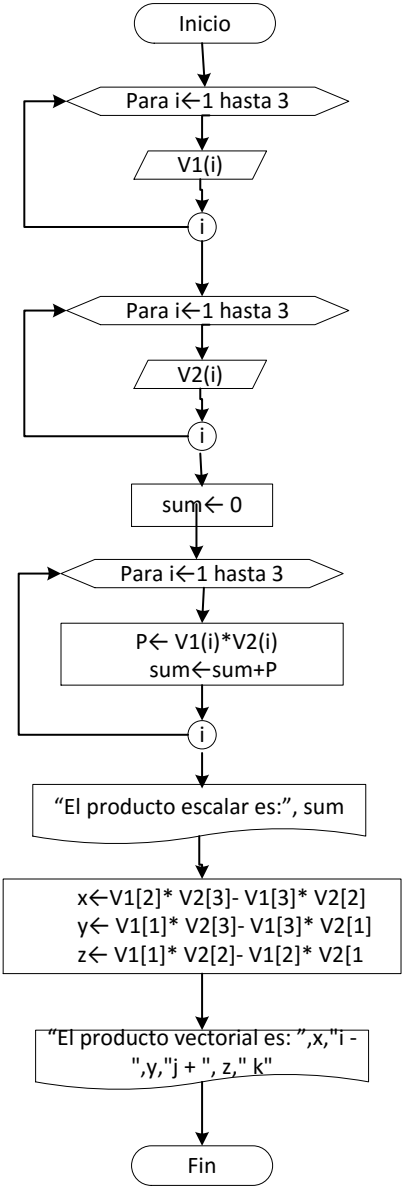
```
-------------------------------------------------------
Complemento4: INTERACTUAR CON UN VECTOR A TRAVÉS DE
OPCIONES.
-------------------------------------------------------
Ingrese tamaño del vector
10

-------------------------------------------------------
Ingrese 1 para añadir un elemento al vector
Ingrese 2 para eliminar un elemento del vector
Ingrese 3 para listar el contenido del vector
Ingrese 4 para contar las apariciones de Un número en el vector
Ingrese 5 para calcular la media y el máximo de los elementos de un vector
Ingrese 0 para terminar
Ingrese Opción: 1
IngreseEntero: 5
```

5) Elaborar algoritmo que busque en forma secuencial un VALOR dentro de un arreglo de N elementos numéricos y retorne su posición.

Pseudocódigo.

```
Algoritmo posicion
    Var
        valor:array(1..m)  de entero
        i,m,r:entero
    Inicio
        Escribir ("Ingrese numero de elementos del arreglo")
        Leer (m)
        Escribir ("Ingrese los elementos del arreglo")
        Para i← 1 hasta m
            Leer (valor(i))
        Fin Para
        //Búsqueda del valor dentro del arreglo
        Escribir ("Ingrese el valor buscado")
        Leer bus
        Escribir ("La posición del valor buscado será:")
        Para i← 1 hasta m
            Si valor(i)=bus entonces
                r← i
                Escribir (r)
            Fin Si
        Fin Para
    Fin
```

Diagrama de Flujo.



El código del complementario 5 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento5: BUSCAR ELEMENTO DENTRO DEL ARRAY.")
print("-------------------------------------------------------")

#Inicializar
valor = []

#Entradas
print ("Ingrese número de elementos del arreglo")
m = int( input())
print ("Ingrese los elementos del arreglo")
for i in range(m) :
    elemento = int( input("Ingrese Elemento: "))
    valor.append(elemento)

#Búsqueda del valor dentro del arreglo
print ("Ingrese el valor buscado: ")
bus = int( input())

print ("La posición del valor buscado será:")
for i in range(m) :
    if valor[i] == bus :
        r =  i
        print ("La posición del elemento es", r+1)
        break
```

```
-------------------------------------------------------
Complemento5: BUSCAR ELEMENTO DENTRO DEL ARRAY.
-------------------------------------------------------
Ingrese número de elementos del arreglo
6
Ingrese los elementos del arreglo
Ingrese Elemento: 2
Ingrese Elemento: 5
Ingrese Elemento: 3
```

```
Ingrese Elemento: 6
Ingrese Elemento: 7
Ingrese Elemento: 8
Ingrese el valor buscado:
6
La posición del valor buscado será:
La posición del elemento es 4
```

6) Elaborar un algoritmo que ordene descendentemente un vector.

Pseudocódigo.

```
Algoritmo desc
    Var
        v1:array(1..n) de cadena
        m,x,c:entero
    Inicio
        Escribir ("Ingrese cantidad de elementos del vector")
        Leer n
        Para x←1 hasta n
            Escribir ("Ingrese los valores del vector:")
            Leer (v1(x))
        Fin para
        Para x←1 hasta n
            Escribir ("Los elementos del vector son:")
            Para x←2 hasta n
                Para x← 1 hasta n-1
                    si v1(x)>v1(x+1)
                        m← v1(x)
                    sino
                        m← v1(x+1)
                    fin si
                Fin para
            Fin para
            Escribir (v1(x))
        Fin para
    Fin
```

El código del complementario 6 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento6: ORDENAR DESCENDENTEMENTE EL VECTOR.")
print("-------------------------------------------------------")

#Inicializar
V1 = []

#Entradas
print ( "Ingrese cantidad de elementos del vector: ")
n = int( input())

print ( "Ingrese los valores del vector: ")
for x in range(n):
    valor = int( input("V{}: ".format(x+1)))
    V1.append(valor)

#Proceso
print ( "Los elementos del vector son: ")
for y in range(n):

    for x in range(n-1):
        if V1[x] < V1[x+1]:
            m = V1[x]
            V1[x] = V1[x+1]
            V1[x+1] = m

print(V1)
```

```
-------------------------------------------------------
Complemento6: ORDENAR DESCENDENTEMENTE EL VECTOR.
-------------------------------------------------------
Ingrese cantidad de elementos del vector:
8
Ingrese los valores del vector:
V1: 2
V2: 30
```

```
V3: 40
V4: 5
V5: 8
V6: 15
V7: 35
V8: 21
Los elementos del vector son:
[40, 35, 30, 21, 15, 8, 5, 2]
```

7) Realizar un algoritmo que introduzca un nuevo elemento en un vector ordenado ascendentemente, el vector debe conservar el orden.

Pseudocódigo.

```
Algoritmo añadir
    Var
        V: array (1..100) de entero
        x,c,y,a:entero
    Inicio
        Escribir ("Ingrese los 10 valores del vector")
        Leer (V(11))
        Para x←1 hasta 10
            Para V←1 hasta 11
                Si c(y) > vec (x) entonces
                    a←  V(y)
                    V(y) ← V(x)
                    V(x) ← a
                Fin si
            Fin para
        Fin para
        Para x ← 1 hasta 10
            Escribir (V(x))
        Fin para
    Fin
```

El código del complementario 7 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento7: INGRESAR VALOR EN EL VECTOR ASCENDENTE.")
print("-------------------------------------------------------")

#Inicializar
V = []
#Entradas
print("Ingrese los 10 valores del vector: ")
for i in range(10):
    valor = int( input("valor {}: ".format(i+1)))
    V.append(valor)

valor11 = int( input("Ingrese valor a insertar: "))
V.append(valor11)


#Proceso
#Ordenar el vector
for x in range(11):
    for y in range(10):
        if V[y] > V[x]:
            a = V[y]
            V[y] = V[x]
            V[x] = a

#Mostrar los elementos del vector
for x  in range(11):
    print(V[x])
```

```
-------------------------------------------------------
Complemento7: INGRESAR VALOR EN EL VECTOR ASCENDENTE.
-------------------------------------------------------
Ingrese los 10 valores del vector:
valor 1: 1
valor 2: 20
valor 3: 27
valor 4: 28
valor 5: 5
valor 6: 6
valor 7: 2
valor 8: 8
valor 9: 11
valor 10: 45
Ingrese valor a insertar: 15
1
2
5
6
8
11
15
20
27
28
45
```

8) Cargar un vector de 100 posiciones con numero enteros, a partir de este crear 2 vectores; uno con los números pares y el otro con los numero impares, además decir de los vectores cual es más grande y el número de elementos en cada vector. Inicialmente los vectores estarán limpios, esto quiere decir que todas las posiciones tendrán el valor 0 (cero).

Pseudocódigo.

```
Algoritmo Par_im
    Var
        vec: array  (1..100) de entero
        pares: array (1..100) de entero
        impares: array (1..100) de entero
        v, vpr, i : entero
    Inicio
        Escribir ("Ingrese los 100 valores del vector")
        Para x← 0 hasta 99
            Leer (vec (x))
        Fin para
        Para x← 1 hasta 99
            Si vec (x) mod 2 = 0 entonces
                pares(vpr)←vec(x)
                vpr←vpr + 1
            sino
                impares(i)  ←vec (x)
                i ←i + 1
            Fin si
        Fin para
        Escribir ("El vector de pares tiene" ,  vpr, "elementos")
        Escribir ("El vector  de impares tiene",i, "elementos")
        Si vpr >  i entonces
            Escribir  ("El vector de pares es el más grande")
        sino
            Si i> vpr entonces
                Escribir ("El vector de impares es el mas grande")
            Si no
                Escribir  ("Los 2 vectores son iguales en numero de elementos")
            Fin si
        Fin si
    Fin
```

El código del complementario 8 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-------------------------------------------------------")
print("Complemento8: SEPARAR PARES E IMPARES.")
print("-------------------------------------------------------")

#Inicializar
Vec = []

#Entradas
print ("Ingrese dimensión del vector: ")
tamVec = int( input())

pares = tamVec*[0]
impares = tamVec*[0]


print ( "Ingrese los", tamVec, "valores del vector")
for x in range(tamVec):
   elemento = int( input("Elemento {}: ".format(x+1)))
   Vec.append(elemento)

vpr = 0
i = 0
for x in range(tamVec):
   if Vec[x] % 2 == 0:
      pares[vpr] = Vec[x]
      vpr = vpr + 1
   else:
      impares[i] = Vec[x]
      i = i + 1

print(pares[0:vpr]) #en [0:vpr], desde el inicio hasta vpr
print ( "El vector de pares tiene" , vpr, "elementos")

print(impares[0:i])
print ( "El vector de impares tiene", i, "elementos")
```

```
if vpr > i :
   print ( "El vector de pares es el más grande")

elif vpr < i:
   print ( "El vector de impares es el mas grande")

else:
   print ( "Los 2 vectores son iguales en número de elementos")
```

```
-------------------------------------------------------

Complemento8: SEPARAR PARES E IMPARES.
-------------------------------------------------------
Ingrese dimensión del vector:
9
Ingrese los 9 valores del vector
Elemento 1: 23
Elemento 2: 4
Elemento 3: 6
Elemento 4: 13
Elemento 5: 78
Elemento 6: 96
Elemento 7: 18
Elemento 8: 32
Elemento 9: 48
[4, 6, 78, 96, 18, 32, 48]
El vector de pares tiene 7 elementos
[23, 13]
El vector de impares tiene 2 elementos
El vector de pares es el más grande
```

9) Se tiene un vector, se pide ingresar 200 nombres de animales, luego se debe buscar el nombre de un animal que se ingrese por teclado, el algoritmo debe imprimir el nombre de los animales que se encuentran al lado de la posición donde está el animal buscado, tanto a la derecha como izquierda. (Pseudocódigo)

Pseudocódigo.

```
Algoritmo Animal
    Var
        animal:array(1..200) de cadena
        x:entero
        nom:cadena
    Inicio
        Para x←1 hasta 200
            Escribir (“Ingrese los nombres de los animales:”)
            Leer (animal(x))
        Fin para
        Escribir (“Ingrese animal buscado:”)
        Leer nom
        Escribir (“Los animales serán:”)
        Para x←1 hasta 200
            Si (animal(x)=nom) entonces
                Escribir (animal(x-1))
                Escribir (animal(x+1))
            Fin si
        Fin para
    Fin
```

El código del complementario 9 en Python es el siguiente:

```
# -*- coding: utf-8 -*-

#Decoración: Nombre del Algoritmo
print("-----------------------------------------------------")
print("Complemento9: JUGANDO CON NOMBRES DE ANIMALES.")
print("-----------------------------------------------------")

#Inicializar
animal = []
#Entradas
print ("Ingresar dimensión del array: ")
tamArray = int( input())

print ( "Ingrese los nombres de los animales:")
for x  in range(tamArray):
    elemento = input( "Ingrese Animal{}: ".format(x+1) )
    animal.append(elemento)
```

```
print ( "Ingrese animal buscado:")
nom = input()
print ( "El animal tiene como vecino a:")
print("-------------------------------------------------------")
for x in range(tamArray):

    if animal[x] == nom:

        if x == 0:
            #Como es Primero: No tiene vecino Izquierdo
            #Imprimir solo el derecho
            print(animal[x+1])

        elif x == tamArray-1:
            #Como es Ultimo: No tiene vecino Derecho
            #Imprimir solo el Izquierdo
            print(animal[x-1])

        else:
            #Imprimir Izquierda y derecha
            print(animal[x-1])
            print(animal[x+1])
```

```
#Salida
-------------------------------------------------------
Complemento9: JUGANDO CON NOMBRES DE ANIMALES.
-------------------------------------------------------
Ingresar dimensión del array:
5
Ingrese los nombres de los animales:
Ingrese Animal1: gato
Ingrese Animal2: perro
Ingrese Animal3: gallo
Ingrese Animal4: caballo
Ingrese Animal5: oveja
Ingrese animal buscado:
gallo
El animal tiene como vecino a:
-------------------------------------------------------
perro
caballo
```

# Glosario

- **Algoritmo**

  Un conjunto de instrucciones redactadas en forma secuencial y representan a un modelo de solución de un determinado problema.

- **Bucle**

  Conjunto de instrucciones que se repiten según una determinada condición

- **Computadora**

  Es una maquina electrónica que procesa y acumula datos a con el apoyo de instrucciones. La estructura básica de una computadora incluye un microprocesador, memoria y dispositivos de entrada/salida, también tiene buses de comunicación que permiten la interconexión entre ellos.

- **Constante**

  Es un valor de tipo permanente, ya que no puede ser modificado, dentro de un determinado contexto. En programación, una constante es un valor que no puede ser modificado durante la ejecución de un programa, únicamente puede ser leído.

- **Dato**

  Es una representación simbólica (numérica, alfabética, algorítmica, etc.) de un atributo o variable cuantitativa o cualitativa. También se puede decir de un dato que es un hecho, elemento, etc., que requiere ser estudiado. Un dato es lo que se suministra a la computadora como entrada para ser procesado

- **Diagrama de flujo**

  Es la representación gráfica del algoritmo o proceso; usado en disciplinas como programación, administración, ingeniería y la psicología . Los diagramas de flujo emplean figuras como los rectángulos, óvalos, rombos y otras numerosas figuras para definir la secuencia de un algoritmo.

- **Estructuras Repetitivas**

  Conjunto de instrucciones de un algoritmo que se utilizan cuando se quiere que se ejecuten un cierto número finito de veces, según una determinada condición.

- **Estructuras Selectivas**

  Se utilizan para tomar decisiones lógicas un algoritmo o programa, de ahí que se suelan denominar también estructuras de decisión. Conjunto de instrucciones que se ejecutan o se realizan según una condición planteada en el algoritmo o programa.

- **Funciones**

  Una función es un subprograma que recibe argumentos o parámetros de tipo de datos y devuelve un único resultado. Las funciones definidas por el usuario se llaman funciones externas. El programa llama a la función por su nombre seguida de una lista de argumentos que deben coincidir en cantidad, tipo y orden que fue declarado.

- **Identificadores**

  Un identificador esta formado por un conjunto de caracteres alfanuméricos, que sirve para identificar las entidades de un programa (clases, funciones, variables, tipos compuestos). Cada lenguaje de programación tiene sus propias reglas que definen como pueden estar construidos.

- **Lenguaje de programación**

  Es un lenguaje formal con una estructura definida que realizar procesos mediante equipos computacionales. Un programdor escribe programas mediante instrucciones en un lenguaje de programación para después ser pruebado, depurado y compilado si fuera necesario.

- **Lenguaje de programación Delphi**

  Un entorno de desarrollo de software diseñado para la programación de propósito general con entornos que permiten una programación visual. Delphi está basado en una versión de Pascal denominada Object Pascal

- **Lenguaje de programación Java**

  Lenguaje de programación de propósito general, orientado a objetos. Los programadores escribin el código de un programa una vez, y éste, pude ejecutarse en cualquier dispositivo, esto es posible gracias a la **Máquina Virtual de Java** (JVM).

- **Lenguaje de programación Python**

  Es un lenguaje de programación interpretado de propsosito general, cuya filosofía hace hincapié en la legibilidad de su código; lenguaje de programación multiparadigma, ya que soporta orientación a objetos y programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

- **Matrices**

  Una matriz es una tabla ordenada de número, también se les llama "arreglos multidimensionales", son una estructura de datos bastante similar a los vectores o arreglos.

- **Postulados**

  Un postulado es una proposición que presenta una verdad sin demostraciones ni evidencias, pero que se acepta, ya que no existe otro principio al que pueda ser referida pero, hay la necesidad de emplearlo en un razonamiento posterior.

- **Procedimientos**

  Un procedimiento es un subprograma que ejecuta una tarea especifica. Está compuesto por un conjunto de instrucciones o sentencias. Son subprogramas, y su tarea se ejecuta cuando es llamado desde cualquier parte del programa. Son declarados obligatoriamente antes de que puedan ser llamados en el cuerpo del programa principal.

- **Programa**

  Es un conjunto de pasos lógicos escritos en un lenguaje de programación que nos permite realizar una tarea específica. Un programa es escrito en un lenguaje de programación que podría ser: C, Java, PHP, Python, y otros.

- **Pseudocódigos**

  Es una forma de expresar los distintos pasos que va a realizar un programa y representa a un algorítmo o problema. El pseudocódigo es una forma de escribir los pasos que va realizar un programa de una forma más cercana al lenguaje de programación que se a utilizar posteriormente.

- **Variables**

  Está formada por un espacio en el sistema de almacenaje (memoria principal de una computadora) y esta representado por un nombre simbólico que se encuentra asociado a dicho espacio. Es una unidad

de datos que puede cambiar de valor y es de diferentes tipos (numéricos, cadena, fecha, etc.)

- **Vectores**

  Los vectores son una forma de almacenar datos que permiten contener una serie de valores del mismo tipo, cada uno de los valores contenidos tiene una posición asociada que se usará para accederlos. Está posición o índice será siempre un número entero positivo.

- **Visual Studio Code (VS)**

  VS Code es un editor de código fuente sofisticado que admite muchas funcionalidades practicas al momento de trabajar con el código.

# Bibliografía


Aguilar, L. J. (2008). *Fundamentos de programación*. McGraw-Hill Interamericana de España S.L. https://books.google.com.pe/books?id=nrNvPwAACAAJ

Cairó, O., Battistutti, O. C., & Buemo, S. G. (1993). *Estructuras de datos*. McGraw-Hill. https://books.google.com.pe/books?id=G4ecAAAACAAJ

Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. A. (2014). El lenguaje de programación Python. *Ciencias Holguín*, *XX*(2), 1-13.

Chazallet, S. (2016). *Python 3: Los fundamentos del lenguaje*. Ediciones ENI.

De la cruz Roca, M., & Condor Caballero, E. (2017). *Algoritmos para Ingenieria* (1ra ed.). Universidad Nacional José María Arguedas.

*Introducción a la Programación en Java.* (2007). ITM.

Jordi, m. G., Angela, m. P., Xavier, m. A., pau, V. A., Pere, & Fatos, X. (2006). *Programación en C++ para ingenieros*. Editorial Paraninfo.

Lecca, E. R. (1998). *Algoritmos y estructuras de datos con C/C++*. Mundigraf. https://books.google.com.pe/books?id=GpWRrgEACAAJ

*Lenguaje Logo Iii. Explorando la Programación*. (s. f.). EUNED.

Quetglás, G. M. (2003). *Introducción a la programación estructurada en C*. Universitat de València.

Regino, E. M. O. (2003). *Lógica de programación*. ECOE EDICIONES.

Vinuesa, J. M. R. (1966). *Fundamentos de programación y uso de computadores*. Universidad de chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Obras Civiles. https://books.google.com.pe/books?id=jwBdAAAAMAAJ